DIWAKAR EDUCATION HUB

Programming Languages and Computer Graphics Unit - 3

As per updated syllabus

DIWAKAR EDUCATION HUB

2020

THE LEARN WITH EXPERTIES

Language Design

Language design techniques are the most important topic to design the Programming Language and to solve various types of problems in the discipline of Comp. Science and IT. The following are the major topics covered in Language Design.

1. Programming Language Concepts
2. PL-Paradigms and Models
3. Programming Environments
4. Translation process

Programming Language Concepts

Computer programming language, any of various languages for expressing a set of detailed instructions for a digital computer. Such instructions can be executed directly when they are in the computer manufacturer-specific numerical form known as machine language, after a simple substitution process when expressed in a corresponding assembly language, or after translation from some "higher-level" language. Although there are many computer languages, relatively few are widely used.

Machine and assembly languages are "low-level," requiring a programmer to manage explicitly all of a computer's idiosyncratic features of data storage and operation. In contrast, high-level languages shield a programmer from worrying about such considerations and provide a notation that is more easily written and read by programmers.

Language Types

Machine and assembly languages

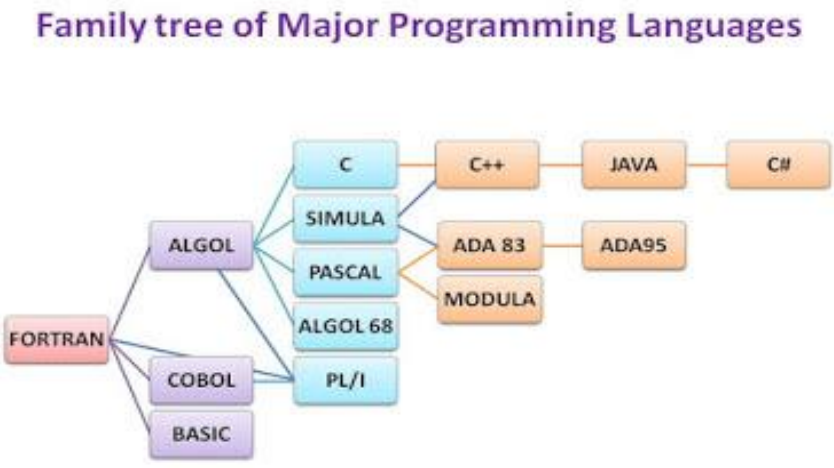
A machine language consists of the numeric codes for the operations that a particular computer can execute directly. The codes are strings of 0s and 1s, or binary digits ("bits"), which are frequently converted both from and to hexadecimal (base 16) for human viewing and modification. Machine language instructions typically use some bits to represent operations, such as addition, and some to represent operands, or perhaps the location of the next instruction. Machine language is difficult to read and write, since it does not resemble

conventional mathematical notation or human language, and its codes vary from computer to computer.

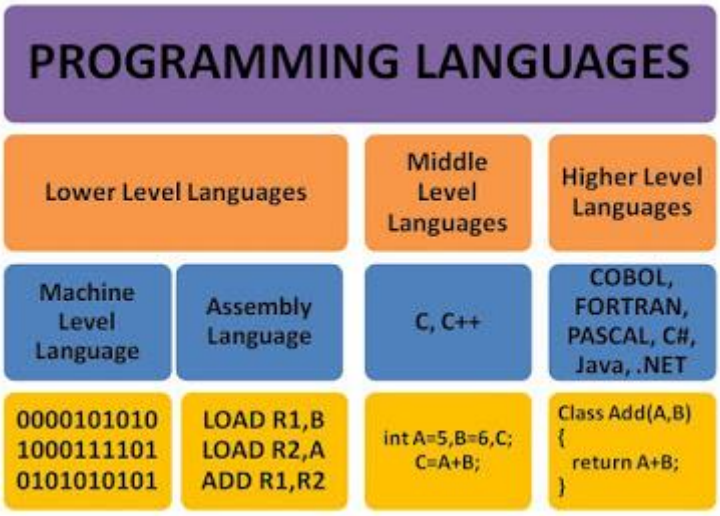
Assembly language is one level above machine language. It uses short mnemonic codes for instructions and allows the programmer to introduce names for blocks of memory that hold data. One might thus write “add pay, total” instead of “0110101100101000” for an instruction that adds two numbers.

Assembly language is designed to be easily translated into machine language. Although blocks of data may be referred to by name instead of by their machine addresses, assembly language does not provide more sophisticated means of organizing complex information. Like machine language, assembly language requires detailed knowledge of internal computer architecture. It is useful when such details are important, as in programming a computer to interact with input/output devices (printers, scanners, storage devices, and so forth).

The built-in competence and defects of the various programming languages like FORTRAN, ALGOL, COBOL, C, C++, and JAVA. List of Differences between higher and lower/machine level languages.



Basic theories like abstract syntax, interpretation, stack, heap organization, compilation techniques, different types of 'type checking' and 'error checking' various for each Programming language.



Algorithmic languages

Algorithmic languages are designed to express mathematical or symbolic computations. They can express algebraic operations in notation similar to mathematics and allow the use of subprograms that package commonly used operations for reuse. They were the first high-level languages.

FORTRAN

The first important algorithmic language was FORTRAN (*for*mula *tran*slation), designed in 1957 by an IBM team led by John Backus. It was intended for scientific computations with real numbers and collections of them organized as one- or multidimensional arrays. Its control structures included conditional IF statements, repetitive loops (so-called DO loops), and a GOTO statement that allowed nonsequential execution of program code. FORTRAN made it convenient to have subprograms for common mathematical operations, and built libraries of them.

FORTRAN was also designed to translate into efficient machine language. It was immediately successful and continues to evolve.

ALGOL

ALGOL (*algo*rithmic *l*anguage) was designed by a committee of American and European computer scientists during 1958–60 for publishing algorithms, as well as

for doing computations. Like LISP (described in the next section), ALGOL had recursive subprograms—procedures that could invoke themselves to solve a problem by reducing it to a smaller problem of the same kind. ALGOL introduced block structure, in which a program is composed of blocks that might contain both data and instructions and have the same structure as an entire program. Block structure became a powerful tool for building large programs out of small components.

ALGOL contributed a notation for describing the structure of a programming language, Backus–Naur Form, which in some variation became the standard tool for stating the syntax (grammar) of programming languages. ALGOL was widely used in Europe, and for many years it remained the language in which computer algorithms were published. Many important languages, such as Pascal and Ada (both described later), are its descendants.

LISP

LISP (*lis*t *p*rocessing) was developed about 1960 by John McCarthy at the Massachusetts Institute of Technology (MIT) and was founded on the mathematical theory of recursive functions (in which a function appears in its own definition). A LISP program is a function applied to data, rather than being a sequence of procedural steps as in FORTRAN and ALGOL. LISP uses a very simple notation in which operations and their operands are given in a parenthesized list. For example, (+ *a* (* *b c*)) stands for *a* + *b***c*. Although this appears awkward, the notation works well for computers. LISP also uses the list structure to represent data, and, because programs and data use the same structure, it is easy for a LISP program to operate on other programs as data.

LISP became a common language for artificial intelligence (AI) programming, partly owing to the confluence of LISP and AI work at MIT and partly because AI programs capable of “learning” could be written in LISP as self-modifying programs. LISP has evolved through numerous dialects, such as Scheme and Common LISP.

C

The C programming language was developed in 1972 by Dennis Ritchie and Brian Kernighan at the AT&T Corporation for programming computer operating systems. Its capacity to structure data and programs through the composition of smaller units is comparable to that of ALGOL. It uses a compact notation and provides the programmer with the ability to operate with the addresses of data as well as with their values. This ability is important in systems programming, and C shares with assembly language the power to exploit all the features of a computer's internal architecture. C, along with its descendant C++, remains one of the most common languages.

Business-oriented languages

COBOL

COBOL (*co*mmon *b*usiness *o*riented *l*anguage) has been heavily used by businesses since its inception in 1959. A committee of computer manufacturers and users and U.S. government organizations established CODASYL (*Co*mmittee on *Da*ta *Sy*stems and *L*anguages) to develop and oversee the language standard in order to ensure its portability across diverse systems.

COBOL uses an English-like notation—novel when introduced. Business computations organize and manipulate large quantities of data, and COBOL introduced the record data structure for such tasks. A record clusters heterogeneous data such as a name, ID number, age, and address into a single unit. This contrasts with scientific languages, in which homogeneous arrays of numbers are common. Records are an important example of "chunking" data into a single object, and they appear in nearly all modern languages.

SQL

SQL (structured query language) is a language for specifying the organization of databases (collections of records). Databases organized with SQL are called relational because SQL provides the ability to query a database for information that falls in a given relation. For example, a query might be "find all records with both last_name Smith and city New York." Commercial database programs commonly use a SQL-like language for their queries.

Education-oriented languages

BASIC

BASIC (beginner's all-purpose symbolic instruction code) was designed at Dartmouth College in the mid-1960s by John Kemeny and Thomas Kurtz. It was intended to be easy to learn by novices, particularly non-computer science majors, and to run well on a time-sharing computer with many users. It had simple data structures and notation and it was interpreted: a BASIC program was translated line-by-line and executed as it was translated, which made it easy to locate programming errors.

Its small size and simplicity also made BASIC a popular language for early personal computers. Its recent forms have adopted many of the data and control structures of other contemporary languages, which makes it more powerful but less convenient for beginners.

Pascal

About 1970 Niklaus Wirth of Switzerland designed Pascal to teach structured programming, which emphasized the orderly use of conditional and loop control structures without GOTO statements. Although Pascal resembled ALGOL in notation, it provided the ability to define data types with which to organize complex information, a feature beyond the capabilities of ALGOL as well as FORTRAN and COBOL. User-defined data types allowed the programmer to introduce names for complex data, which the language translator could then check for correct usage before running a program.

During the late 1970s and '80s, Pascal was one of the most widely used languages for programming instruction. It was available on nearly all computers, and, because of its familiarity, clarity, and security, it was used for production software as well as for education.

Logo

Logo originated in the late 1960s as a simplified LISP dialect for education; Seymour Papert and others used it at MIT to teach mathematical thinking to schoolchildren. It had a more conventional syntax than LISP and featured "turtle graphics," a simple method for generating computer graphics. (The name came from an early project to program

a turtlelike robot.) Turtle graphics used body-centred instructions, in which an object was moved around a screen by commands, such as “left 90” and “forward,” that specified actions relative to the current position and orientation of the object rather than in terms of a fixed framework. Together with recursive routines, this technique made it easy to program intricate and attractive patterns.

Hypertalk

Hypertalk was designed as “programming for the rest of us” by Bill Atkinson for Apple’s Macintosh. Using a simple English-like syntax, Hypertalk enabled anyone to combine text, graphics, and audio quickly into “linked stacks” that could be navigated by clicking with a mouse on standard buttons supplied by the program. Hypertalk was particularly popular among educators in the 1980s and early ’90s for classroom multimedia presentations. Although Hypertalk had many features of object-oriented languages (described in the next section), Apple did not develop it for other computer platforms and let it languish; as Apple’s market share declined in the 1990s, a new cross-platform way of displaying multimedia left Hypertalk all but obsolete (*see* the section World Wide Web display languages).

Object-oriented languages

Object-oriented languages help to manage complexity in large programs. Objects package data and the operations on them so that only the operations are publicly accessible and internal details of the data structures are hidden. This information hiding made large-scale programming easier by allowing a programmer to think about each part of the program in isolation. In addition, objects may be derived from more general ones, “inheriting” their capabilities. Such an object hierarchy made it possible to define specialized objects without repeating all that is in the more general ones.

Object-oriented programming began with the Simula language (1967), which added information hiding to ALGOL. Another influential object-oriented language was Smalltalk (1980), in which a program was a set of objects that interacted by sending messages to one another.

C++

The C++ language, developed by Bjarne Stroustrup at AT&T in the mid-1980s, extended C by adding objects to it while preserving the efficiency of C programs. It has been one of the most important languages for both education and industrial programming. Large parts of many operating systems, such as the Microsoft Corporation's Windows 98, were written in C++.

Ada

Ada was named for Augusta Ada King, countess of Lovelace, who was an assistant to the 19th-century English inventor Charles Babbage, and is sometimes called the first computer programmer. Ada, the language, was developed in the early 1980s for the U.S. Department of Defense for large-scale programming. It combined Pascal-like notation with the ability to package operations and data into independent modules. Its first form, Ada 83, was not fully object-oriented, but the subsequent Ada 95 provided objects and the ability to construct hierarchies of them. While no longer mandated for use in work for the Department of Defense, Ada remains an effective language for engineering large programs.

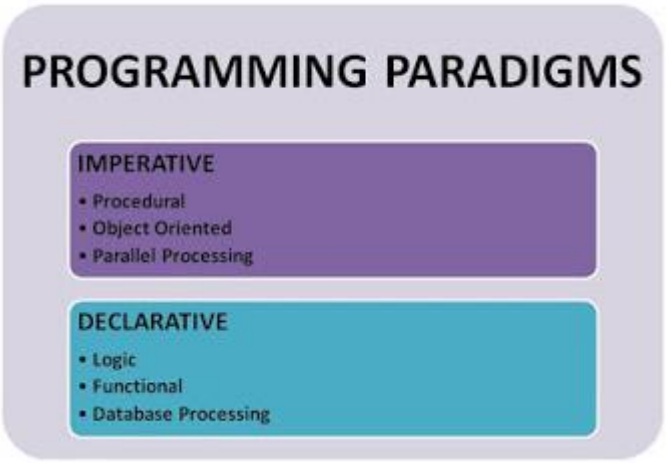
Java

In the early 1990s, Java was designed by Sun Microsystems, Inc., as a programming language for the World Wide Web (WWW). Although it resembled C++ in appearance, it was fully object-oriented. In particular, Java dispensed with lower-level features, including the ability to manipulate data addresses, a capability that is neither desirable nor useful in programs for distributed systems. In order to be portable, Java programs are translated by a Java Virtual Machine specific to each computer platform, which then executes the Java program. In addition to adding interactive capabilities to the Internet through Web "applets," Java has been widely used for programming small and portable devices, such as mobile telephones.

PL-Paradigms and Models

A programming paradigm is a various system design of ideas that have been used to guide/implement the programming.

Paradigm can also be termed as method to solve some problem or do some task. Programming paradigm is an approach to solve problem using some programming language or also we can say it is a method to solve a problem using tools and techniques that are available to us following some approach. There are lots for programming language that are known but all of them need to follow some strategy when they are implemented and this methodology/strategy is paradigms. Apart from varieties of programming language there are lots of paradigms to fulfil each and every demand.



Following list of paradigms

Imperative: Explicit sequence [such as assignment statements] of statements that change the state of the computer.

It is one of the oldest programming paradigm. It features close relation relation to machine architecture. It is based on Von Neumann architecture. It works by changing the program state through assignment statements. It performs step by step task by changing state. The main focus is on how to achieve the goal. The paradigm consist of several statements and after execution of all the result is stored.

Advantage:

1. Very simple to implement
2. It contains loops, variables etc.

Disadvantage:

1. Complex problem cannot be solved
2. Less efficient and less productive
3. Parallel programming is not possible

Examples of **Imperative** programming paradigm:

C : developed by Dennis Ritchie and Ken Thompson
Fortan : developed by John Backus for IBM
Basic : developed by John G Kemeny and Thomas E Kurtz

```
// average of five number in C

int marks[5] = { 12, 32, 45, 13, 19 } int sum = 0;

float average = 0.0;

for (int i = 0; i < 5; i++) {

   sum = sum + marks[i];

}

average = sum / 5;
```

Imperative programming is divided into three broad categories: Procedural, OOP and parallel processing. These paradigms are as follows:

Procedural: Imperative programming with procedure calls something like the re-use of code.

This paradigm emphasizes on procedure in terms of under lying machine model. There is no difference in between procedural and imperative approach. It has the ability to reuse the code and it was boon at that time when it was in use because of its reusability.

Examples of **Procedural** programming paradigm:

C : developed by Dennis Ritchie and Ken Thompson

C++ : developed by Bjarne Stroustrup

Java : developed by James Gosling at Sun Microsystems

ColdFusion : developed by J J Allaire

Pascal : developed by Niklaus Wirth

Object-Oriented: By defining the objects that transfers messages to each objects or program. Objects have their own internal state with public interface calls [encapsulation].

The program is written as a collection of classes and object which are meant for communication. The smallest and basic entity is object and all kind of computation is performed on the objects only. More emphasis is on data rather procedure. It can handle almost all kind of real life problems which are today in scenario.

Advantages:

1. Data security
2. Inheritance
3. Code reusability
4. Flexible and abstraction is also present

Parallel processing: It is a type of processing that works like a divide and conquer technique by dividing the task into multiple small processes and processing them parallel to run a program in the least time.

Parallel processing is the processing of program instructions by dividing them among multiple processors. A parallel processing system posses many numbers of processor with the objective of running a program in less time by dividing them. This approach seems to be like divide and conquer. Examples are NESL (one of the oldest one) and C/C++ also supports because of some library function.

Declarative

Programming by mentioning/specifying the value/result that what we need and mentioning how to get the result.

It is divided as Logic, Functional, Database. In computer science the *declarative programming* is a style of building programs that expresses logic of computation without talking about its control flow. It often considers programs as theories of some logic.It may simplify writing parallel programs. The focus is on what needs to be done rather how it should be done basically emphasize on what code code is actually doing. It just declare the result we want rather how it has be produced. This is the only difference between imperative (how to do) and declarative (what to do) programming paradigms. Getting into deeper we would see logic, functional and database.

Logic (as per defined rules): Specifying a set of rules/business logic in the programming. It is used to solve logic problems.

It can be termed as abstract model of computation. It would solve logical problems like puzzles, series etc. In logic programming we have a knowledge base which we know before and along with the question and knowledge base which is given to machine, it produces result. In normal programming languages, such concept of knowledge base is not available but while using the concept of artificial intelligence, machine learning we have some models like Perception model which is using the same mechanism.
In logical programming the main emphasize is on knowledge base and the problem.

sum of two number in prolog:

```
 predicates
 sumoftwonumber(integer, integer)
clauses
 sum(0, 0).
  sum(n, r):-
```

```
n1=n-1,
sum(n1, r1),
r=r1+n
```

Functional (Application-oriented): Expressing the calculations as evaluation of math-function, Function call programming with to avoid any global state change.

The functional programming paradigms has its roots in mathematics and it is language independent. The key principal of this paradigms is the execution of series of mathematical functions. The central model for the abstraction is the function which are meant for some specific computation and not the data structure. Data are loosely coupled to functions.The function hide their implementation. Function can be replaced with their values without changing the meaning of the program. Some of the languages like perl, javascript mostly uses this paradigm.

Examples of **Functional** programming paradigm:

JavaScript : developed by Brendan Eich

Haskwell : developed by Lennart Augustsson, Dave Barton

Scala : developed by Martin Odersky

Erlang : developed by Joe Armstrong, Robert Virding

Lisp : developed by John Mccarthy

ML : developed by Robin Milner

Clojure : developed by Rich Hickey

Structured: Clean programming like goto-free, with proper nested control structures.

This programming methodology is based on data and its movement. Program statements are defined by data rather than hard-coding a series of steps. A database program is the heart of a business information system and provides file creation, data entry, update, query and reporting functions. There are several

programming languages that are developed mostly for database application. For example SQL. It is applied to streams of structured data, for filtering, transforming, aggregating (such as computing statistics), or calling other programs. So it has its own wide application.

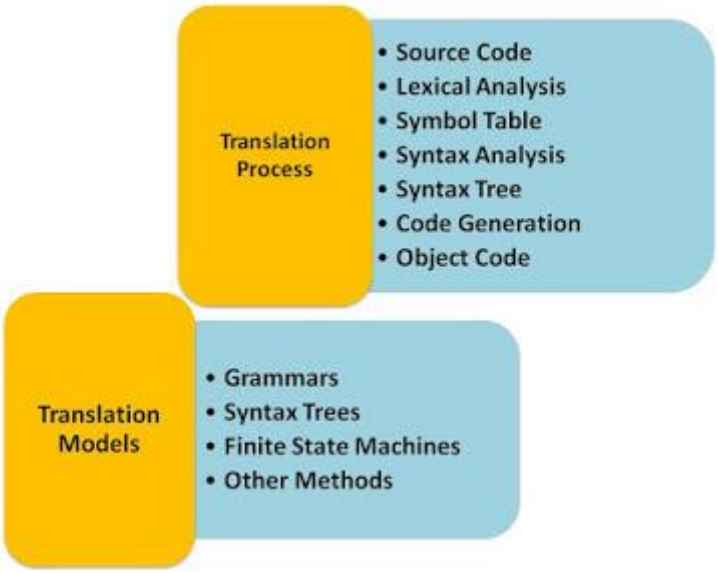
```
CREATE DATABASE databaseAddress;
CREATE TABLE Addr (
   PersonID int,
   LastName varchar(200),
   FirstName varchar(200),
   Address varchar(200),
   City varchar(200),
   State varchar(200)
);
```

Programming Environments

Environment/Setup is not a part of any Programming Language; It is a base step to be completed before start writing the program i.e. required software/hardware prerequisite.

Translation process

Translation process: A computer processor only understands the machine code. Programs are written in high-level languages such as Java cannot be run directly in the processor.



To execute a program written in any kind of language, it has to be translated to machine level language first. The source code will be translated to Machine level language / Object code by a translator.

There are *three types* of translators available

Assembles: Assemblers convert the assembly language mnemonics codes to the machine code.

Compilers: Compiler converts the full source code program to machine level code. Hence it will run in the machine without any further translation process. But, the error correction process is tough in the compilers.

Interpreters: Interpreters converts each instruction line by line into the object code [like Java].

Virtual Machine

A virtual machine (or "VM") is an emulated computer system created using software. It uses physical system resources, such as the CPU, RAM, and disk

storage, but is isolated from other software on the computer. It can easily be created, modified, or destroyed without affecting the host computer.

Virtual machines provide similar functionality to physical machines, but they do not run directly on the hardware. Instead, a software layer exists between the hardware and the virtual machine. The software that manages one or more VMs is called a "hypervisor" and the VMs are called "guests" or virtualized instances. Each guest can interact with the hardware, but the hypervisor controls them. The hypervisor can start up and shut down virtual machines and also allocate a specific amount of system resources to each one.

virtual machine can be created using virtualization software. Examples include Microsoft Hyper-V Manager, VMware Workstation Pro, and Parallels Desktop. These applications allow us to run multiple VMs on a single computer. For example, Parallels Desktop for Mac allows us to run Windows, Linux, and macOS virtual machines on our Mac.

VMs are ideal for testing software since developers can install one or more applications and revert to a saved state (or "snapshot") whenever needed. Testing software on a regular operating system can cause unexpected crashes and may leave some files lingering behind after the software is uninstalled. It is safer to test software on a virtual machine that is isolated from the operating system and can be fully reset as needed.

Binding Time

As we have just seen, operating systems use various kinds of names to refer to objects. Sometimes the mapping between a name and an object is fixed, but sometimes it is not. In the latter case, it may matter when the name is bound to the object. In general, **early binding** is simple, but is not flexible, whereas **late binding** is more complicated but often more flexible.

To clarify the concept of binding time, look at some real-world examples. An example of early binding is the practice of some colleges to allow parents to enroll a baby at birth and prepay the current tuition. When the student shows up 18

DIWAKAR EDUCATION HUB

years later, the tuition is fully paid up, no matter how high it may be at that moment.

In manufacturing, ordering parts in advance and maintaining an inventory of them is early binding. In contrast, just-in-time manufacturing requires suppliers to be able to provide parts on the spot, with no advance notice required. This is late binding.

Programming languages often support multiple binding times for variables. Global variables are bound to a particular virtual address by the compiler. This exemplifies early binding. Variables local to a procedure are assigned a virtual address (on the stack) at the time the procedure is invoked. This is intermediate binding. Variables stored on the heap (those allocated by *malloc* in C or *new* in Java) are assigned virtual addresses only at the time they are actually used. Here we have late binding.

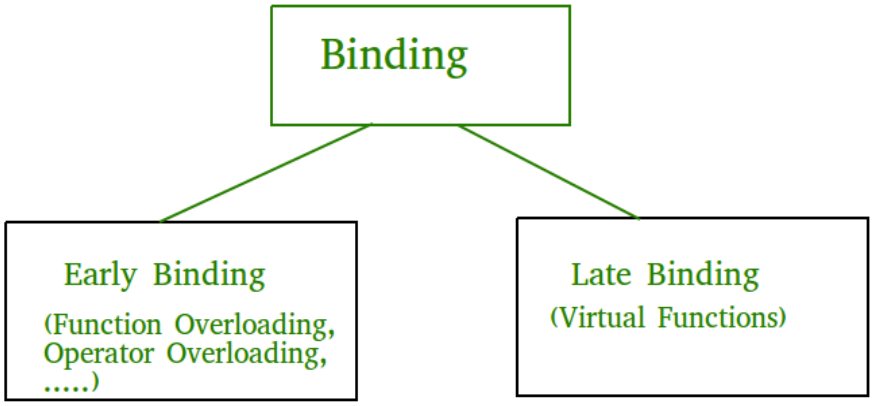
Operating systems often use early binding for most data structures, but occasionally use late binding for flexibility. Memory allocation is a case in point. Early multiprogramming systems on machines lacking address relocation hardware had to load a program at some memory address and relocate it to run there. If it was ever swapped out, it had to be brought back at the same memory address or it would fail. In contrast, paged virtual memory is a form of late binding. The actual physical address corresponding to a given virtual address is not known until the page is touched and actually brought into memory.

Another example of late binding is window placement in a GUI. In contrast to the early graphical systems, in which the programmer had to specify the absolute screen coordinates for all images on the screen, in modern GUIs, the software uses coordinates relative to the window's origin, but that is not determined until the window is put on the screen, and it may even be changed later.

Binding refers to the process of converting identifiers (such as variable and performance names) into addresses. Binding is done for each variable and

DIWAKAR EDUCATION HUB

functions. For functions, it means that matching the call with the right function definition by the compiler. It takes place either at compile time or at runtime.



Early Binding (compile-time time polymorphism) As the name indicates, compiler (or linker) directly associate an address to the function call. It replaces the call with a machine language instruction that tells the mainframe to leap to the address of the function.

By default early binding happens in C++. Late binding is achieved with the help of virtual keyword)

```
// CPP Program to illustrate early binding.
// Any normal function call (without virtual)
// is binded early. Here we have taken base
// and derived class example so that readers
// can easily compare and see difference in
// outputs.
#include<iostream>
using namespace std;
```

DIWAKAR EDUCATION HUB

```
class Base
{
public:
   void show() { cout<<" In Base \n"; }
};
class Derived: public Base
{
public:
   void show() { cout<<"In Derived \n"; }
};
int main(void)
{
   Base *bp = new Derived;
   // The function call decided at
   // compile time (compiler sees type
   // of pointer and calls base class
   // function.
   bp->show();
   return 0;
}
```

Output:

DIWAKAR EDUCATION HUB

```
In Base
```

Late Binding : (Run time polymorphism) In this, the compiler adds code that identifies the kind of object at runtime then matches the call with the right function definition. This can be achieved by declaring a virtual function.

```
// CPP Program to illustrate late binding
#include<iostream>
using namespace std;
class Base
{
public:
   virtual void show() { cout<<" In Base \n"; }
};
class Derived: public Base
{
public:
   void show() { cout<<"In Derived \n"; }
};
int main(void)
{
   Base *bp = new Derived;
   bp->show();  // RUN-TIME POLYMORPHISM
```

DIWAKAR EDUCATION HUB

```
  return 0;

}
```

Output:

```
In Derived
```

Syntax in Programming

Syntax is the set of rules that define what the various combinations of symbols mean. This tells the computer how to read the code. Syntax refers to a concept in writing code dealing with a very specific set of words and a very specific order to those words when we give the computer instructions. This order and this strict structure is what enables us to communicate effectively with a computer. Syntax is to code, like grammar is to English or any other language. A big difference though is that computers are really exacting in how we structure that grammar or our syntax.

This syntax is why we call programming coding. Even amongst all the different languages that are out there. Each programming language uses different words in a different structure in how we give it information to get the computer to follow our instructions.

Web developers primarily focus on HTML, CSS, and JavaScript. That is what we're going to focus on in this course as well. By focusing on these languages and mastering them, We'll be able to write websites that can be opened by any browser in the world.

Syntax in computer programming means the rules that control the structure of the symbols, punctuation, and words of a programming language.

Without syntax, the meaning or semantics of a language is nearly impossible to understand.

DIWAKAR EDUCATION HUB

For example, a series of English words, such as — subject a need and does sentence a verb — has little meaning without syntax.

Applying basic syntax results in the sentence — Does a sentence need a subject and verb?

Programming languages function on the same principles.

If the syntax of a language is not followed, the code will not be understood by a compiler or interpreter.

Compilers convert programming languages like Java or C++ into binary code that computers can understand. If the syntax is incorrect, the code will not compile.

Interpreters execute programming languages such as JavaScript or Python at runtime. The incorrect syntax will cause the code to fail.

That's why it is crucial that a programmer pays close attention to a language's syntax. No programmer likes to get a syntax error.

Basic Syntax

Basic syntax represents the fundamental rules of a programming language. Without these rules, it is impossible to write functioning code.

Every language has its own set of rules that make up its basic syntax. Naming conventions are a primary component of basic syntax conventions and vary by language.

1. **Case Sensitive**. Java, C++, and Python are examples of languages that are case-sensitive. Identifiers such as ***world*** and ***World*** have different meanings in these languages. Languages such as Basic and SQL are insensitive, meaning world and World have the same meaning.
2. **Class Names.** Java requires the first letter of each word in class names be upper case. For example, ***class FirstJavaClass.*** Languages such as C or C++

DIWAKAR EDUCATION HUB

use an underscore to separate words. In C, the class name would be first_java_class.

3. **Program Filenames.** The name of a Java program file must match the class name with the extension '*.java" added to the name. For example, ***FirstJavaClass.java*** would be the name of the program file for the class *FirstJavaClass*. C and C++ files require a "*.c" or "*.cpp" extension but have no other stipulations.

Different languages may have rules for adding comments, using white space, or declaring variables.

Object-oriented languages such as Java and C use methods that have different syntax requirements.

The first step in learning any programming language is to understand the basics such as phrase structure, proper syntax and correctly structured code.

Understanding Syntax

Human languages have syntax. These rules stipulate word order, punctuation and sentence structure.

Without these rules, it would be impossible to communicate in a given language. When learning a foreign language, one of the first steps is learning its syntax.

Writing code requires the same focus on syntax. Once the code is written, it is read multiple times by different people.

Sometimes the code may be read years after it is written, making coding standards necessary. Coding standards can make the code easy to understand.

C Syntax

look at a C program that prints the sentence — My first line of code. All C instructions are written in lower case.

#include

DIWAKAR EDUCATION HUB

int main() {

printf(“My first line of code”);

}

For this example, ignore the include statement at the start of the program. All C programs must begin with main() followed by a left curly bracket ({).

This convention indicates the start of the program. A right curly bracket (}) indicates the end.

The print function (printf) is called, followed by what is to print. The text to print must be surrounded by quotation marks and enclosed in parentheses.

All statements must end with a semicolon (;). For example, to print a second sentence add the following statement before the right curly bracket:

printf(“just printed!”);

}

The executed program would display: My first line of code just printed!

Why Is Syntax Important in Programming?

Syntax improves code readability. It ensures that the **four C’s of coding** are maintained:

1. **Communication**
2. **Code integration**
3. **Consistency**
4. **Clarity**

The concept behind conventions is to make the code explain itself. If the code is self-explanatory, the focus can be on design and program improvements and not on what does this mean?

DIWAKAR EDUCATION HUB

Using consistent standards means that code is predictable and discoverable when read by other programmers.

When code does not follow conventions, it becomes disorganized and difficult to read. It becomes what is known as spaghetti code.

The term has a negative connotation indicating that the programmer did not have the skills or experience needed to write readable code.

The Translation Process.

A computer system can only understand machine code. A program written for example in a high level language such as Java cannot be run directly. To execute a computer program written in any programming language, it must first be translated.

The source code which is written by the programmer needs to be translated. When translated, the source code becomes object code which is understandable by the computer system. This process can be seen in the following diagram;



There are three main types of translators; Assemblers,Compilers and Interpreters. A description of each of these type of translators is given in the following table.

The three main types of translators

Type	Description
Assembler	Assemblers convert assembly language mnemonics

DWARKA EDUCATION HUB

	into machine code.
Interpreters	Interpreters convert each instruction of the source code into the object code as the program is being run. This gives a better interactive environment but is slower.
Compilers	A compiler converts the entire source code into machine code so that it can be run on the machine without further translation. However, error correction is tedious.

Stages in Translating a Program

Lexical analysis (Scanner): Breaking a program into primitive components, called tokens (identifiers, numbers, keywords, …)

Syntactic analysis (Parsing): Creating a syntax tree of the program.

Symbol table: Storing information about declared objects (identifiers, procedure names, …)

Semantic analysis: Understanding the relationship among the tokens in the program.

Optimization: Rewriting the syntax tree to create a more efficient program.

Code generation: Converting the parsed program into an executable form.

Elementary Data Types

Variables and Constants

Variables: A variable is it Quadruple which is composed of a name,a set of attributes, a reference and a value.

DIWAKAR EDUCATION HUB

A simple variable is an elementary data object with a name and binding of data object to value may change during its lifetime. This data objects are basically defined and named by programmer explicitly.

Attribute of a variable:

example of variable in "ALGOL Language"

y:=9;

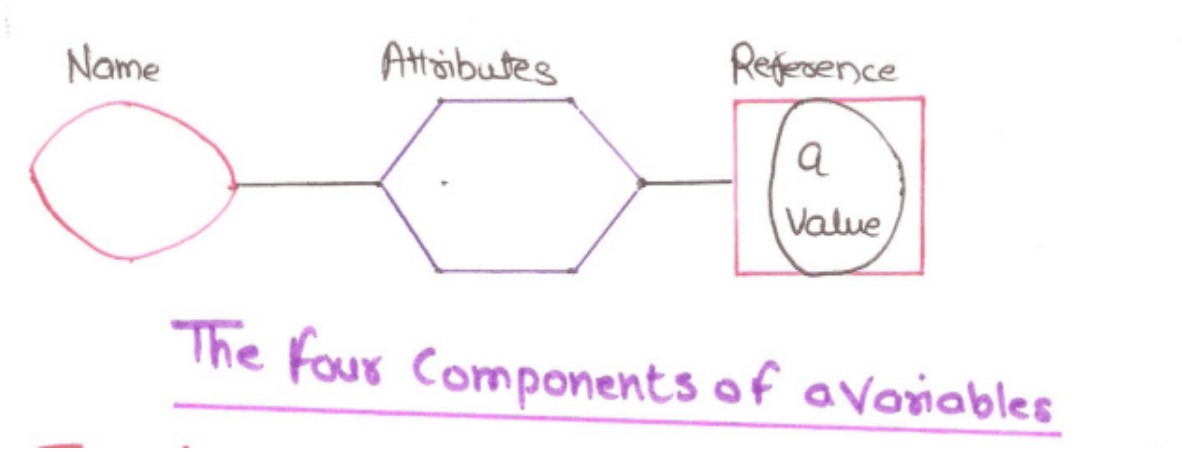
We can say that it has four attributes

- 1) the name of the box:y
- 2) The name of description of a current contents.

i.e 9 we can also say that square of 3.

- 3) the box or storage location(s) which holds(s) the value.
- 4) the content of the box or 9.

The name of the box and its storage location are fixed, but the contents and it name may vary over time.



The four Components of a Variables

another example in C language:

int N;- It declares a simple data object N of type integer.

R EDUCATION HUB

N=27; beta value 27 is assigned to variable N.

- 1) declare the variable name N of type integer.
- 2) lifetime of N is execution end.
- 3) data object bound to N during end of execution time.
- 4) value 27 is assigned and may be changed during life of N.
- 5) hidden from the programmer are other binding made by virtual computer like creating Activation Record, Storage for this activation record in Run-TimeStack etc.

Data Object:

Definition: A Data object represents a container for data values, a place where data values may be stored and later retrieved.

Definition: A runtime grouping of one or more pieces of data in a virtual computer.

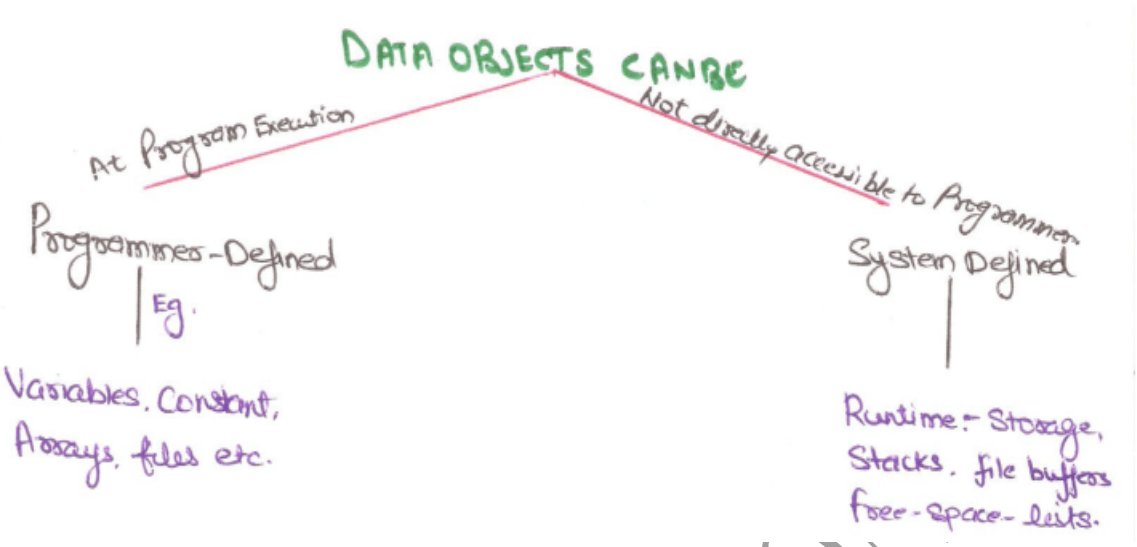
Definition: A Location in memory with an assignment name in the actual computer.

Data objects can be:

• 1) at program execution- Programmer-defined -(example variables, constant, arrays, files etc)

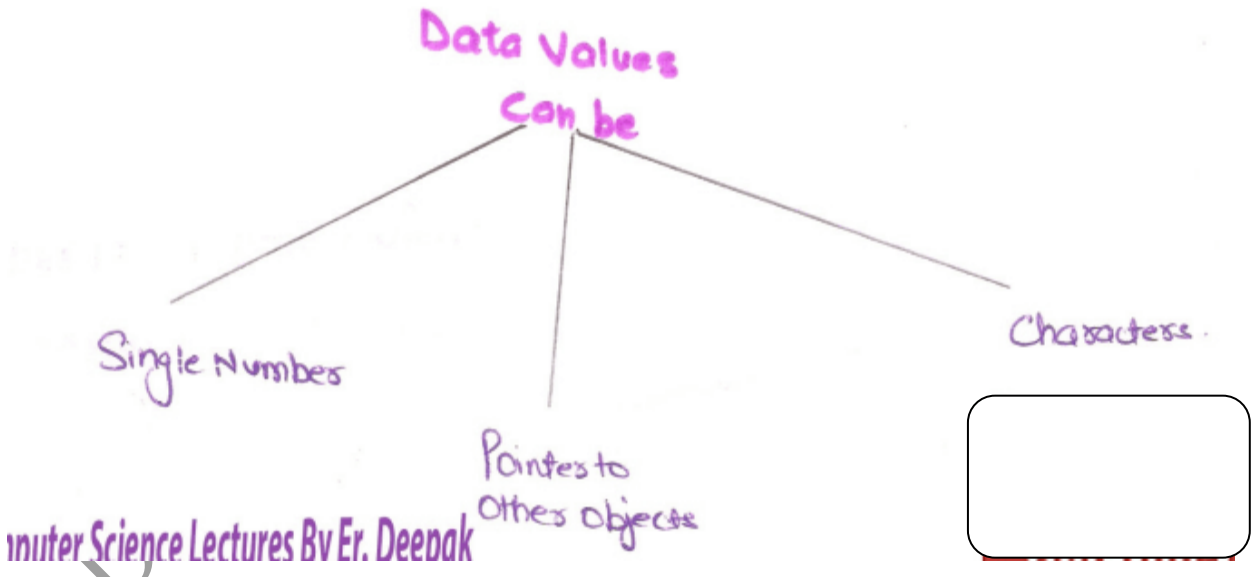
• 2) not directly accessible to programmer- system defined- run time storage, stacks, file buffers, free space lists.

DIWAKAR EDUCATION HUB



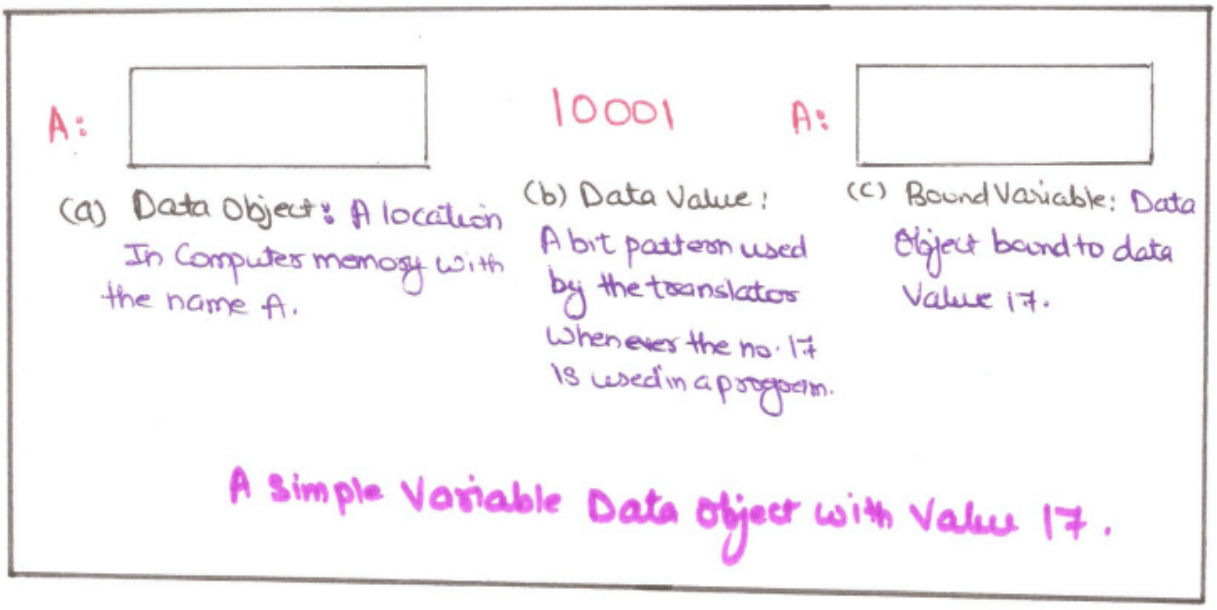
Data values can be:

- Single number
- Pointer to other objects and characters.



nuter Science Lectures By Er. Deepak

Data object is usually represented as storage in Computer memory and a data value is represented by a pattern of bits. **So we can represent the relation between Data Object and Data value.**



A Data Object is elementary if it contains a data value that is always manipulated as a unit.

A Data Object is an **Data Structure** if it is an aggregate of the data object

Binding and Attributes of Data Object:

Binding is an association of data values and entity.

1) Type: This associates the data object with the set of data values that the object may take.

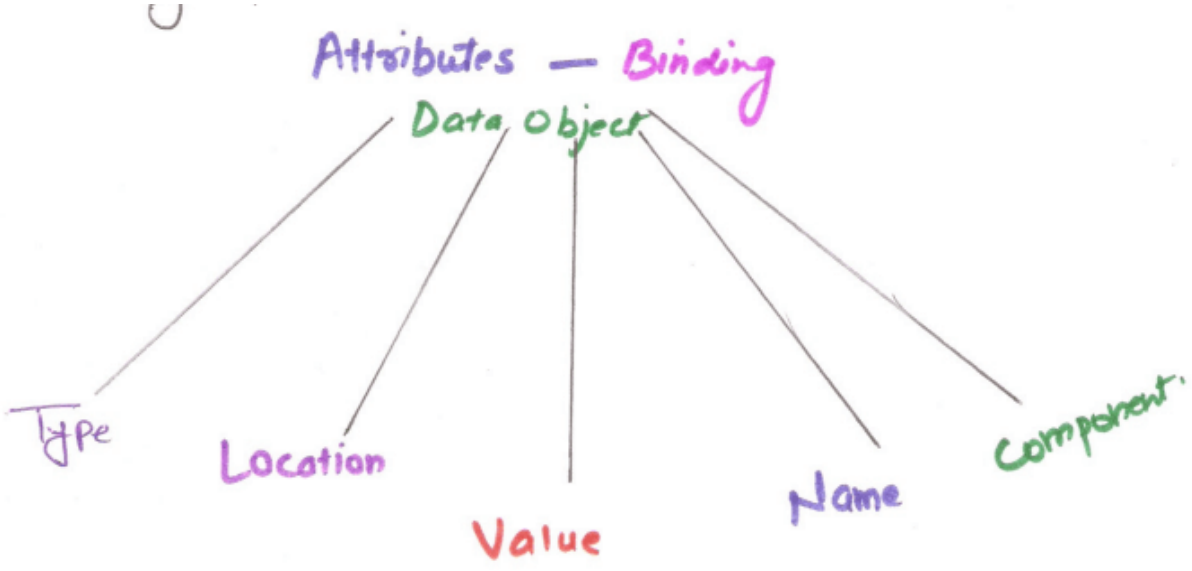
2) location: This associate the binding of a storage location in memory where the data object is represented. Only storage management routines can only change add data object in the virtual computer.

3) Value: This binding is usually the result of an assignment operation.

4) Name: The binding to one or more names by which the object maybe referenced during program execution is usually set up by declaration and modified by subprogram calls the returns.

DIWAKAR EDUCA

5) Component: The binding of a data object to one or more data object of which it is a component is often represented by a pointer value. And may be modified by a change in pointer.



Elementary Data Type: Variables And Constants:

Constant:

A data object with a name that is bound to a value (or values) permanently during its lifetime. The constant value can only be a number, string or identifier which denotes constant.

A constant definition in Pascal introduces an identifier as a synonym for the constant value.

Pascal uses the reserved word const to begin a constant declaration.

const PI=3.1415;

In ALGOL68 we can define constant by

real root2=1.4142135;

That was much acceptable that time.

DIWAKAR EDUC

In Ada, provides a uniform notation for setting constants to initial values and for initializing variables.

X: Constant INTEGER:=17;

In C language: const is used to initialise the constant value

const int MAX=80;

The constant MAX is a programmer defined constant because the programmer explicitly defines the name for the value 30.

In C, there is micro definition which is used for control the execution of program and can be used for declaring constant.

Example #define MAX30

It is a compile-time operation that causes all references to MAX in program to be changed to the constant 30.

In this 30 has two names, the programmer defined MAX and literal name 30. Both of which may be used to refer to a data object containing the value 30.

define MAX 30 is a command, which the translator used to equate "MAX" with the value "30", where as the const attribute in C is a translator directive starting that MAX will always contain the value 30.

Specification of Elementary Data Types

An elementary data object contains a single data values and class of such data objects over which various operations are defined is termed as elementary data type.

Some elementary data types: Integer, real, character, Boolean, enumeration and pointer and specification may differ significantly between two languages.

Attributes: Basic attributes of any data object, such as data type and name are usually invariant during its lifetime.

DIWAKAR EDUCATION HUB

Some attributes may be stored in a descriptor as a part of the data object daily program execution. Others may be used only to determine the storage representation of the data object.

The value of an attribute of a data object is different from the value that the data object contains.

Values: The type of a data object determines the set of possible values that it may contain.

For Example: C defines the following four classes of integer types

int, short, long and char

because most hardware implements multiple Precision integer arithmetic(example 16 bit and 32 bit integers or 32 bit and 64 integers) We can use' short' for shortest value of the integer word length.

long uses the longest value implemented by the hardware.

int uses the most efficient value that the hardware implements.

In C, Characters are stored as 8 bit integers in the type char, which is subtype of integer.

Operations:- The set of operations Defined by language is basically refers that how data object of that data type may be manipulated.

If the operations are primitive operation, means specified as part of language.

Programmer defined operations, in the form of subprograms or method declarations as part of class definitions.

Example

Integer* integer-> **integer**

a) integer addition is an operation that take to integer data objects as an arguments and produces an integer data object as a result.

DIWAKAR EDUCATION HUB

b) SQRT: real-> real

A Square-root operation, SQRT,on real number data object is specified.

(Port of operation)

An algorithm that specifies how to compute the results for any given set of arguments is a common method for specifying the action of an operation.

In C, we have concept a function prototype which signature of an operation, the number, order and data types of the arguments in the domain of an operation are given as well as the order and the data type of the resulting range.

Binary operation: Two arguments with single result

Monolic operation: Single argument with single result.

Implementation of Elementary Data Types:"

Implementation of Elementary data type consists of

- Storage representation for data objects
- Values of that type
- Set of algorithms or procedures that define the operations of the type in terms of manipulations of the storage representation.

Storage representation of Elementary data type:

1) Hardware Influence: Computer hardware influence the storage of elementary data type.

In this case computer hardware executes the program. If the hardware storage representation are used, then the basic operations on data of that data type not implemented using hardware provided operations.

2) Software influenced: If we do not use hardware storage representation, then the operation must be software simulated and some operation will execute much less efficient.

DIWAKAR EDUCATION HUB

Two methods to treat Attributes:

It has to be determined by the compiler and not stored in discriptors during execution or not stored in runtime storage representation. It is usually a method in C language.

It is stored in a descriptor as part of the data object at runtime in LISP, Prolog language .

The storage representation is usually described in terms of

- Size of the block of memory required(the number of memory words bytes, ot bits needed)
- Layout of attributes and data values within the block.

Implementation of operations:

Each operation defined for data objects of a given type may be implemented in one of three main ways:-

1) Directly as a hardware operation: If simple data types are stored using the hardware representation, when the primitive operations are implemented using the arithmetic operations built in to hardware.

2) As a Subprogram or procedure: A square root for an example, this operation is not provided directly as a hardware operation. So it is software simulated implemented as a procedure or function.

3) as an inline code sequence: It is software implementation of the code and its operation. Instead of using a subprogram, operation in the subprogram are copied into the program at the point where the subprogram would otherwise have been invoked.

For Example:

The absolute value of function on numbers

DIWAKAR EDUCATION HUB

abs(x)= if x<0 then -x else x

is usually implemented as an inline code sequence.

a) fetch value of x from memory

b) if x >0, skip the next instruction

c) set x=-x

d) store new value of x in memory

Here each line is implemented by a single hardware operation.

Scalar Data Types

Scalar data types represent

a single object, i.e. only one value can be derived.

In general, scalar objects follow the hardware architecture of a computer.

Numeric Data Types

Integers

Sub -ranges

Floating-point real numbers

Fixed-point real numbers

Other Data Types -

Complex numbers

Rational numbers

Enumerations Booleans

Booleans

DIWAKAR EDUCATION HUB

Characters

Integers Specification

Maximal and minimal values

Operations:

Arithmetic Relational Assignment Bit operations

Implementation - hardware defined

Sub-ranges Specification:

subtype of integer

a sequence of integer values within some restricted range Example:

Pascal declaration **A: 1..10** means that the variable A may be assigned integer values from 1 through 10.

Implementation

smaller storage requirements, better type checking

Floating-point real numbers Specification:

Minimum and maximal value

Round-off issues - the check for equality may fail due to round -off

Implementation

Mantissa - exponent model. Example: 10.5 = 0.105 x 102,

Mantissa: 105, Exponent: 2

Fixed-point real numbers Specification:

real numbers with predefined decimal places

DIWAKAR EDUCATION HUB

Implementation :

Directly supported by hardwareor simulatedby software

Other scalar data types

Complex numbers: software simulated with two storage locations one the real portion and one for the imaginary portion.

Rational numbers: the quotient of two integers. **Enumerations:** Ordered list of different values **Booleans**

Characters

Scalar Data Type Enumeration

An Enumerated data type is a data type whose domain values are given in a list or ordered list and who's only operations are equality and assignment.
or
An Enumeration is an ordered list of distinct values.
or
An Enumeration is a complete ordered listing of all items in a collection.

Pascal was first language which introduced enumeration. To make enumeration facility useful, a programming language must provide a mechanism for declaring and defining the new data type and for declaring variables whose value will come from the element of type.

It is assumed that this literals are distinct and does equality can be directly defined.

"Before an era of enumeration what we had ?"

For example: A variable student class might have only 4 possible values representing fresher, sophomore, junior and senior. Similarly, a variable StudentSex might have only two values representing Male and Female.
Before the contact of enumeration the language like

DIWAKAR EDUCATION HUB

Fortran or Cobol such variables is declared as integer type and distinct values are assigned. like

fresher=1 , sophomore=2, and so on

and male=0, female =1

Then translator manipulate values as integers.

That creates big problem like

Sophomore =1 and female=1

As both have some values can we apply integer based operation on it. As a point of view of programmer it should not be but according to translator it can apply as they are of integer types.

Then languages such as C, article Pascal and Ada includes an Enumeration data type that allows the programmer to define and manipulate such variables directly.

Specification of Enumeration

The programmers defined both the literal name to be used for the values and their ordering using a declaration such as in pascal.

Type months=(jan,feb, mar, apr, june, jul, aug, sep, oct,nov,dec);

In C

enum studentclass{ fresh, soph, junior, senior};

enum studentsex {male, female};

In Pascal, C example can be written as

type class =(fresh, soph, junior, senior};

Followed by declaration for variables such as

Studentclass: Class;

Studentsex class: Class;

Here type definition introduces the type name class, which may be used wherever the primitive type name such as integer might be used time.

It also introduces the **literals of fresh, soph, junior, senior** which may be used

DIWAKAR EDUCATION HUB

wherever a language- defined literal such as "27" might be used. Thus we can write.

if studentclass= junior then.....

Instead of the less understandable

if studentclass= 3 then .....

Which would be required if integer variables were used. Static compiler can find error such as

if student class= Male then

As Male is part of student class. Operations which we can perform-

- Relational operations(equal, less-than, greater-than,etc)
- Assignment
- Successor and Predecessor

Implementation of Enumeration

- Each value in the enumeration sequence is represented at run-time by one of the integers 0,1,2,..... as only a small set of values is involved and the values are never negative.
- In this integer representation is often shortened to omit the sign bit and use only enough bits for the range of values required, as with the sub-range values.
- Only and maximum 2 bits are required to represent the senior=3 in memory because 3=11(binary)/ 2 bits only

In C, the programmer main override default and set any values desired for enumeration values for example.

Enum class{ fresh=74, soph=89, junio=7, senior=28}

With this storage representation for enumeration types. Relational operations such as **=,>, and <** may be implemented.

Scalar Data Types: Booleans

The Boolean data type is a data type, having two values(usually denoted true or false), intended to represent the truth values of logic and Boolean algebra.

DIWAKAR EDUCATION HUB

Specification: In Pascal and Ada, the Boolean data type is considered simply a language - defined enumeration, viz;

type Boolean=(false, true);

Which both defines the names true and false for the values of the types and define ordering false<true

Common Operations in boolean are

and : Boolean*Boolean->Boolean(conjunction)

or : Boolean*Boolean->Boolean(inclusive disjunction)

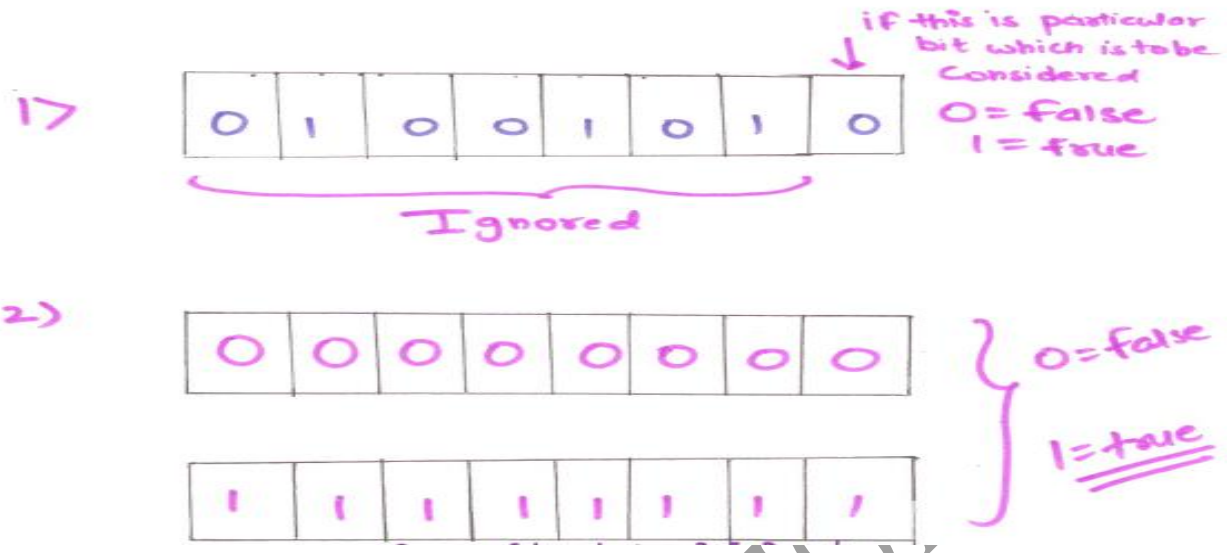
not : Boolean ->Boolean(negative or complement)

Implementation of boolean data type:

Single bit of storage is provided, no descriptor designated the data type is needed. Because single bit may not be separately addressable in memory which often takes a byte or word to represent it if extended. Then the value true and false might be represented in two ways within the storage unit:

- a particular bit is used for the value(often the sign bit of the number representation), with **'0= false', '1=true'**, and the rest of the byte or word ignored, or
- A zero value in the entire storage unit represent False, and any other non zero value represents **true**

DIWAKAR EDUCATION HUB



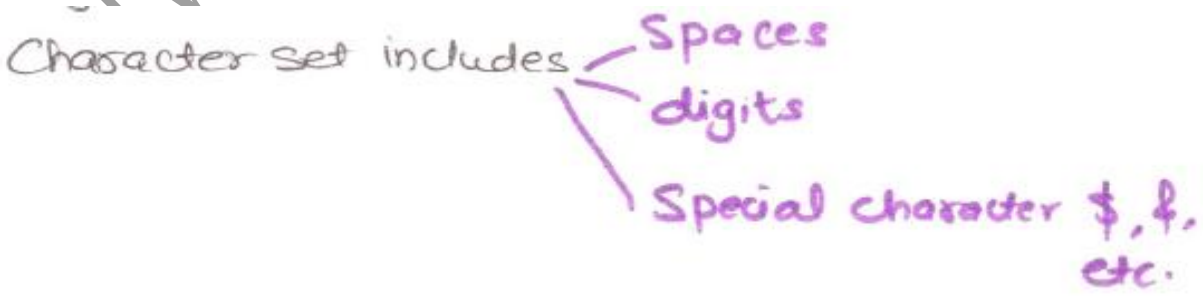
Scalar Data Types: Characters

Specification of Characters

A character data type provides data objects that have a single character as their value. **Set of values in character data type depends upon "Hardware and Operating System"** like ASCII character set. and ordering of the characters in this character set is called **Collecting Sequence** And ordering given by the **'Relational Operations'.**

Character set includes

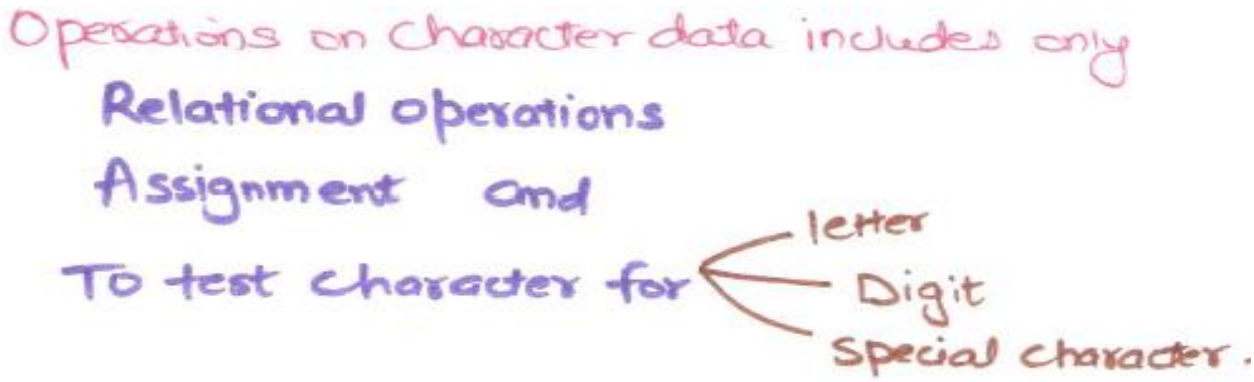
- Spaces
- Digits
- Special character @,#,$,& etc.



Operations on character data include only

WAKAR EDUCATION

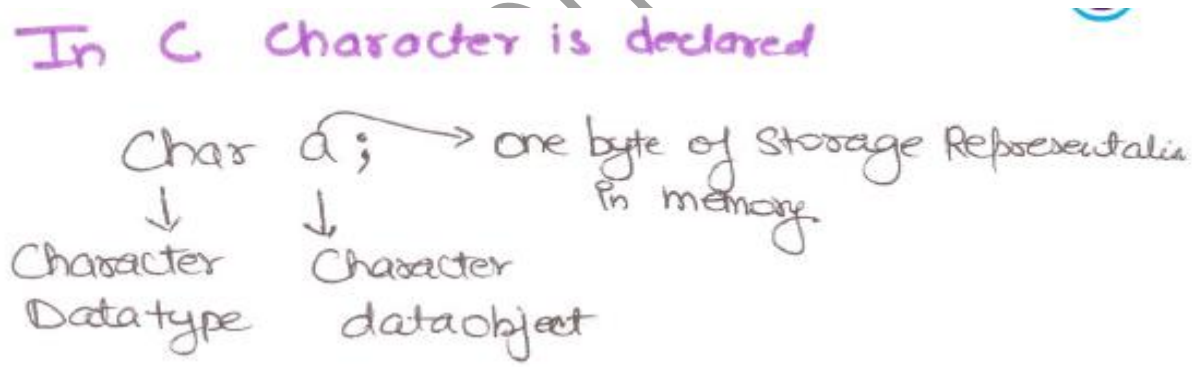
- Relational operations
- Assignment and
- To test character for- Letter, Digit, Special Character.



Implementation of Character Data Type

Character data values are almost directly supported by the underlying hardware and operating system because their use in input-output.

In C Character is declared

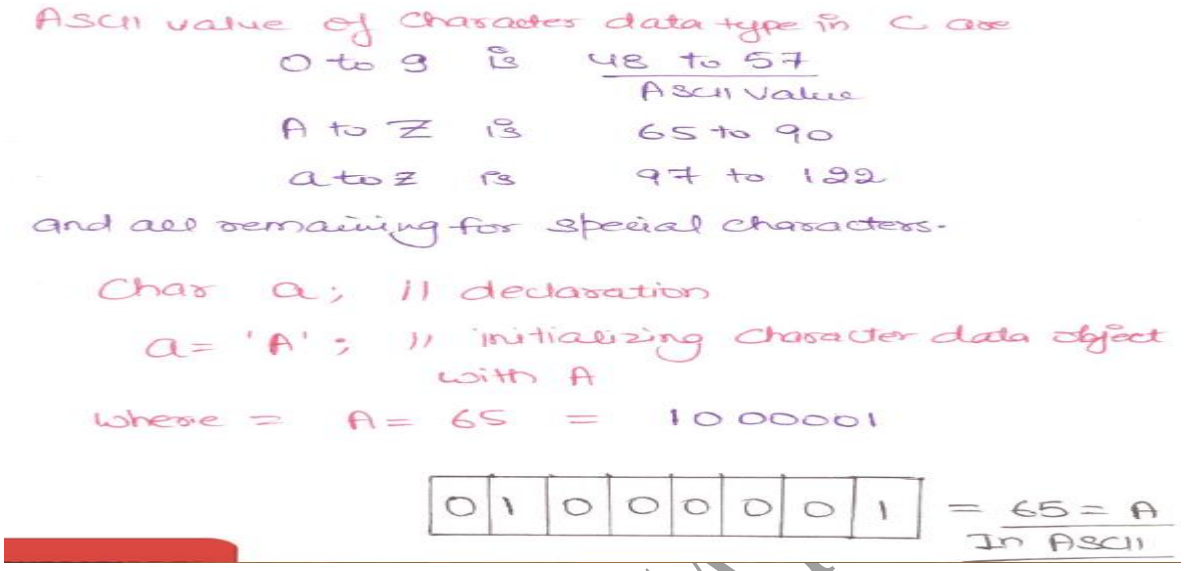


ASCII value of character data type in C are

0 to 9 is 48 to 57/ ASCII value
A to z is 65 to 90
a to z is 97 to 122
And all remaining for special characters.

char a ; //declaration

a='A'; // initializing character data object with A
Where= A=65=1000001



Composite Data Types

Characterized by a complex data structure organization, processed by the compiler.

- Character Strings
- Pointers and Programmer- Constructed Objects
- Files

Character Strings Specification:

Fixed declared length : storage allocation at translation time. Strings longer than the declared length are truncated.

Variable length to a declared bound: storage allocation at translation time. An upper bound for length is set and any string over that length is truncated

Unbounded length: storage allocation at run time. String can be any length

Character Strings - operations

Concatenation – appending two strings

DIWAKAR EDUCA

- **Relational operations – equal, less than, greater than**
    - Substring selection using positioning subscripts
    - Substring selection using pattern matching
    - Input / output formatting
    - Dynamic strings - the string is evaluated at run time.

implementation

- **Fixed declared length**: A packed vector of characters
- **Variable length to a declared bound**: a descriptor that contains the maximum length and the current length
- **Unbounded length:** Either a linked storage of fixed-length data objects or a contiguous array of characters with dynamic run-time storage allocation

Pointers and Programmer- Constructed Objects

Objects

Specification:

Reference data objects only of **a single type** – C, Pascal, Ada.

Reference data objects of **any type** – Smalltalk

C, C++: pointers are data objects and can be manipulated by the program

Java: pointers are hidden data structures, managed by the language implementation

Pointers - implementation

Absolute addresses stored in the pointer. Allows for storing the new object anywhere in the memory

Relative addresses: offset with respect to some base address.

DIWAKAR EDUCATION HUB

Advantages: the entire block can be moved to another location without invalidating the addresses in the pointers, since they are relative, not absolute.

Pointers – implementation problems

- **Management of a general heap storage area**: to create objects of different size
- **Garbage** - the contents of pointer is destroyed, and the object still exists
- **Dangling references**: the object is destroyed however the pointer still contains the address of the used location, and can be wrongly used by the program.

Characteristics:

- Usually reside on secondary storage devices as disks, tapes.
- Lifetime is greater than the lifetime of the program that has created the files.

Implementation – as part of the operating system

Types of Files

Sequential file: a data structure composed of a linear sequence of components of the same type.

Interactive Input-Output: sequential files used in interactive mode.

Direct Access Files: Any single component can be accessed at random just as in an array.

Key: the subscript to access a component.

Implementation: a key table is kept in main memory

Indexed Sequential Files: Similar to direct access files using a key combined with being able to sequentially process the file. The file must be ordered by the key

DIWAKAR EDUCATION HUB

C programming

C programming is a general-purpose, procedural, imperative computer programming language developed in 1972 by Dennis M. Ritchie at the Bell Telephone Laboratories to develop the UNIX operating system. C is the most widely used computer language. It keeps fluctuating at number one scale of popularity along with Java programming language, which is also equally popular and most widely used among modern software programmers.

Why to Learn C Programming?

C programming language is a MUST for students and working professionals to become a great Software Engineer specially when they are working in Software Development Domain. I will list down some of the key advantages of learning C Programming:

1. Easy to learn
2. Structured language
3. It produces efficient programs
4. It can handle low-level activities
5. It can be compiled on a variety of computer platforms

Facts about C

1. C was invented to write an operating system called UNIX.
2. C is a successor of B language which was introduced around the early 1970s.
3. The language was formalized in 1988 by the American National Standard Institute (ANSI).
4. The UNIX OS was totally written in C.
5. Today C is the most widely used and popular System Programming Language.
6. Most of the state-of-the-art software have been implemented using C.

DIWAKAR EDUCATION HUB

7. Today's most popular Linux OS and RDBMS MySQL have been written in C.

Hello World using C Programming.

```
#include <stdio.h

int main() {

  /* my first program in C */

  printf("Hello, World! \n");

  return 0;

}
```

Applications of C Programming

C was initially used for system development work, particularly the programs that make-up the operating system. C was adopted as a system development language because it produces code that runs nearly as fast as the code written in assembly language. Some examples of the use of C are -

1. Operating Systems
2. Language Compilers
3. Assemblers
4. Text Editors
5. Print Spoolers
6. Network Drivers
7. Modern Programs
8. Databases
9. Language Interpreters
10. Utilities

Tokens

DIWAKAR EDUCATION HUB

Tokens are the smallest elements of a program, which are meaningful to the compiler.

The following are the types of tokens: Keywords, Identifiers, Constant, Strings, Operators, etc.

begin with Keywords.

Keywords

Keywords are predefined, reserved words in C and each of which is associated with specific features. These words help us to use the functionality of C language. They have special meaning to the compilers.

There are total 32 keywords in C.

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
continue	for	signed	void
do	if	static	while
default	goto	sizeof	volatile
const	float	short	unsigned

Identifiers

DIWAKAR EDUCATION HUB

Each program element in C programming is known as an identifier. They are used for naming of variables, functions, array etc. These are user-defined names which consist of alphabets, number, underscore '_'. Identifier's name should not be same or same as keywords. Keywords are not used as identifiers.

Rules for naming C identifiers –

- It must begin with alphabets or underscore.
- Only alphabets, numbers, underscore can be used, no other special characters, punctuations are allowed.
- It must not contain white-space.
- It should not be a keyword.
- It should be up to 31 characters long.

Rules for constructing C identifiers

- The first character of an identifier should be either an alphabet or an underscore, and then it can be followed by any of the character, digit, or underscore.
- It should not begin with any numerical digit.
- In identifiers, both uppercase and lowercase letters are distinct. Therefore, we can say that identifiers are case sensitive.
- Commas or blank spaces cannot be specified within an identifier.
- Keywords cannot be represented as an identifier.
- The length of the identifiers should not be more than 31 characters.
- Identifiers should be written in such a way that it is meaningful, short, and easy to read.

Example of valid identifiers

- total, sum, average, _m _, sum_1, etc.

Example of invalid identifiers

DIWAKAR EDUCATION HUB

- 2sum (starts with a numerical digit)
- int (reserved word)
- char (reserved word)
- m+n (special character, i.e., '+')

Types of identifiers

- Internal identifier
- External identifier

Internal Identifier

If the identifier is not used in the external linkage, then it is known as an internal identifier. The internal identifiers can be local variables.

External Identifier

If the identifier is used in the external linkage, then it is known as an external identifier. The external identifiers can be function names, global variables.

Differences between Keyword and Identifier

Keyword	Identifier
Keyword is a pre-defined word.	The identifier is a user-defined word
It must be written in a lowercase letter.	It can be written in both lowercase and uppercase letters.

KARE EDUCATION HUB

Its meaning is pre-defined in the c compiler.	Its meaning is not defined in the c compiler.
It is a combination of alphabetical characters.	It is a combination of alphanumeric characters.
It does not contain the underscore character.	It can contain the underscore character.

an example.

```
int main()
{
   int a=10;
   int A=20;
   printf("Value of a is : %d",a);
   printf("\nValue of A is :%d",A);
   return 0;
}
```

Output

```
Value of a is : 10
Value of A is :20
```

The above output shows that the values of both the variables, 'a' and 'A' are different. Therefore, we conclude that the identifiers are case sensitive.

Strings

DIWAKAR EDUCATION

A string is an array of characters ended with a null character(\0). This null character indicates that string has ended. Strings are always enclosed with double quotes(“ “).

how to declare String in C language –

1) char string[20] = {‘s’,’t’,’u’,’d’,’y’, ‘\0’};
2) char string[20] = “demo”;
3) char string [] = “demo”;

Here is an example of tokens in C language,

Example

```
#include >stdio.h>

int main() {

  // using keyword char

  char a1 = 'H';

  int b = 8;

  float d = 5.6;

  // declaration of string

  char string[200] = "demodotcom";

  if(b<10)

  printf("Character Value : %c\n",a1);

  else

  printf("Float value : %f\n",d);

  printf("String Value : %s\n", string);
```

DIWAKAR EDUCATION HUB

```
  return 0;

}
```

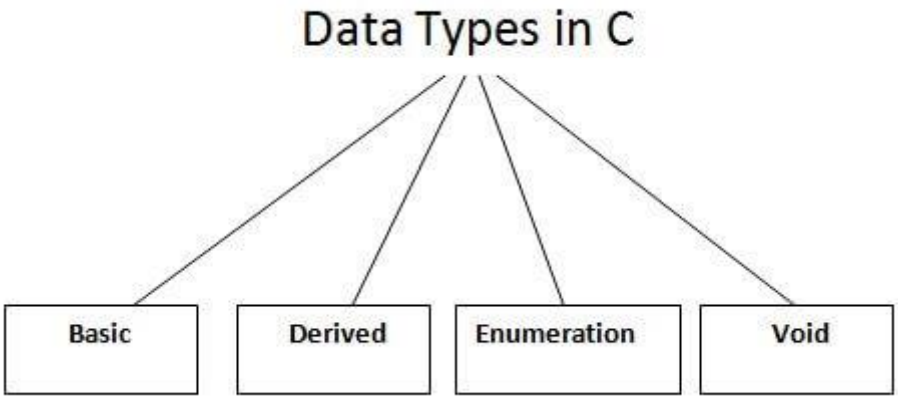
Output

```
Character Value : H

String Value : demodotcom
```

Data Types in C

A data type specifies the type of data that a variable can store such as integer, floating, character, etc.



There are the following data types in C language.

Types	Data Types
Basic Data Type	int, char, float, double
Derived Data Type	array, pointer, structure, union

Enumeration Data Type	enum
Void Data Type	void

Basic Data Types

The basic data types are integer-based and floating-point based. C language supports both signed and unsigned literals.

The memory size of the basic data types may change according to 32 or 64-bit operating system.

the basic data types. Its size is given **according to 32-bit architecture**.

Data Types	Memory Size	Range
char	1 byte	−128 to 127
signed char	1 byte	−128 to 127
unsigned char	1 byte	0 to 255
short	2 byte	−32,768 to 32,767
signed short	2 byte	−32,768 to 32,767
unsigned short	2 byte	0 to 65,535

CATION HUB

int	2 byte	−32,768 to 32,767
signed int	2 byte	−32,768 to 32,767
unsigned int	2 byte	0 to 65,535
short int	2 byte	−32,768 to 32,767
signed short int	2 byte	−32,768 to 32,767
unsigned short int	2 byte	0 to 65,535
long int	4 byte	-2,147,483,648 to 2,147,483,647
signed long int	4 byte	-2,147,483,648 to 2,147,483,647
unsigned long int	4 byte	0 to 4,294,967,295
float	4 byte	
double	8 byte	
long double	10 byte	

Subprogram

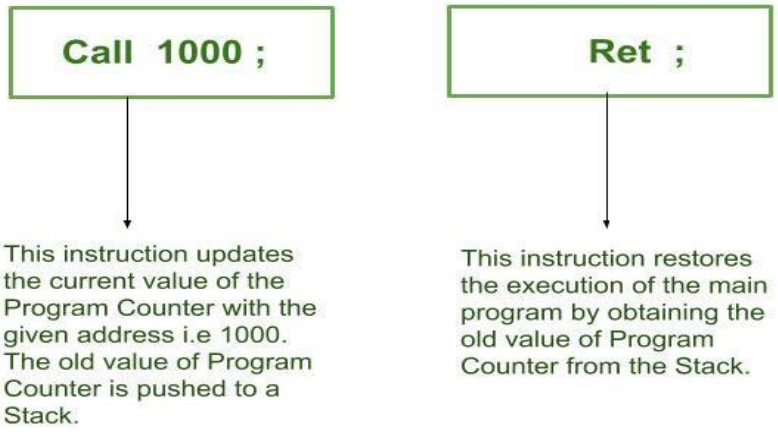
A **Subprogram** is a program inside any larger program that can be reused any number of times.

Control at the subprogram level is concerned with subprogram invocation and the relationship between the calling module and the called module. What follows is a list of possible ways the caller and the callee are related.

Characteristics of a Subprogram:

(1) A Subprogram is implemented using the **Call & Return** instructions in Assembly Language.

(2) The Call Instruction is present in the Main Program and the Return(Ret) Instruction is present in the subprogram itself.

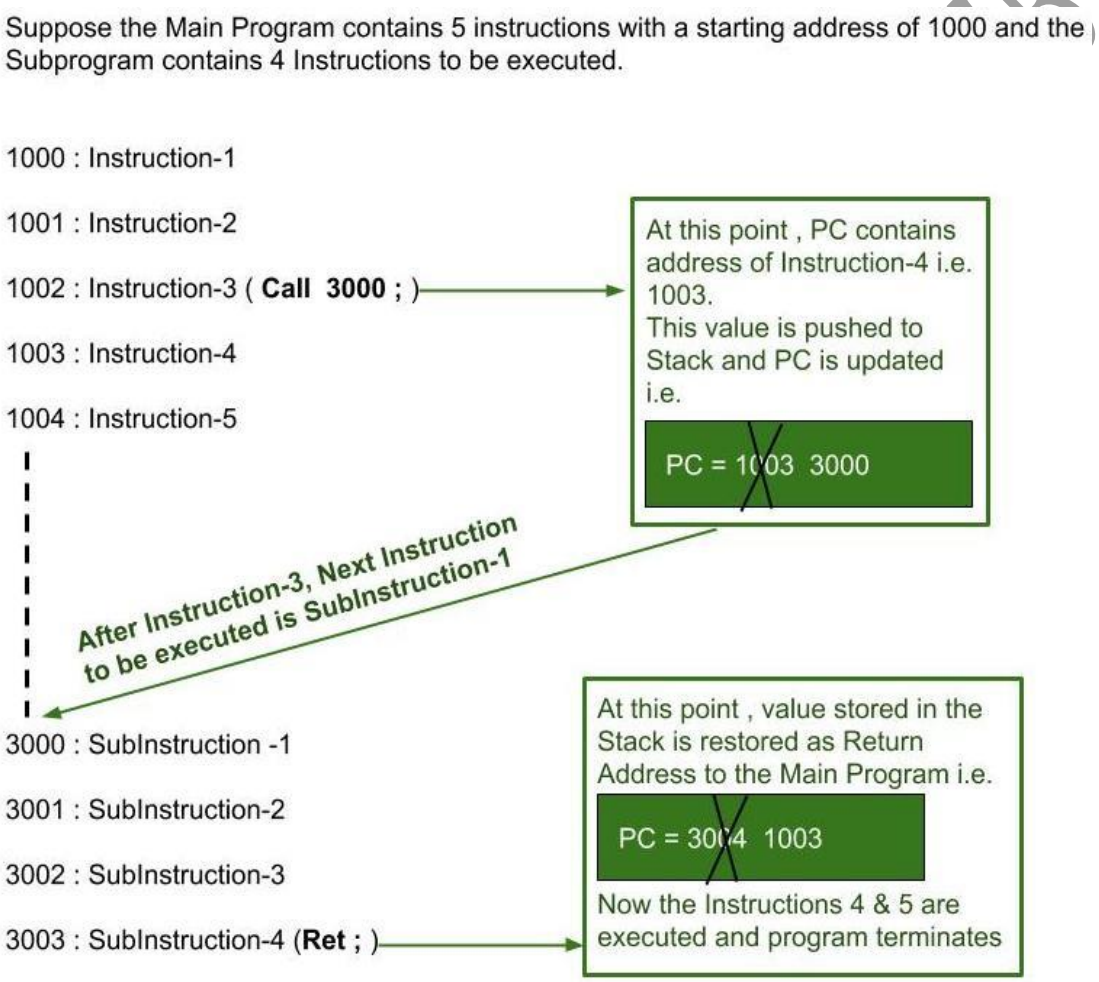


(3) It is important to note that the Main Program is suspended during the execution of any subprogram. Moreover, after the completion of the subprogram the main program executes from the next sequential address present in the Program Counter .

(4) For the implementation of any subprogram, a **"Stack"** is used to store the **"Return Address"** to the Main Program . Here, Return Address means the immediately next instruction address after the Call Instruction in the Main program. This Return Address is present inside the Program Counter . Thus during the execution of the Call Instruction, the Program Counter value is first pushed to the Stack as the Return Address and then the Program Counter value is updated

to the given address in the Call Instruction . Similarly, during the execution of Return(Ret) Instruction, the value present in the stack is popped and the Program Counter value is restored for further execution of the Main Program .

(5) The Main advantage of Subprogram is that it avoids repetition of Code and allows us to reuse the same code again and again.



ARRAY

Introduction

- Arrays are also known as subscript variable.

► Array is a collection of similar elements

► Whatever may the size of array, it always consumed memory in a contiguous manner

Need of array

Till now we have been designing solution to small problems that require less number of variables to handle program data. Think about a scenario where We need to handle hundreds of variables or even more than that.

In such scenario, We might be thinking about what variable names should be used, how to reduce redundant code, etc.

Assume We have to store marks of 100 students and then think about the following:

- What could be our variable naming convention?
- How We efficiently write input instruction to store 100 data.
- How could We easily manipulate data like adding all of them in a less complex style?

The answer to all these questions is subscript notation also known as **arrays**.

Array Declaration

When We want to create large number of variables We need not to think about 100s of names. Assume that We want to create 100 variables to store marks of 100 students.

Here is the way:

```
int marks[100];
```

➢ Notice square brackets after array name marks. This pair of square brackets is used to depict array usage.

DIWAKAR EDUCATION HUB

- Number mention in the square bracket is size of array. In our example we wrote 100, so size of the array is 100. Thus we have declared 100 variables in one go.
- These 100 variables are all of type int. Since array is a collection of similar elements, data type for all 100 variables is int.
- Name of the first variable is marks[0], second variable is marks[1], and so on. Therefore the last variable is marks[99]. Here it is important to note that in C language array indexing starts with 0 and not from 1.

Example: Program to calculate average of 10 marks

```
#include<stdio.h>
#include<conio.h>
main()
{
        int i, marks[10], sum=0;
        float avg;
        clrscr();
        printf("Enter 10 numbers");
        for(i=0;i<=9;i++)
                scanf("%d",&marks[i]);
        for(i=0;i<=9;i++)
                sum=sum+marks[i];
        avg=sum/10.0;
        printf("Average is %f",avg);
        getch();
```

DIWAKAR EDUCATION HUB

}

Explanation:

i. Total numbers of variables in this program are 13. Their names are i, sum, avg, marks[0], marks[1],…marks[9].
ii. Notice the input statement, scanf() is repeated 10 times, this is possible as we can access array index with the help of variable. Observe **&marks[i]** in scanf(), here i is used for indexing. As the loop proceed value of i changes from 0 to 9. In this way wee need not to write scanf() 10 times.
iii. Data manipulation becomes also easy, we added all 10 values stored in an array putting statement **sum=sum+marks[i]** in the loop.
iv. Since we have an array of size 10 and each of these blocks are of type int, to tal memory consumed for this array is 20 bytes (2 bytes for each).

One dimension array

To use an array variable in a program, it must be declared. When defining an array in a program, three things need to be specified.

- What kind of data it can hold, i.e., int, char, double, string, etc.
- How many values it can hold, i.e., the maximum number it can hold
- A name

In the previous example the array declared known as one dimensional array.

Note: According to our previous example marks is not a variable name but marks[0] is a variable. Thus we can not assign anything to marks like marks=5;

Two dimension array

DIWAKAR EDUCATION HUB

C language supports multidimensional arrays also. The simplest form of a multidimensional array is the two-dimensional array. Both the row's and column's index begins from 0.

Two dimensional arrays is actually array of arrays. So here we are creating an array of several identical arrays.

Consider the following declaration style

int a[2][3];

this declaration means, we have an array of 2 arrays containing 3 int blocks each.

- Total numbers of blocks are 6, they are all of type int. Memory allocation done is always sequential but we can assume it as two arrays each of size 3.
- Logically we can see it as a row column structure. First row is our 0th array and second row is 1st array.
- Two dimensional arrays are used to handle data which is logically two dimensional like matrix.

Example: **Program to add two matrix of order 3 x 3**.

```
#include<conio.h>
#include<stdio.h>
main()
{
int a[3][3],b[3][3],c[3][3];
        int i,j;
        clrscr();
        printf("Enter 9 numbers for first matrix\n");
```

DIWAKAR EDUCATION HUB

```
        for(i=0;i<3;i++)
                for(j=0;j<3;j++)
                        scanf("%d",&a[i][j]);
printf("Enter 9 numbers for second matrix\n");
        for(i=0;i<3;i++)
                for(j=0;j<3;j++)
                        scanf("%d",&b[i][j]);
for(i=0;i<3;i++)
                for(j=0;j<3;j++)
                        c[i][j]=a[i][j]+b[i][j];
printf("\nSum of matrix is: \n");
        for(i=0;i<3;i++)
        {
                for(j=0;j<3;j++)
                        printf("%d ",c[i][j]);
                printf("\n");
        }
        getch();
}
```

Explanation:

DIWAKAR EDUCATION HUB

1) Notice the declaration of arrays. We have declared three two dimensional arrays.

2) Observe the style of input. scanf() is repeated with nested loop.

3) Lastly sum of corresponding elements of two matrices are added and stored in third array.

4) Finally sum is displayed on the screen

Initialization of array at the time of declaration

Initializing one dimension array

int a[5]={ 23,45,11,67,55};

int a[ ]= { 23,45,11,67,55};

int a[5]= {22,45};

int a[5]= {12,23,76,85,43,33};

In the first style we declared an array with size 5 and assign 5 values to them, First value is stored in a[0] and last in a[4].

Second style is also valid. When We initialize array at the time of declaration it is not necessary to mention size of array, otherwise it is compulsory. Compiler assumes the size of array by counting number of values assigned to it.

Third notation style is also valid as two variables a[0] and a[1] initializes with 22 and 45, remaining variables are initialized to 0.

Fourth style leads to compilation error. an array can not be intialized with data more than the size of array.

Initialization of two dimension array

int b[2] [3] = {12,65,78,45,33,21};

int b[ ] [3] = {12,65,78,45,33,21};

DIWAKAR EDUCATION HUB

int b[2] [ ] = {12,65,78,45,33,21};

int b[ ] [ ] = {12,65,78,45,33,21};

Last two styles are invalid and lead to compile time error.

C Structures

Structure is a user-defined datatype in C language which allows us to combine data of different types together. Structure helps to construct a complex data type which is more meaningful. It is somewhat similar to an Array, but an array holds data of similar type only. But structure on the other hand, can store data of any type, which is practical more useful.

For example: If I have to write a program to store Student information, which will have Student's name, age, branch, permanent address, father's name etc, which included string values, integer values etc, how can I use arrays for this problem, I will require something which can hold data of different types together.

In structure, data is stored in form of **records**.

Defining a structure

struct keyword is used to define a structure. struct defines a new data type which is a collection of primary and derived datatypes.

Syntax:

```
struct [structure_tag]
{
   //member variable 1
   //member variable 2
   //member variable 3
   ...
```

DIWAKAR EDUCATION HUB

```
}[structure_variables];
```

As We can see in the syntax above, we start with the struct keyword, then it's optional to provide our structure a name, we suggest to give it a name, then inside the curly braces, we have to mention all the member variables, which are nothing but normal C language variables of different types like int, float, array etc.

After the closing curly brace, we can specify one or more structure variables, again this is optional.

Note: The closing curly brace in the structure type declaration must be followed by a semicolon(;).

Example of Structure

```
struct Student
{
   char name[25];
   int age;
   char branch[10];
   // F for female and M for male
   char gender;
};
```

Here struct Student declares a structure to hold the details of a student which consists of 4 data fields, namely name, age, branch and gender. These fields are called **structure elements or members**.

Each member can have different datatype, like in this case, name is an array of char type and age is of int type etc. **Student** is the name of the structure and is called as the **structure tag**.

DIWAKAR EDUCATION HUB

Declaring Structure Variables

It is possible to declare variables of a **structure**, either along with structure definition or after the structure is defined. **Structure** variable declaration is similar to the declaration of any normal variable of any other datatype. Structure variables can be declared in following two ways:

1) Declaring Structure variables separately

```
struct Student
{
   char name[25];
   int age;
   char branch[10];
   //F for female and M for male
   char gender;
};
struct Student S1, S2;     //declaring variables of struct Student
```

2) Declaring Structure variables with structure definition

```
struct Student
{
   char name[25];
   int age;
   char branch[10];
   //F for female and M for male
```

DIWAKAR EDUCATION HUB

```
   char gender;
}S1, S2;
```

Here S1 and S2 are variables of structure Student. However this approach is not much recommended.

Accessing Structure Members

Structure members can be accessed and assigned values in a number of ways. Structure members have no meaning individually without the structure. In order to assign a value to any structure member, the member name must be linked with the **structure** variable using a dot . operator also called **period** or **member access** operator.

For example:

```
#include<stdio.h>
#include<string.h>
struct Student
{
   char name[25];
   int age;
   char branch[10];
   //F for female and M for male
   char gender;
};
int main()
{
```

DIWAKAR EDUCATION HUB

```
    struct Student s1;
    /*
       s1 is a variable of Student type and
       age is a member of Student
    */
    s1.age = 18;
    /*
       using string function to add name
    */
    strcpy(s1.name, "Viraaj");
    /*
       displaying the stored values
    */
    printf("Name of Student 1: %s\n", s1.name);
    printf("Age of Student 1: %d\n", s1.age);
    return 0;
}
```

DIWAKAR EDUCATION HUB

Output

Name of Student 1: Viraaj

Age of Student 1: 18

We can also use scanf() to give values to structure members through terminal.

```
scanf(" %s ", s1.name);

scanf(" %d ", &s1.age);
```

Structure Initialization

Like a variable of any other datatype, structure variable can also be initialized at compile time.

```
struct Patient
{
   float height;
   int weight;
   int age;
};


struct Patient p1 = { 180.75 , 73, 23 };   //initialization
```

or,

```
struct Patient p1;
p1.height = 180.75;    //initialization of each member separately
p1.weight = 73;
p1.age = 23;
```

C Array of Structures

Why use an array of structures?

Consider a case, where we need to store the data of 5 students. We can store it by using the structure as given below.

DIWAKAR EDUCATION HUB

```
#include<stdio.h>
struct student
{
   char name[20];
   int id;
   float marks;
};
void main()
{
   struct student s1,s2,s3;
   int dummy;
   printf("Enter the name, id, and marks of student 1 ");
   scanf("%s %d %f",s1.name,&s1.id,&s1.marks);
   scanf("%c",&dummy);
   printf("Enter the name, id, and marks of student 2 ");
   scanf("%s %d %f",s2.name,&s2.id,&s2.marks);
   scanf("%c",&dummy);
   printf("Enter the name, id, and marks of student 3 ");
   scanf("%s %d %f",s3.name,&s3.id,&s3.marks);
   scanf("%c",&dummy);
   printf("Printing the details....\n");
```

DIWAKAR EDUCATION HUB

```
    printf("%s %d %f\n",s1.name,s1.id,s1.marks);
    printf("%s %d %f\n",s2.name,s2.id,s2.marks);
    printf("%s %d %f\n",s3.name,s3.id,s3.marks);
}
```

Output

```
Enter the name, id, and marks of student 1 James 90 90
Enter the name, id, and marks of student 2 Adoms 90 90
Enter the name, id, and marks of student 3 Nick 90 90
Printing the details....
James 90 90.000000
Adoms 90 90.000000
Nick 90 90.000000
```

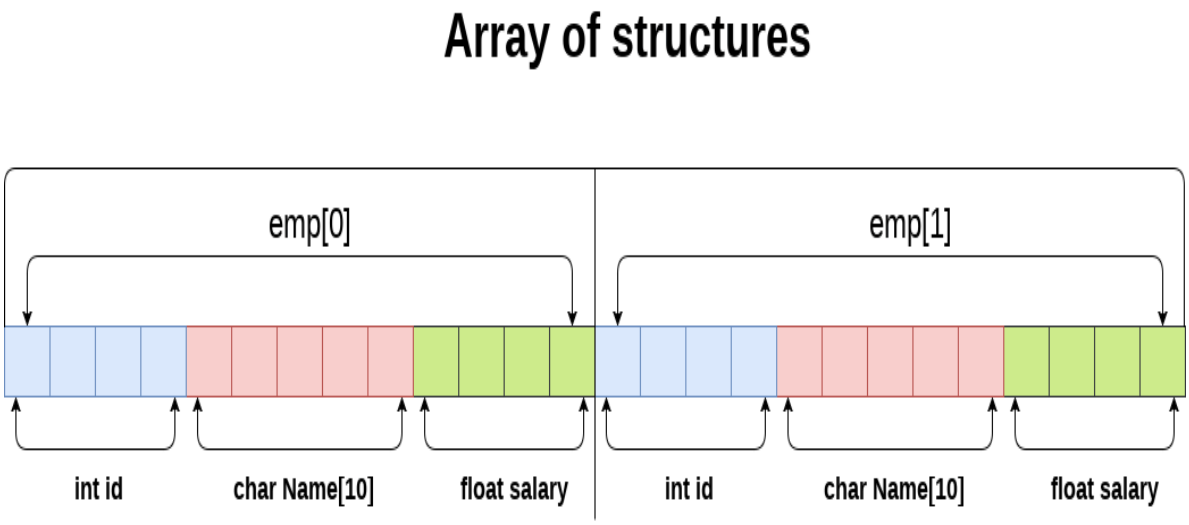
In the above program, we have stored data of 3 students in the structure. However, the complexity of the program will be increased if there are 20 students. In that case, we will have to declare 20 different structure variables and store them one by one. This will always be tough since we will have to declare a variable every time we add a student. Remembering the name of all the variables is also a very tricky task. However, c enables us to declare an array of structures by using which, we can avoid declaring the different structure variables; instead we can make a collection containing all the structures that store the information of different entities.

Array of Structures in C

An array of structres in C can be defined as the collection of multiple structures variables where each variable contains information about different entities. The

DIWAKAR EDUCATION HUB

array of structures in C are used to store information about multiple entities of different data types. The array of structures is also known as the collection of structures.




```
struct employee
{
    int id;
    char name[5];
    float salary;
};
struct employee emp[2];
```

sizeof (emp) = 4 + 5 + 4 = 13 bytes

sizeof (emp[2]) = 26 bytes

an example of an array of structures that stores information of 5 students and prints it.

```
#include<stdio.h>

#include <string.h>

struct student{

int rollno;
```

```
char name[10];
};
int main(){
int i;
struct student st[5];
printf("Enter Records of 5 students");
for(i=0;i<5;i++){
printf("\nEnter Rollno:");
scanf("%d",&st[i].rollno);
printf("\nEnter Name:");
scanf("%s",&st[i].name);
}
printf("\nStudent Information List:");
for(i=0;i<5;i++){
printf("\nRollno:%d, Name:%s",st[i].rollno,st[i].name);
}
  return 0;
}
```

DIWAKAR EDUCATION HUB

Output:

```
Enter Records of 5 students
Enter Rollno:1
```

```
Enter Name:Sonoo

Enter Rollno:2

Enter Name:Ratan

Enter Rollno:3

Enter Name:Vimal

Enter Rollno:4

Enter Name:James

Enter Rollno:5

Enter Name:Sarfraz


Student Information List:

Rollno:1, Name:Sonoo

Rollno:2, Name:Ratan

Rollno:3, Name:Vimal

Rollno:4, Name:James

Rollno:5, Name:Sarfraz
```

DIWAKAR EDUCATION HUB

Nested Structure in C

Nesting of structures, is also permitted in C language. Nested structures means, that one structure has another stucture as member variable.

C provides us the feature of nesting one structure within another structure by using which, complex data types are created. For example, we may need to store the address of an entity employee in a structure. The attribute address may also

have the subparts as street number, city, state, and pin code. Hence, to store the address of the employee, we need to store the address of the employee into a separate structure and nest the structure address into the structure employee. Consider the following program.

```
#include<stdio.h>
struct address
{
   char city[20];
   int pin;
   char phone[14];
};
struct employee
{
   char name[20];
   struct address add;
};
void main ()
{
   struct employee emp;
   printf("Enter employee information?\n");
   scanf("%s %s %d %s",emp.name,emp.add.city, &emp.add.pin, emp.add.pho
ne);
```

DIWAKAR EDUCATION HUB

```
    printf("Printing the employee information....\n");

    printf("name: %s\nCity: %s\nPincode: %d\nPhone: %s",emp.name,emp.add.
city,emp.add.pin,emp.add.phone);

}
```

Output

```
Enter employee information?

Arun

Delhi

110001

1234567890

Printing the employee information....


name: Arun

City: Delhi

Pincode: 110001

Phone: 1234567890
```

The structure can be nested in the following ways.

- **By separate structure**
- **By Embedded structure**

1) Separate structure

Here, we create two structures, but the dependent structure should be used inside the main structure as a member. Consider the following example.

DIWAKAR EDUCATION HUB

```
struct Date
{
  int dd;
  int mm;
  int yyyy;
};
struct Employee
{
  int id;
  char name[20];
  struct Date doj;
}emp1;
```

As We can see, doj (date of joining) is the variable of type Date. Here doj is used as a member in Employee structure. In this way, we can use Date structure in many structures.

2) Embedded structure

The embedded structure enables us to declare the structure inside the structure. Hence, it requires less line of codes but it can not be used in multiple data structures. Consider the following example.

- **struct** Employee
- {
- **int** id;

DIWAKAR EDUCATION HUB

- **char** name[20];
- **struct** Date
- {
- **int** dd;
- **int** mm;
- **int** yyyy;
- }doj;
- }emp1;

Accessing Nested Structure

We can access the member of the nested structure by Outer_Structure.Nested_Structure.member as given below:

```
e1.doj.dd

e1.doj.mm

e1.doj.yyyy
```

C Nested Structure example

a simple example of the nested structure in C language.

```
#include <stdio.h>

#include <string.h>

struct Employee

{

  int id;
```

DIWAKAR EDUCATION HUB

```
  char name[20];
  struct Date
   {
    int dd;
    int mm;
    int yyyy;
   }doj;
}e1;
int main( )
{
  //storing employee information
  e1.id=101;
  strcpy(e1.name, "Sonoo Jaiswal");//copying string into char array
  e1.doj.dd=10;
  e1.doj.mm=11;
  e1.doj.yyyy=2014;

  //printing first employee information
  printf( "employee id : %d\n", e1.id);
  printf( "employee name : %s\n", e1.name);
```

DIWAKAR EDUCATION HUB

```
  printf( "employee date of joining (dd/mm/yyyy) : %d/%d/%d\n", e1.doj.dd,e
1.doj.mm,e1.doj.yyyy);

  return 0;

}
```

Output:

employee id : 101

employee name : Sonoo Jaiswal

employee date of joining (dd/mm/yyyy) : 10/11/2014

Passing structure to function

Just like other variables, a structure can also be passed to a function. We may pass the structure members into the function or pass the structure variable at once. Consider the following example to pass the structure variable employee to a function display() which is used to display the details of an employee.

```
#include<stdio.h>

struct address

{

   char city[20];

   int pin;

   char phone[14];

};

struct employee

{
```

DIWAKAR EDUCATION HUB

```
   char name[20];
   struct address add;
};
void display(struct employee);
void main ()
{
   struct employee emp;
   printf("Enter employee information?\n");
   scanf("%s %s %d %s",emp.name,emp.add.city, &emp.add.pin, emp.add.pho
ne);
   display(emp);
}
void display(struct employee emp)
{
  printf("Printing the details....\n");
  printf("%s %s %d %s",emp.name,emp.add.city,emp.add.pin,emp.add.phone);
}
```

DIWAKAR EDUCATION HUB

Structure as Function Arguments

We can pass a structure as a function argument just like we pass any other variable or an array as a function argument.

Example:

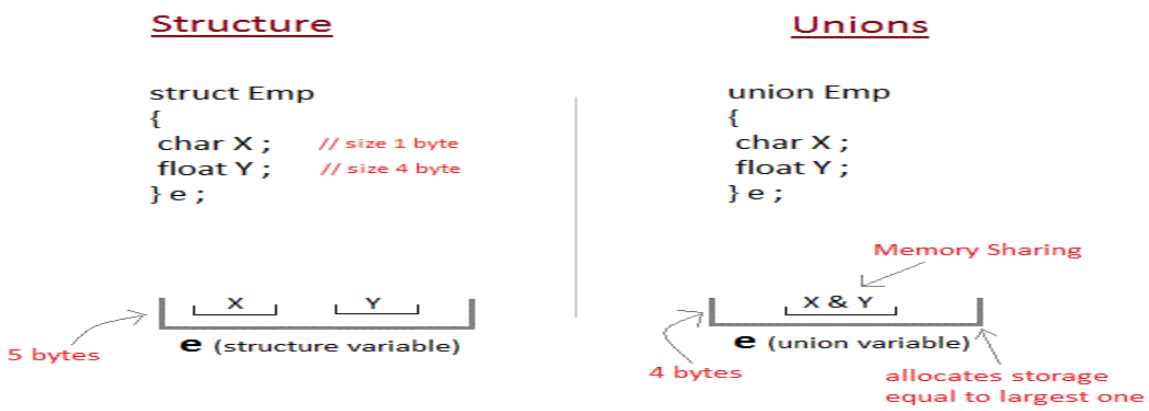
```
#include<stdio.h>
```

```
struct Student
{
   char name[10];
   int roll;
};
void show(struct Student st);
void main()
{
   struct Student std;
   printf("\nEnter Student record:\n");
   printf("\nStudent name:\t");
   scanf("%s", std.name);
   printf("\nEnter Student rollno.:\t");
   scanf("%d", &std.roll);
   show(std);
}
void show(struct Student st)
{
   printf("\nstudent name is %s", st.name);
   printf("\nroll is %d", st.roll);
}
```

DIWAKAR EDUCATION HUB

C Unions

Unions are conceptually similar to **structures**. The syntax to declare/define a union is also similar to that of a structure. The only differences is in terms of storage. In **structure** each member has its own storage location, whereas all members of **union** uses a single shared memory location which is equal to the size of its largest data member.



This implies that although a **union** may contain many members of different types, **it cannot handle all the members at the same time**. A **union** is declared using the union keyword.

```
union item

{

   int m;

   float x;

   char c;

}It1;
```

This declares a variable It1 of type union item. This union contains three members each with a different data type. However only one of them can be used at a time. This is due to the fact that only one location is allocated for all

the union variables, irrespective of their size. The compiler allocates the storage that is large enough to hold the largest variable type in the union.

In the union declared above the member x requires **4 bytes** which is largest amongst the members for a 16-bit machine. Other members of union will share the same memory address.

Accessing a Union Member in C

Syntax for accessing any union member is similar to accessing structure members,

```
union test
{
   int a;
   float b;
   char c;
}t;
t.a;    //to access members of union t
t.b;
t.c;
```

Time for an Example

```
#include <stdio.h>
union item
{
   int a;
   float b;
```

DIWAKAR EDUCATION HUB

```
   char ch;
};
int main( )
{
   union item it;
   it.a = 12;
   it.b = 20.2;
   it.ch = 'z'
   printf("%d\n", it.a);
   printf("%f\n", it.b);
   printf("%c\n", it.ch);
   return 0;
}
```

Output

```
-26426
20.1999
z
```

As we can see here, the values of a and b get corrupted and only variable c prints the expected result. This is because in union, the memory is shared among different data types. Hence, the only member whose value is currently stored will have the memory.

DIWAKAR EDUCATION HUB

In the above example, value of the variable c was stored at last, hence the value of other variables is lost.

Advantage of union over structure

It **occupies less memory** because it occupies the size of the largest member only.

Disadvantage of union over structure

Only the last entered data can be stored in the union. It overwrites the data previously stored in the union.

String

String is nothing but a collection of characters in a linear sequence. 'C' always treats a string a single data even though it contains whitespaces. A single character is defined using single quote representation. A string is represented using double quote marks.

Example, "Welcome to the world of programming!"

'C' provides standard library <string.h> that contains many functions which can be used to perform complicated string operations easily.

Declare and initialize a String

A string is a simple array with char as a data type. 'C' language does not directly support string as a data type. Hence, to display a string in 'C', We need to make use of a character array.

The general syntax for declaring a variable as a string is as follows,

```
char string_variable_name [array_size];
```

The classic string declaration can be done as follow:

```
char string_name[string_length] = "string";
```

DIWAKAR EDUCATION HUB

The size of an array must be defined while declaring a string variable because it used to calculate how many characters are going to be stored inside the string variable. Some valid examples of string declaration are as follows,

```
char first_name[15];    //declaration of a string variable

char last_name[15];
```

The above example represents string variables with an array size of 15. This means that the given character array is capable of holding 15 characters at most. The indexing of array begins from 0 hence it will store characters from a 0-14 position. **The C compiler automatically adds a NULL character '\0' to the character array created.**

The initialization of a string variable. Following example demonstrates the initialization of a string variable,

```
char first_name[15] = "ANTHONY";

char first_name[15] = {'A','N','T','H','O','N','Y','\0'}; // NULL character '\0' is required at end in this declaration

char string1 [6] = "hello";/* string size = 'h'+'e'+'l'+'l'+'o'+"NULL" = 6 */

char string2 [ ] = "world";  /* string size = 'w'+'o'+'r'+'l'+'d'+"NULL" = 6 */

char string3[6] = {'h', 'e', 'l', 'l', 'o', '\0'} ; /*Declaration as set of characters ,Size 6*/
```

In string3, the NULL character must be added explicitly, and the characters are enclosed in single quotation marks.

'C' also allows us to initialize a string variable without defining the size of the character array. It can be done in the following way,

```
char first_name[ ] = "NATHAN";
```

The name of a string acts as a pointer because it is basically an array.

DIWAKAR EDUCATION HUB

String Input: Read a String

When writing interactive programs which ask the user for input, C provides the scanf(), gets(), and fgets() functions to find a line of text entered from the user.

When we use scanf() to read, we use the "%s" format specifier without using the "&" to access the variable address because an array name acts as a pointer. For example:

```
#include <stdio.h>

int main() {

char name[10];

int age;

printf("Enter your first name and age: \n");

scanf("%s %d", name, &age);

printf("You entered: %s %d",name,age);

}
```

Output:

```
Enter your first name and age:

John_Smith 48
```

The problem with the scanf function is that it never reads an entire string. It will halt the reading process as soon as whitespace, form feed, vertical tab, newline or a carriage return occurs. Suppose we give input as "Guru99 Tutorials" then the scanf function will never read an entire string as a whitespace character occurs between the two names. The scanf function will only read Guru99.

DIWAKAR EDUCATION HUB

In order to read a string contains spaces, we use the gets() function. Gets ignores the whitespaces. It stops reading when a newline is reached (the Enter key is pressed).For example:

```
#include <stdio.h>

int main() {

char full_name[25];

printf("Enter your full name: ");

gets(full_name);

printf("My full name is %s ",full_name);

return 0;

}
```

Output:

```
Enter your full name: Dennis Ritchie

My full name is Dennis Ritchie
```

Another safer alternative to gets() is fgets() function which reads a specified number of characters. For example:

```
#include <stdio.h>

int main() {

char name[10];

printf("Enter your  name plz: ");

fgets(name, 10, stdin);

printf("My name is %s ",name);
```

DIWAKAR EDUCATION HUB

```
return 0;}
```

Output:

```
Enter your name plz: Carlos

My name is Carlos
```

The fgets() arguments are :

- the string name,
- the number of characters to read,
- stdin means to read from the standard input which is the keyboard.

String Output: Print/Display a String

The standard printf function is used for printing or displaying a string on an output device. The format specifier used is %s

Example,

```
printf("%s", name);
```

String output is done with the fputs() and printf() functions.

fputs() function

The fputs() needs the name of the string and a pointer to where We want to display the text. We use stdout which refers to the standard output in order to print to the screen.For example:

```
#include <stdio.h>

int main()

{char town[40];
```

DIWAKAR EDUCATION HUB

```
printf("Enter your town: ");
gets(town);
fputs(town, stdout);
return 0;}
```

Output:

```
Enter your town: New York
New York
```

puts function

The puts function prints the string on an output device and moves the cursor back to the first position. A puts function can be used in the following way,

```
#include <stdio.h>
int main() {
char name[15];
gets(name);       //reads a string
puts(name);       //displays a string
return 0;}
```

The syntax of this function is comparatively simple than other functions.

The string library

The standard 'C' library provides various functions to manipulate the strings within a program. These functions are also called as string handlers. All these handlers are present inside <string.h> header file.

DIWAKAR EDUCATION HUB

Function	Purpose
strlen()	This function is used for finding a length of a string. It returns how many characters are present in a string excluding the NULL character.
strcat(str1, str2)	This function is used for combining two strings together to form a single string. It Appends or concatenates str2 to the end of str1 and returns a pointer to str1.
strcmp(str1, str2)	This function is used to compare two strings with each other. It returns 0 if str1 is equal to str2, less than 0 if str1 < str2, and greater than 0 if str1 > str2.

The program below which demonstrates string library functions:

```
#include <stdio.h>
#include <string.h>
int main () {
//string initialization
char string1[15]="Hello";
char string2[15]=" World!";
char string3[15];
```

DIWAKAR EDUCATION HUB

```
int val;

//string comparison

val= strcmp(string1,string2);

if(val==0){

   printf("Strings are equal\n");

}

else{

   printf("Strings are not equal\n");

}

//string concatenation

printf("Concatenated string:%s",strcat(string1,string2)); //string1 contains hello world!

//string length

printf("\nLength of first string:%d",strlen(string1));

printf("\nLength of second string:%d",strlen(string2));

//string copy

printf("\nCopied string is:%s\n",strcpy(string3,string1));  //string1 is copied into string3

return 0;

}
```

Output:

```
Strings are not equal
```

DIWAKAR EDUCATION HUB

```
Concatenated string:Hello World!

Length of first string:12

Length of second string:7

Copied string is:Hello World!
```

Other important library functions are:

- strncmp(str1, str2, n) :it returns 0 if the first n characters of str1 is equal to the first n characters of str2, less than 0 if str1 < str2, and greater than 0 if str1 > str2.
- strncpy(str1, str2, n) This function is used to copy a string from another string. Copies the first n characters of str2 to str1
- strchr(str1, c): it returns a pointer to the first occurrence of char c in str1, or NULL if character not found.
- strrchr(str1, c): it searches str1 in reverse and returns a pointer to the position of char c in str1, or NULL if character not found.
- strstr(str1, str2): it returns a pointer to the first occurrence of str2 in str1, or NULL if str2 not found.
- strncat(str1, str2, n) Appends (concatenates) first n characters of str2 to the end of str1 and returns a pointer to str1.
- strlwr() :to convert string to lower case
- strupr() :to convert string to upper case
- strrev() : to reverse string

Converting a String to a Number

In C programming, we can convert a string of numeric characters to a numeric value to prevent a run-time error. The stdio.h library contains the following functions for converting a string to a number:

DIWAKAR EDUCATION HUB

- int atoi(str) Stands for ASCII to integer; it converts str to the equivalent int value. 0 is returned if the first character is not a number or no numbers are encountered.
- double atof(str) Stands for ASCII to float, it converts str to the equivalent double value. 0.0 is returned if the first character is not a number or no numbers are encountered.
- long int atol(str) Stands for ASCII to long int, Converts str to the equivalent long integer value. 0 is returned if the first character is not a number or no numbers are encountered.

The following program demonstrates atoi() function:

```
#include <stdio.h>

int main()

{char *string_id[10];

 int ID;

 printf("Enter a number: ");

 gets(string_id);

 ID = atoi(string_id);

  printf("you enter %d ",ID);

 return 0;}
```

Output:

```
Enter a number: 221348

you enter 221348
```

- A string pointer declaration such as char *string = "language" is a constant and cannot be modified.

DIWAKAR EDUCATION HUB

Summary

- A string is a sequence of characters stored in a character array.
- A string is a text enclosed in double quotation marks.
- A character such as 'd' is not a string and it is indicated by single quotation marks.
- 'C' provides standard library functions to manipulate strings in a program. String manipulators are stored in <string.h> header file.
- A string must be declared or initialized before using into a program.
- There are different input and output string functions, each one among them has its features.
- Don't forget to include the string library to work with its functions
- We can convert string to number through the atoi(), atof() and atol() which are very useful for coding and decoding processes.
- We can manipulate different strings by defining a string array.

Pointer?

POINTER is a variable that stores address of another variable. A pointer can also be used to refer to another pointer function. A pointer can be incremented/decremented, i.e., to point to the next/ previous memory location. The purpose of pointer is to save memory space and achieve faster execution time.

How does Pointer Work?

If we declare a variable v of type int, v will actually store a value.

```
int v = 0;
```

v is equal to zero now.

DIWAKAR EDUCATION HUB

However, each variable, apart from value, also has its address (or, simply put, where it is located in the memory). The address can be retrieved by putting an ampersand (&) before the variable name.

```
&v
```

If We print the address of a variable on the screen, it will look like a totally random number (moreover, it can be different from run to run).

```
#include <stdio.h>

int main() {
    int v = 0;
    printf("%d\n", &v);
    return 0;
}
```

The output of this program is -480613588.

Now, what is a pointer? Instead of storing a value, a pointer will y store the address of a variable.

Int *y = &v;

VARIABLE	POINTER
A **value** stored in a **named** storage/memory address	A **variable** that **points to** the storage/memory address of **another** variable

Declaring a pointer

Like variables, pointers have to be declared before they can be used in our program. Pointers can be named anything We want as long as they obey C's naming rules. A pointer declaration has the following form.

```
data_type * pointer_variable_name;
```

Here,

- **data_type** is the pointer's base type of C's variable types and indicates the type of the variable that the pointer points to.
- The asterisk (*: the same asterisk used for multiplication) which is indirection operator, declares a pointer.

Some valid pointer declarations

```
int    *ptr_thing;          /* pointer to an integer */

int *ptr1,thing;/* ptr1 is a pointer to type integer and thing is an integer variable */

double    *ptr2;    /* pointer to a double */

float    *ptr3;      /* pointer to a float */

char    *ch1 ;       /* pointer to a character */

float  *ptr, variable;/*ptr is a pointer to type float and variable is an ordinary float variable */
```

Initialize a pointer

After declaring a pointer, we initialize it like standard variables with a variable address. If pointers are not uninitialized and used in the program, the results are unpredictable and potentially disastrous.

To get the address of a variable, we use the ampersand (&)operator, placed before the name of a variable whose address we need. Pointer initialization is done with the following syntax.

DIWAKAR EDUCATION HUB

pointer = &variable;

A simple program for pointer illustration is given below:

```
#include <stdio.h>
int main()
{
  int a=10;    //variable declaration
  int *p;      //pointer variable declaration
  p=&a;        //store address of variable a in pointer p
  printf("Address stored in a variable p is:%x\n",p);  //accessing the address
  printf("Value stored in a variable p is:%d\n",*p);   //accessing the value
  return 0;
}
```

Output:

```
Address stored in a variable p is:60ff08
Value stored in a variable p is:10
```

Operator	Meaning
*	Serves 2 purpose • Declaration of a pointer • Returns the value of the referenced variable

DIWAKAR EDUCATION HUB

&	Serves only 1 purpose ➢ Returns the address of a variable

Types of a pointer

Null pointer

We can create a null pointer by assigning null value during the pointer declaration. This method is useful when We do not have any address assigned to the pointer. A null pointer always contains value 0.

Following program illustrates the use of a null pointer:

```
#include <stdio.h>

int main()

{

        int *p = NULL;        //null pointer

        printf("The value inside variable p is:\n%x",p);

        return 0;

}
```

Output:

```
The value inside variable p is:

0
```

Void Pointer

DIWAKAR EDUCATION HUB

In C programming, a void pointer is also called as a generic pointer. It does not have any standard data type. A void pointer is created by using the keyword void. It can be used to store an address of any variable.

Following program illustrates the use of a void pointer:

```
#include <stdio.h>

int main()

{

void *p = NULL;      //void pointer

printf("The size of pointer is:%d\n",sizeof(p));

return 0;

}
```

Output:

```
The size of pointer is:4
```

Wild pointer

A pointer is said to be a wild pointer if it is not being initialized to anything. These types of pointers are not efficient because they may point to some unknown memory location which may cause problems in our program and it may lead to crashing of the program. One should always be careful while working with wild pointers.

Following program illustrates the use of wild pointer:

```
#include <stdio.h>

int main()

{
```

DIWAKAR EDUCATION HUB

```
int *p;          //wild pointer

printf("\n%d",*p);

return 0;

}
```

Output

```
timeout: the monitored command dumped core

sh: line 1: 95298 Segmentation fault   timeout 10s main
```

Other types of pointers in 'c' are as follows:

- Dangling pointer
- Complex pointer
- Near pointer
- Far pointer
- Huge pointer

Direct and Indirect Access Pointers

In C, there are two equivalent ways to access and manipulate a variable content

- Direct access: we use directly the variable name
- Indirect access: we use a pointer to the variable

understand this with the help of program below

```
#include <stdio.h>

/* Declare and initialize an int variable */

int var = 1;
```

DIWAKAR EDUCATION HUB

```
/* Declare a pointer to int */

int *ptr;

int main( void )

{

/* Initialize ptr to point to var */

ptr = &var;

/* Access var directly and indirectly */

printf("\nDirect access, var = %d", var);

printf("\nIndirect access, var = %d", *ptr);

/* Display the address of var two ways */

printf("\n\nThe address of var = %d", &var);

printf("\nThe address of var = %d\n", ptr);

/*change the content of var through the pointer*/

*ptr=48;

printf("\nIndirect access, var = %d", *ptr);

return 0;}
```

After compiling the program without any errors, the result is:

```
Direct access, var = 1

Indirect access, var = 1

The address of var = 4202496

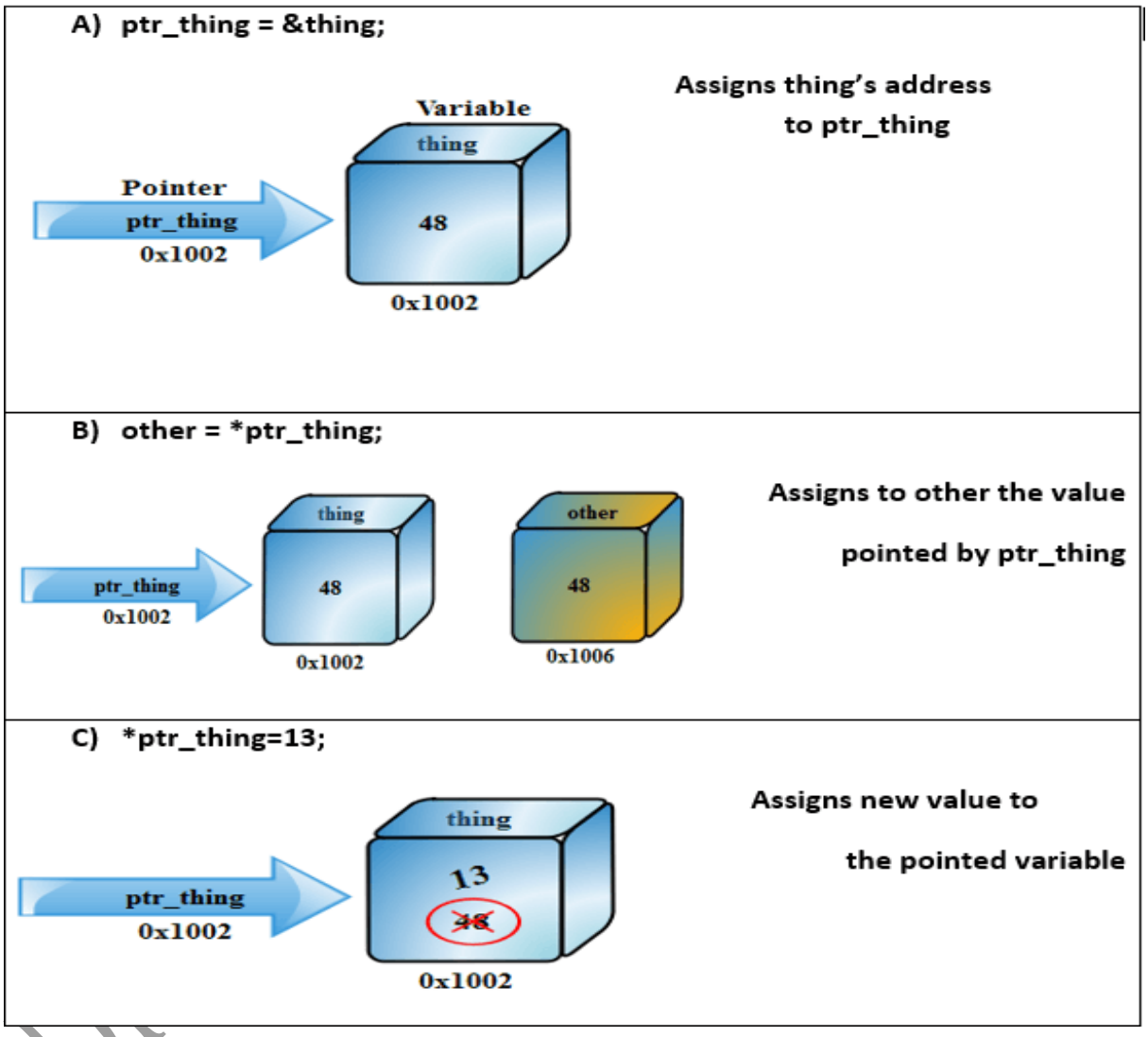
The address of var = 4202496
```

DIWAKAR EDUCATION HUB

Indirect access, var = 48

Pointers Arithmetic

The pointer operations are summarized in the following figure



Pointer Operations

Priority operation (precedence)

When working with pointers, we must observe the following priority rules:

- The operators * and & have the same priority as the unary operators (the negation!, the incrementation++, decrement--).
- In the same expression, the unary operators *, &,!, ++, - are evaluated from right to left.

If a P pointer points to an X variable, then * P can be used wherever X can be written.

The following expressions are equivalent:

int X =10

int *P = &Y;

For the above code, below expressions are true

Expression	Equivalent Expression
Y=*P+1	Y=X+1
*P=*P+10	X=X+10
*P+=2	X+=2
++*P	++X
(*P)++	X++

In the latter case, parentheses are needed: as the unary operators * and ++ are evaluated from right to left, without the parentheses the pointer P would be incremented, not the object on which P points.

Below table shows the arithmetic and basic operation that can be used when dealing with pointers

Operation	**Explanation**
Assignment	int *P1,*P2 P1=P2; P1 and P2 point to the same integer variable
Incrementation and decrementation	Int *P1; P1++;P1-- ;
Adding an offset (Constant)	This allows the pointer to move N elements in a table. The pointer will be increased or decreased by N times the number of byte (s) of the type of the variable. P1+5;

Pointers and Arrays

Traditionally, we access the array elements using its index, but this method can be eliminated by using pointers. Pointers make it easy to access each array element.

```
#include <stdio.h>
int main()
{
   int a[5]={1,2,3,4,5};   //array initialization
   int *p;     //pointer declaration
          /*the ptr points to the first element of the array*/
```

DIWAKAR EDUCATION HUB

```
   p=a; /*We can also type simply ptr==&a[0] */


   printf("Printing the array elements using pointer\n");
   for(int i=0;i<5;i++)    //loop for traversing array elements
   {
        printf("\n%x",*p);  //printing array elements
        p++;    //incrementing to the next element, you can also write p=p+1
   }
   return 0;
}
```

Output

```
1
2
3
4
5
```

Adding a particular number to a pointer will move the pointer location to the value obtained by an addition operation. Suppose p is a pointer that currently points to the memory location 0 if we perform following addition operation, p+1 then it will execute in this manner:

DIWAKAR EDUCATION HUB



Pointer Addition/Increment

Since p currently points to the location 0 after adding 1, the value will become 1, and hence the pointer will point to the memory location 1.

Pointers and Strings

A string is an array of char objects, ending with a null character '\ 0'. We can manipulate strings using pointers. Here is an example that explains this section

```
#include <stdio.h>

#include <string.h>

int main()

{

char str[]="Hello Manish99";

char *p;

p=str;

printf("First character is:%c\n",*p);

p =p+1;

printf("Next character is:%c\n",*p);

printf("Printing all the characters in a string\n");

p=str;  //reset the pointer

for(int i=0;i<strlen(str);i++)
```

DIWAKAR EDUCATION HUB

```
{

printf("%c\n",*p);

p++;

}

return 0;

}
```

Output

```
First character is:H

Next character is:e

Printing all the characters in a string

H
e
l
l
o

M
a
n
i
s
h
9
9
```

Another way to deal strings is with an array of pointers like in the following program:

```
#include <stdio.h>
```

DIWAKAR EDUCATION HUB

```
int main(){
char *materials[ ] = {  "iron",  "copper",  "gold"};
printf("Please remember these materials :\n");
int i ;
for (i = 0; i < 3; i++) {
  printf("%s\n", materials[ i ]);}
  return 0;}
```

Output:

```
Please remember these materials:
iron
copper
gold
```

Advantages of Pointers

- Pointers are useful for accessing memory locations.
- Pointers provide an efficient way for accessing the elements of an array structure.
- Pointers are used for dynamic memory allocation as well as deallocation.
- Pointers are used to form complex data structures such as linked list, graph, tree, etc.

Disadvantages of Pointers

1. Pointers are a little complex to understand.
2. Pointers can lead to various errors such as segmentation faults or can access a memory location which is not required at all.

DIWAKAR EDUCATION HUB

3. If an incorrect value is provided to a pointer, it may cause memory corruption.
4. Pointers are also responsible for memory leakage.
5. Pointers are comparatively slower than that of the variables.
6. Programmers find it very difficult to work with the pointers; therefore it is programmer's responsibility to manipulate a pointer carefully.

Summary

- A pointer is nothing but a memory location where data is stored.
- A pointer is used to access the memory location.
- There are various types of pointers such as a null pointer, wild pointer, void pointer and other types of pointers.
- Pointers can be used with array and string to access elements more efficiently.
- We can create function pointers to invoke a function dynamically.
- Arithmetic operations can be done on a pointer which is known as pointer arithmetic.
- Pointers can also point to function which make it easy to call different functions in the case of defining an array of pointers.
- When We want to deal different variable data type, We can use a typecast void pointer.

Functions in C

A **function** is a block of code that performs a particular task.

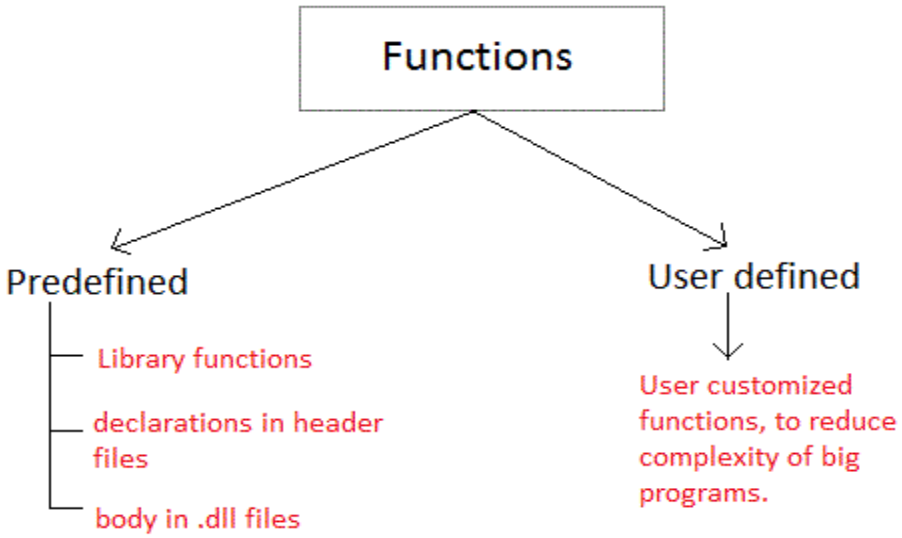
There are many situations where we might need to write same line of code for more than once in a program. This may lead to unnecessary repetition of code, bugs and even becomes boring for the programmer. So, C language provides an approach in which We can declare and define a group of statements once in the form of a function and it can be called and used whenever required.

These functions defined by the user are also know as **User-defined Functions**

DIWAKAR EDUCATION HUB

C functions can be classified into two categories,

- **Library functions**
- **User-defined functions**



Library functions are those functions which are already defined in C library, example printf(), scanf(), strcat() etc. We just need to include appropriate header files to use these functions. These are already declared and defined in C libraries.

A **User-defined functions** on the other hand, are those functions which are defined by the user at the time of writing program. These functions are made for code reusability and for saving time and space.

Benefits of Using Functions

1. It provides modularity to Wer program's structure.
2. It makes our code reusable. We just have to call the function by its name to use it, wherever required.
3. In case of large programs with thousands of code lines, debugging and editing becomes easier if We use functions.

DIWAKAR EDU

4. It makes the program more readable and easy to understand.

Function Declaration

General syntax for function declaration is,

```
returntype functionName(type1 parameter1, type2 parameter2,...);
```

Like any variable or an array, a function must also be declared before its used. Function declaration informs the compiler about the function name, parameters is accept, and its return type. The actual body of the function can be defined separately. It's also called as **Function Prototyping**. Function declaration consists of 4 parts.

1. returntype
2. function name
3. parameter list
4. terminating semicolon

returntype

When a function is declared to perform some sort of calculation or any operation and is expected to provide with some result at the end, in such cases, a return statement is added at the end of function body. Return type specifies the type of value(int, float, char, double) that function is expected to return to the program which called the function.

Note: In case our function doesn't return any value, the return type would be void.

functionName

Function name is an identifier and it specifies the name of the function. The function name is any valid C identifier and therefore must follow the same naming rules like other variables in C language.

parameter list

DIWAKAR EDUCATION HUB

The parameter list declares the type and number of arguments that the function expects when it is called. Also, the parameters in the parameter list receives the argument values when the function is called. They are often referred as **formal parameters**.

An Example

a simple program with a main() function, and a user defined function to multiply two numbers, which will be called from the main() function.

```
#include<stdio.h>


int multiply(int a, int b);     // function declaration
```

```
int main()
{
   int i, j, result;
   printf("Please enter 2 numbers you want to multiply...");
   scanf("%d%d", &i, &j);
   result = multiply(i, j);      // function call
   printf("The result of muliplication is: %d", result);
   return 0;
}
int multiply(int a, int b)
{
   return (a*b);      // function defintion, this can be done in one line
```

DIWAKAR EDUCATION HUB

```
}
```

Function definition Syntax

Just like in the example above, the general syntax of function definition is,

```
returntype functionName(type1 parameter1, type2 parameter2,...)
{
   // function body goes here
}
```

The first line *returntype* **functionName(type1 parameter1, type2 parameter2,...)** is known as **function header** and the statement(s) within curly braces is called **function body**.

Note: While defining a function, there is no semicolon(;) after the parenthesis in the function header, unlike while declaring the function or calling the function.

functionbody

The function body contains the declarations and the statements(algorithm) necessary for performing the required task. The body is enclosed within curly braces { ... } and consists of three parts.

a) **local** variable declaration(if required).

b) **function statements** to perform the task inside the function.

c) a **return** statement to return the result evaluated by the function(if return type is void, then no return statement is required).

Calling a function

When a function is called, control of the program gets transferred to the function.

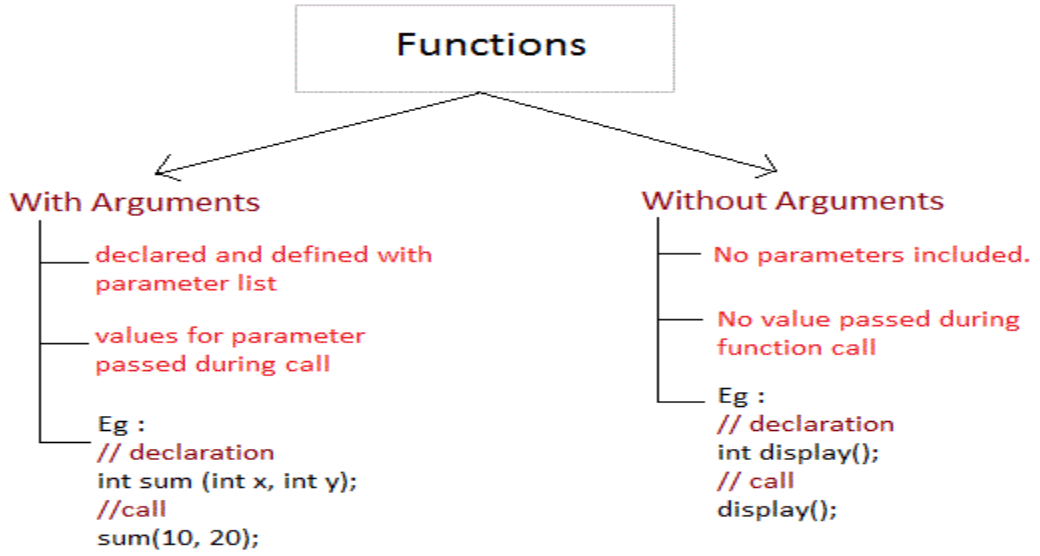
functionName(argument1, argument2,...);

DIWAKAR EDUCATION HUB

In the example above, the statement multiply(i, j); inside the main() function is function call.

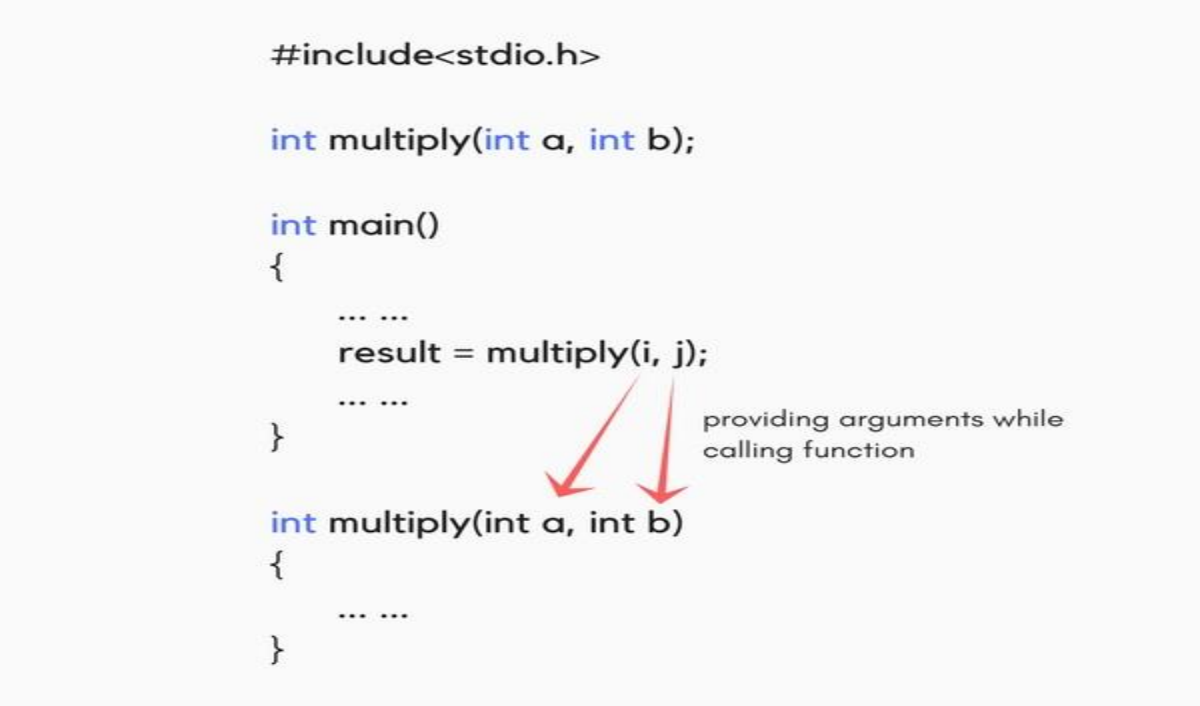
Passing Arguments to a function

Arguments are the values specified during the function call, for which the formal parameters are declared while defining the function.



It is possible to have a function with parameters but no return type. It is not necessary, that if a function accepts parameter(s), it must return a result too.

DIWAKAR

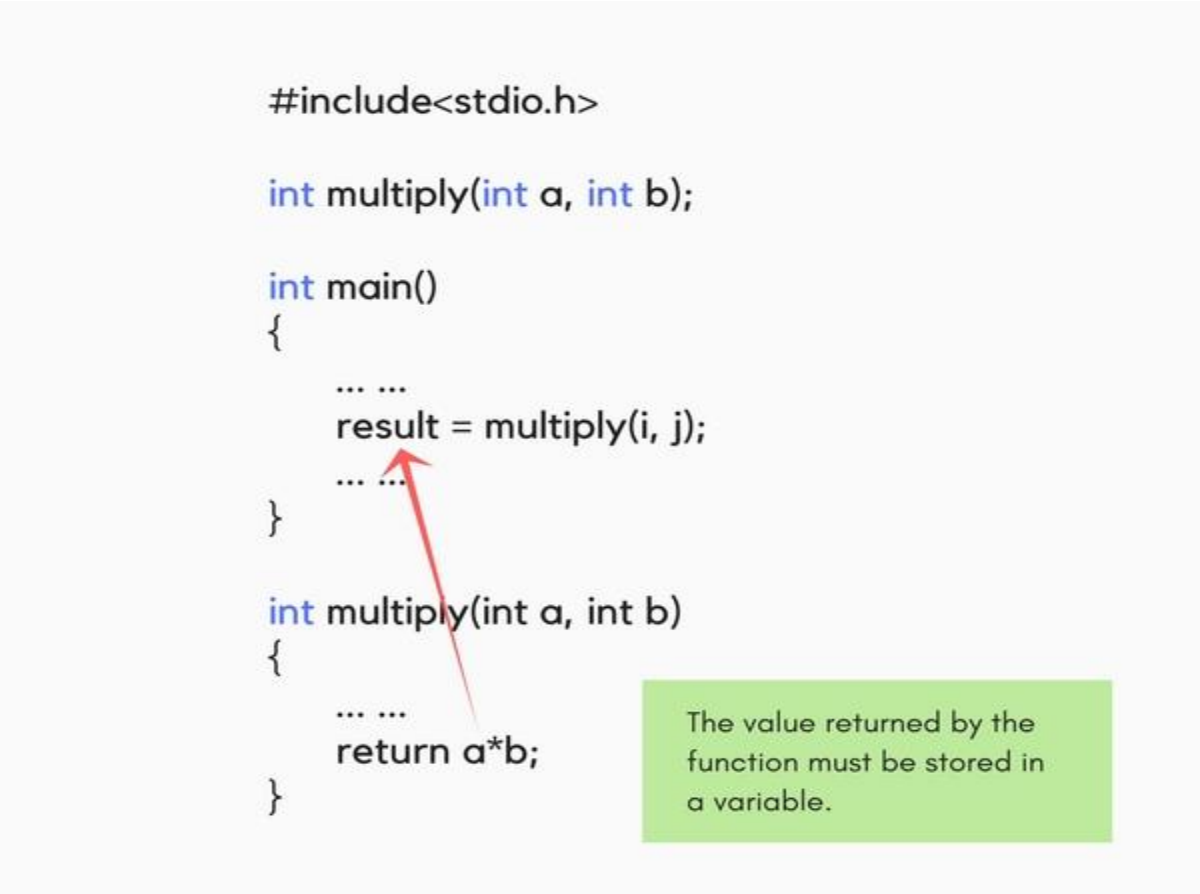


While declaring the function, we have declared two parameters a and b of type int. Therefore, while calling that function, we need to pass two arguments, else we will get compilation error. And the two arguments passed should be received in the function definition, which means that the function header in the function definition should have the two parameters to hold the argument values. These received arguments are also known as **formal parameters**. The name of the variables while declaring, calling and defining a function can be different.

Returning a value from function

A function may or may not return a result. But if it does, we must use the return statement to output the result. return statement also ends the function execution, hence it must be the last statement of any function. If We write any statement after the return statement, it won't be executed.

DIWAKAR EDUCA



The datatype of the value returned using the return statement should be same as the return type mentioned at function declaration and definition. If any of it mismatches, We will get compilation error.

In the next tutorial, we will learn about the different types of user defined functions in C language and the concept of Nesting of functions which is used in recursion.

Type of User-defined Functions in C

There can be 4 different types of user-defined functions, they are:

a) Function with no arguments and no return value

b) Function with no arguments and a return value

c) Function with arguments and no return value

DIWAKAR LL

d) Function with arguments and a return value

Below, about all these types, along with program examples.

Function with no arguments and no return value

Such functions can either be used to display information or they are completely dependent on user inputs.

Below is an example of a function, which takes 2 numbers as input from user, and display which is the greater number.

```
#include<stdio.h>

void greatNum();      // function declaration

int main()

{

   greatNum();       // function call

   return 0;

}

void greatNum()      // function definition

{

   int i, j;

   printf("Enter 2 numbers that you want to compare...");

   scanf("%d%d", &i, &j);

   if(i > j) {

      printf("The greater number is: %d", i);

   }
```

DIWAKAR EDUCATION HUB

```
    else {
        printf("The greater number is: %d", j);
    }
}
```

Function with no arguments and a return value

We have modified the above example to make the function greatNum() return the number which is greater amongst the 2 input numbers.

```
#include<stdio.h>
int greatNum();      // function declaration
int main()
{
    int result;
    result = greatNum();      // function call
    printf("The greater number is: %d", result);
    return 0;
}
int greatNum()      // function definition
{
    int i, j, greaterNum;
    printf("Enter 2 numbers that you want to compare...");
    scanf("%d%d", &i, &j);
```

DIWAKAR EDUCATION HUB

```
   if(i > j) {
      greaterNum = i;
   }
   else {
      greaterNum = j;
   }
   // returning the result
   return greaterNum;
}
```

Function with arguments and no return value

We are using the same function as example again and again, to demonstrate that to solve a problem there can be many different ways.

This time, we have modified the above example to make the function greatNum() take two int values as arguments, but it will not be returning anything.

```
#include<stdio.h>
void greatNum(int a, int b);      // function declaration
int main()
{
   int i, j;
   printf("Enter 2 numbers that you want to compare...");
   scanf("%d%d", &i, &j);
```

DIWAKAR EDUCATION HUB

```
   greatNum(i, j);       // function call
   return 0;
}
void greatNum(int x, int y)       // function definition
{
   if(x > y) {
      printf("The greater number is: %d", x);
   }
   else {
      printf("The greater number is: %d", y);
   }
}
```

Function with arguments and a return value

This is the best type, as this makes the function completely independent of inputs and outputs, and only the logic is defined inside the function body.

```
#include<stdio.h>
int greatNum(int a, int b);       // function declaration
int main()
{
   int i, j, result;
   printf("Enter 2 numbers that you want to compare...");
```

DIWAKAR EDUCATION HUB

```
   scanf("%d%d", &i, &j);

   result = greatNum(i, j); // function call

   printf("The greater number is: %d", result);

   return 0;

}

int greatNum(int x, int y)       // function definition

{

   if(x > y) {

      return x;

   }

   else {

      return y;

   }

}
```

Nesting of Functions

C language also allows nesting of functions i.e to use/call one function inside another function's body. We must be careful while using nested functions, because it may lead to infinite nesting.

```
function1()

{

   // function1 body here

   function2();
```

DIWAKAR EDUCATION HUB

```
   // function1 body here

}
```

If function2() also has a call for function1() inside it, then in that case, it will lead to an infinite nesting. They will keep calling each other and the program will never terminate.

consider that inside the main() function, function1() is called and its execution starts, then inside function1(), we have a call for function2(), so the control of program will go to the function2(). But as function2() also has a call to function1() in its body, it will call function1(), which will again call function2(), and this will go on for infinite times, until We forcefully exit from program execution.

Recursion

Recursion is a special way of nesting functions, where a function calls itself inside it. We must have certain conditions in the function to break out of the recursion, otherwise recursion will occur infinite times.

```
function1()

{

   // function1 body

   function1();

   // function1 body

}
```

Example: Factorial of a number using Recursion

```
#include<stdio.h>

int factorial(int x);      //declaring the function
```

DIWAKAR EDUCATION HUB

```
void main()
{
   int a, b;
   printf("Enter a number...");
   scanf("%d", &a);
   b = factorial(a);      //calling the function named factorial
   printf("%d", b);
}
int factorial(int x) //defining the function
{
   int r = 1;
   if(x == 1)
      return 1;
   else
      r = x*factorial(x-1);     //recursion, since the function calls itself

   return r;
}
```

DIWAKAR EDUCATION HUB

Similarly, there are many more applications of recursion in C language. Go to the programs section, to find out more programs using recursion.

File Handling in C

In programming, we may require some specific input data to be generated several numbers of times. Sometimes, it is not enough to only display the data on the console. The data to be displayed may be very large, and only a limited amount of data can be displayed on the console, and since the memory is volatile, it is impossible to recover the programmatically generated data again and again. However, if we need to do so, we may store it onto the local file system which is volatile and can be accessed every time. Here, comes the need of file handling in C.

File handling in C enables us to create, update, read, and delete the files stored on the local file system through our C program. The following operations can be performed on a file.

i. Creation of the new file
ii. Opening an existing file
iii. Reading from the file
iv. Writing to the file
v. Deleting the file

Functions for file handling

There are many functions in the C library to open, read, write, search and close the file. A list of file functions are given below:

No.	Function	Description
1	fopen()	opens new or existing file
2	fprintf()	write data into the file
3	fscanf()	reads data from the file

KAR EDUCATION HUB

4	fputc()	writes a character into the file
5	fgetc()	reads a character from file
6	fclose()	closes the file
7	fseek()	sets the file pointer to given position
8	fputw()	writes an integer to file
9	fgetw()	reads an integer from file
10	ftell()	returns current position
11	rewind()	sets the file pointer to the beginning of the file

Opening File: fopen()

We must open a file before it can be read, write, or update. The fopen() function is used to open a file. The syntax of the fopen() is given below.

- **FILE** *fopen(**const char** * filename, **const char** * mode);

The fopen() function accepts two parameters:

- The file name (string). If the file is stored at some specific location, then we must mention the path at which the file is stored. For example, a file name can be like **"c://some_folder/some_file.ext"**.
- The mode in which the file is to be opened. It is a string.

We can use one of the following modes in the fopen() function.

DIWAKAR EDU

Mode	Description
r	opens a text file in read mode
w	opens a text file in write mode
a	opens a text file in append mode
r+	opens a text file in read and write mode
w+	opens a text file in read and write mode
a+	opens a text file in read and write mode
rb	opens a binary file in read mode
wb	opens a binary file in write mode
ab	opens a binary file in append mode
rb+	opens a binary file in read and write mode
wb+	opens a binary file in read and write mode
ab+	opens a binary file in read and write mode

The fopen function works in the following way.

i. Firstly, It searches the file to be opened.

ii. Then, it loads the file from the disk and place it into the buffer. The buffer is used to provide efficiency for the read operations.

iii. It sets up a character pointer which points to the first character of the file.

Consider the following example which opens a file in write mode.

```
#include<stdio.h>
void main( )
{
FILE *fp ;
char ch ;
fp = fopen("file_handle.c","r") ;
while ( 1 )
{
ch = fgetc ( fp ) ;
if ( ch == EOF )
break ;
printf("%c",ch) ;
}
fclose (fp ) ;
}
```

DIWAKAR EDUCATION HUB

Output

The content of the file will be printed.

```
#include;

void main( )

{

FILE *fp; // file pointer

char ch;

fp = fopen("file_handle.c","r");

while ( 1 )

{

ch = fgetc ( fp ); //Each character of the file is read and stored in the character file.

if ( ch == EOF )

break;

printf("%c",ch);

}

fclose (fp );

}
```

Closing File: fclose()

The fclose() function is used to close a file. The file must be closed after performing all the operations on it. The syntax of fclose() function is given below:

1. **int** fclose(**FILE** *fp);

C fprintf() and fscanf()

C fprintf() and fscanf() example

DIWAKAR EDUCATION HUB

C fputc() and fgetc()

C fputc() and fgetc() example

C fputs() and fgets()

C fputs() and fgets() example

C fseek()

C fseek() example

C fprintf() and fscanf()

Writing File : fprintf() function

The fprintf() function is used to write set of characters into file. It sends formatted output to a stream.

Syntax:

```
int fprintf(FILE *stream, const char *format [, argument, ...])
```

Example:

```
#include <stdio.h>
main(){
  FILE *fp;
  fp = fopen("file.txt", "w");//opening file
  fprintf(fp, "Hello file by fprintf...\n");//writing data into file
```

DIWAKAR EDUCATION HUB

```
  fclose(fp);//closing file

}
```

Reading File : fscanf() function

The fscanf() function is used to read set of characters from file. It reads a word from the file and returns EOF at the end of file.

Syntax:

```
int fscanf(FILE *stream, const char *format [, argument, ...])
```

Example:

```
#include <stdio.h>

main(){

  FILE *fp;

  char buff[255];//creating char array to store data of file

  fp = fopen("file.txt", "r");

  while(fscanf(fp, "%s", buff)!=EOF){

  printf("%s ", buff );

  }

  fclose(fp);
}
```

Output:

```
Hello file by fprintf...
```

C File Example: Storing employee information

DIWAKAR EDUCATION HUB

A file handling example to store employee information as entered by user from console. We are going to store id, name and salary of the employee.

```
#include <stdio.h>

void main()

{

   FILE *fptr;

   int id;

   char name[30];

   float salary;

   fptr = fopen("emp.txt", "w+");/*  open for writing */

   if (fptr == NULL)

   {

      printf("File does not exists \n");

      return;

   }

   printf("Enter the id\n");

   scanf("%d", &id);

   fprintf(fptr, "Id= %d\n", id);

   printf("Enter the name \n");

   scanf("%s", name);

   fprintf(fptr, "Name= %s\n", name);
```

DIWAKAR EDUCATION HUB

```
    printf("Enter the salary\n");
    scanf("%f", &salary);
    fprintf(fptr, "Salary= %.2f\n", salary);
    fclose(fptr);
}
```

Output:

```
Enter the id
1
Enter the name
sonoo
Enter the salary
120000
```

Now open file from current directory. For windows operating system, go to TC\bin directory, We will see emp.txt file. It will have following information.

emp.txt

Id= 1

Name= sonoo

Salary= 120000

Command Line Arguments in C

The arguments passed from command line are called command line arguments. These arguments are handled by main() function.

DIWAKAR EDUCATION HUB

To support command line argument, We need to change the structure of main() function as given below.

```
int main(int argc, char *argv[] )
```

Here, **argc** counts the number of arguments. It counts the file name as the first argument.

The **argv[]** contains the total number of arguments. The first argument is the file name always.

Example

the example of command line arguments where we are passing one argument with file name.

```
#include <stdio.h>

void main(int argc, char *argv[] ) {


  printf("Program name is: %s\n", argv[0]);


  if(argc < 2){

    printf("No argument passed through command line.\n");

  }

  else{

    printf("First argument is: %s\n", argv[1]);

  }

}
```

DIWAKAR EDUCATION HUB

Run this program as follows in Linux:

```
./program hello
```

Run this program as follows in Windows from command line:

```
program.exe hello
```

Output:

```
Program name is: program

First argument is: hello
```

If we pass many arguments, it will print only one.

```
./program hello c how r u
```

Output:

```
Program name is: program

First argument is: hello
```

But if we pass many arguments within double quote, all arguments will be treated as a single argument only.

```
./program "hello c how r u"
```

Output:

```
Program name is: program

First argument is: hello c how r u
```

We can write our program to print all the arguments. In this program, we are printing only argv[1], that is why it is printing only one argument.

C - Preprocessors

DIWAKAR EDUCATION HUB

The **C Preprocessor** is not a part of the compiler, but is a separate step in the compilation process. In simple terms, a C Preprocessor is just a text substitution tool and it instructs the compiler to do required pre-processing before the actual compilation. We'll refer to the C Preprocessor as CPP.

All preprocessor commands begin with a hash symbol (#). It must be the first nonblank character, and for readability, a preprocessor directive should begin in the first column. The following section lists down all the important preprocessor directives −

Sr.No.	Directive & Description
1	**#define** Substitutes a preprocessor macro.
2	**#include** Inserts a particular header from another file.
3	**#undef** Undefines a preprocessor macro.
4	**#ifdef** Returns true if this macro is defined.
5	**#ifndef** Returns true if this macro is not defined.
6	**#if** Tests if a compile time condition is true.

DIWAKAR EDUCATION HUB

7	**#else** The alternative for #if.
8	**#elif** #else and #if in one statement.
9	**#endif** Ends preprocessor conditional.
10	**#error** Prints error message on stderr.
11	**#pragma** Issues special commands to the compiler, using a standardized method.

Preprocessors Examples

Analyze the following examples to understand various directives.

#define MAX_ARRAY_LENGTH 20

This directive tells the CPP to replace instances of MAX_ARRAY_LENGTH with 20. Use *#define* for constants to increase readability.

```
#include <stdio.h>
#include "myheader.h"
```

These directives tell the CPP to get stdio.h from **System Libraries** and add the text to the current source file. The next line tells CPP to get **myheader.h** from the local directory and add the content to the current source file.

```
#undef  FILE_SIZE
#define FILE_SIZE 42
```

It tells the CPP to undefine existing FILE_SIZE and define it as 42.

DIWAKAR EDUCATION HUB

```
#ifndef MESSAGE
  #define MESSAGE "You wish!"
#endif
```

It tells the CPP to define MESSAGE only if MESSAGE isn't already defined.

```
#ifdef DEBUG
  /* Your debugging statements here */
#endif
```

It tells the CPP to process the statements enclosed if DEBUG is defined. This is useful if We pass the *-DDEBUG* flag to the gcc compiler at the time of compilation. This will define DEBUG, so We can turn debugging on and off on the fly during compilation.

Predefined Macros

ANSI C defines a number of macros. Although each one is available for use in programming, the predefined macros should not be directly modified.

Sr.No.	Macro & Description
1	**__DATE__** The current date as a character literal in "MMM DD YYYY" format.
2	**__TIME__** The current time as a character literal in "HH:MM:SS" format.
3	**__FILE__** This contains the current filename as a string literal.
4	**__LINE__** This contains the current line number as a decimal constant.

DIWAKAR EDUCATION HUB

5	**__STDC__** Defined as 1 when the compiler complies with the ANSI standard.

try the following example −

```
#include <stdio.h>

int main() {

   printf("File :%s\n", __FILE__ );
   printf("Date :%s\n", __DATE__ );
   printf("Time :%s\n", __TIME__ );
   printf("Line :%d\n", __LINE__ );
   printf("ANSI :%d\n", __STDC__ );

}
```

When the above code in a file **test.c** is compiled and executed, it produces the following result −

```
File :test.c
Date :Jun 2 2012
Time :03:36:24
Line :8
ANSI :1
```

Preprocessor Operators

The C preprocessor offers the following operators to help create macros −

The Macro Continuation (\) Operator

A macro is normally confined to a single line. The macro continuation operator (\) is used to continue a macro that is too long for a single line. For example −

```
#define  message_for(a, b)  \
   printf(#a " and " #b ": We love you!\n")
```

DIWAKAR EDUCATION HUB

The Stringize (#) Operator

The stringize or number-sign operator ('#'), when used within a macro definition, converts a macro parameter into a string constant. This operator may be used only in a macro having a specified argument or parameter list. For example –

```
#include <stdio.h>

#define  message_for(a, b)  \
  printf(#a " and " #b ": We love you!\n")

int main(void) {
  message_for(Carole, Debra);
  return 0;
}
```

When the above code is compiled and executed, it produces the following result –

Carole and Debra: We love you!

The Token Pasting (##) Operator

The token-pasting operator (##) within a macro definition combines two arguments. It permits two separate tokens in the macro definition to be joined into a single token. For example –

```
#include <stdio.h>

#define tokenpaster(n) printf ("token" #n " = %d", token##n)

int main(void) {
  int token34 = 40;
  tokenpaster(34);
  return 0;
}
```

DIWAKAR EDUCATION HUB

When the above code is compiled and executed, it produces the following result –

token34 = 40

It happened so because this example results in the following actual output from the preprocessor −

```
printf ("token34 = %d", token34);
```

This example shows the concatenation of token##n into token34 and here we have used both **stringize** and **token-pasting**.

The Defined() Operator

The preprocessor **defined** operator is used in constant expressions to determine if an identifier is defined using #define. If the specified identifier is defined, the value is true (non-zero). If the symbol is not defined, the value is false (zero). The defined operator is specified as follows −

```
#include <stdio.h>

#if !defined (MESSAGE)
  #define MESSAGE "You wish!"
#endif

int main(void) {
  printf("Here is the message: %s\n", MESSAGE);
  return 0;
}
```

When the above code is compiled and executed, it produces the following result –

Here is the message: You wish!

Parameterized Macros

One of the powerful functions of the CPP is the ability to simulate functions using parameterized macros. For example, we might have some code to square a number as follows −

DIWAKAR EDUCATION HUB

```
int square(int x) {
  return x * x;
}
```

We can rewrite above the code using a macro as follows –

```
#define square(x) ((x) * (x))
```

Macros with arguments must be defined using the **#define** directive before they can be used. The argument list is enclosed in parentheses and must immediately follow the macro name. Spaces are not allowed between the macro name and open parenthesis. For example –

```
#include <stdio.h>

#define MAX(x,y) ((x) > (y) ? (x) : (y))

int main(void) {
  printf("Max between 20 and 10 is %d\n", MAX(10, 20));
  return 0;
}
```

When the above code is compiled and executed, it produces the following result –

Max between 20 and 10 is 20

Object-oriented programming (OOP)

Object-oriented programming (OOP) is a computer programming model that organizes software design around data, or objects, rather than functions and logic. An object can be defined as a data field that has unique attributes and behavior.

OOP focuses on the objects that developers want to manipulate rather than the logic required to manipulate them. This approach to programming is well-suited for programs that are large, complex and actively updated or maintained.

UNAKAR EDUCATION HUB

The organization of an object-oriented program also makes the method beneficial to collaborative development, where projects are divided into groups.

Additional benefits of OOP include code reusability, scalability and efficiency. Even when using microservices, developers should continue to apply the principles of OOP.

The first step in OOP is to collect all of the objects a programmer wants to manipulate and identify how they relate to each other -- an exercise often known as data modeling.

Examples of an object can range from physical entities, such as a human being who is described by properties like name and address, down to small computer programs, such as widgets.

Once an object is known, it is labeled with a class of objects that defines the kind of data it contains and any logic sequences that can manipulate it. Each distinct logic sequence is known as a method. Objects can communicate with well-defined interfaces called messages.

Principles of OOP

Object-oriented programming is based on the following principles:

Encapsulation. The implementation and state of each object are privately held inside a defined boundary, or class. Other objects do not have access to this class or the authority to make changes but are only able to call a list of public functions, or methods. This characteristic of data hiding provides greater program security and avoids unintended data corruption.

Hiding the implementation details of the class from the user through an object's methods is known as data encapsulation. In object oriented programming, it binds the code and the data together and keeps them safe from outside interference.

DIWAKAR EDUCATION HUB

Abstraction. Objects only reveal internal mechanisms that are relevant for the use of other objects, hiding any unnecessary implementation code. This concept helps developers more easily make changes and additions over time.

Inheritance. Relationships and subclasses between objects can be assigned, allowing developers to reuse a common logic while still maintaining a unique hierarchy. This property of OOP forces a more thorough data analysis, reduces development time and ensures a higher level of accuracy.

Inheritance as in general terms is the process of acquiring properties. In OOP one object inherit the properties of another object.

Polymorphism. Objects can take on more than one form depending on the context. The program will determine which meaning or usage is necessary for each execution of that object, cutting down the need to duplicate code.

Polymorphism is the process of using same method name by multiple classes and redefines methods for the derived classes.

Object-oriented programming languages

While Simula is credited as the first object-oriented programming language, the most popular OOP languages are:

- Java
- JavaScript
- Python
- C++
- Visual Basic .NET
- Ruby
- Scala
- PHP

OOPSLA is the annual conference for Object-Oriented Programming Systems, Languages and Applications.

DIWAKAR EDUCATION HUB

Criticism of OOP

The object-oriented programming model has been criticized by developers for multiple reasons. The largest concern is that OOP overemphasizes the data component of software development and does not focus enough on computation or algorithms. Additionally, OOP code may be more complicated to write and take longer to compile.

Alternative methods to OOP include:

- functional programming
- structured programming
- imperative programming

Most advanced programming languages give developers the option to combine these models.

Object Oriented Programming Paradigm (OOPP)

The Object Oriented programming paradigm plays an important role in human computer interface. It has different components that takes real world objects and performs actions on them, making live interactions between man and the machine. Following are the components of OOPP –

- This paradigm describes a real-life system where interactions are among real objects.
- It models applications as a group of related objects that interact with each other.
- The programming entity is modeled as a class that signifies the collection of related real world objects.
- Programming starts with the concept of real world objects and classes.
- Application is divided into numerous packages.
- A package is a collection of classes.

DIWAKAR EDUCATION HUB

- A class is an encapsulated group of similar real world objects.

Objects

Real-world objects share two characteristics – They all have state and behavior. see the following pictorial example to understand Objects.



In the above diagram, the object 'Dog' has both state and behavior.

An object stores its information in attributes and discloses its behavior through methods. now discuss in brief the different components of object oriented programming.

Public Interface

The point where the software entities interact with each other either in a single computer or in a network is known as pubic interface. This help in data security. Other objects can change the state of an object in an interaction by using only those methods that are exposed to the outer world through a public interface.

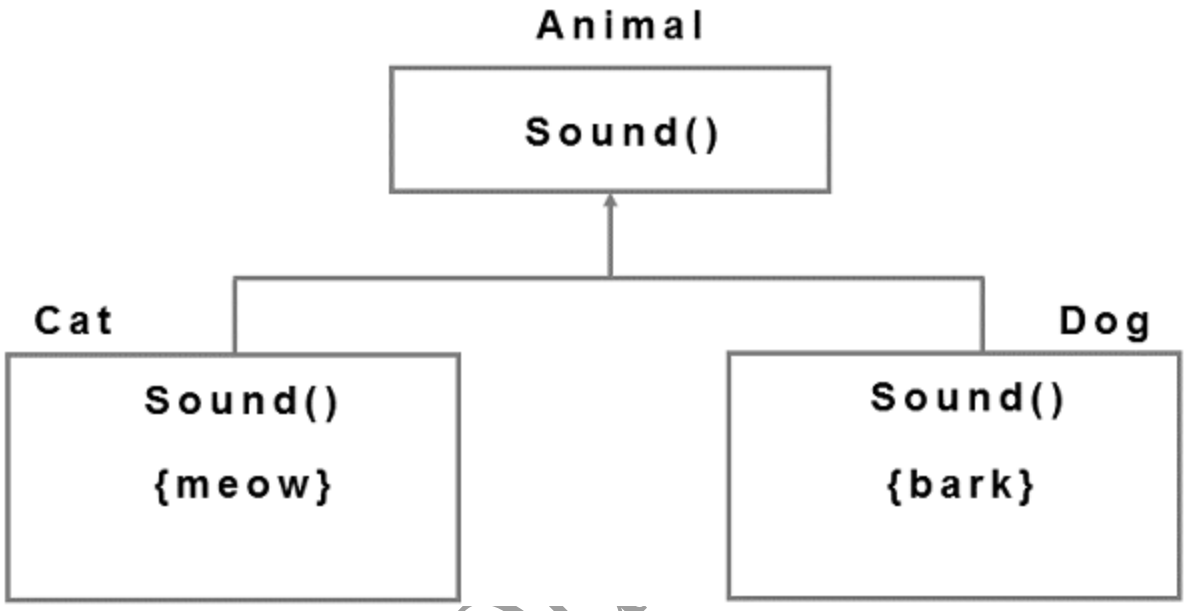
Class

A class is a group of objects that has mutual methods. It can be considered as the blueprint using which objects are created.

DIWAKAR EDU HUB

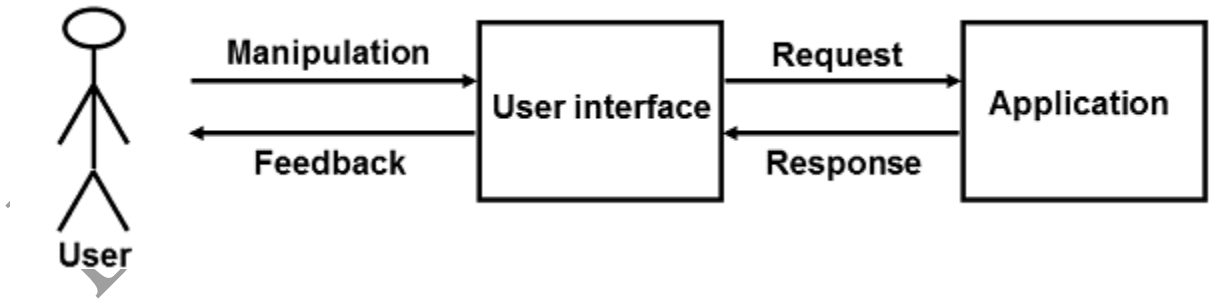
Classes being passive do not communicate with each other but are used to instantiate objects that interact with each other.

Example



Object Oriented Modeling of User Interface Design

Object oriented interface unites users with the real world manipulating software objects for designing purpose. see the diagram.



Interface design strive to make successful accomplishment of user's goals with the help of interaction tasks and manipulation.

While creating the OOM for interface design, first of all analysis of user requirements is done. The design specifies the structure and components

required for each dialogue. After that, interfaces are developed and tested against the Use Case. Example – Personal banking application.

The sequence of processes documented for every Use Case are then analyzed for key objects. This results into an object model. Key objects are called analysis objects and any diagram showing relationships between these objects is called object diagram.

C++ Identifiers

C++ identifiers in a program are used to refer to the name of the variables, functions, arrays, or other user-defined data types created by the programmer. They are the basic requirement of any language. Every language has its own rules for naming the identifiers.

In short, we can say that the C++ identifiers represent the essential elements in a program which are given below:

- Constants
- Variables
- Functions
- Labels
- Defined data types

Some naming rules are common in both C and C++. They are as follows:

- Only alphabetic characters, digits, and underscores are allowed.
- The identifier name cannot start with a digit, i.e., the first letter should be alphabetical. After the first letter, we can use letters, digits, or underscores.
- In C++, uppercase and lowercase letters are distinct. Therefore, we can say that C++ identifiers are case-sensitive.
- A declared keyword cannot be used as a variable name.

For example, suppose we have two identifiers, named as 'FirstName', and 'Firstname'. Both the identifiers will be different as the letter 'N' in the first case in

DIWAKAR EDUCATION HUB

uppercase while lowercase in second. Therefore, it proves that identifiers are case-sensitive.

Valid Identifiers

The following are the examples of valid identifiers are:

```
Result

Test2

_sum

power
```

Invalid Identifiers

The following are the examples of invalid identifiers:

```
Sum-1   // containing special character '-'

2data   // the first letter is a digit.

break   // use of a keyword.
```

Note: Identifiers cannot be used as the keywords. It may not conflict with the keywords, but it is highly recommended that the keywords should not be used as the identifier name. We should always use a consistent way to name the identifiers so that our code will be more readable and maintainable.

The major difference between C and C++ is the limit on the length of the name of the variable. ANSI C considers only the first 32 characters in a name while ANSI C++ imposes no limit on the length of the name.

Constants are the identifiers that refer to the fixed value, which do not change during the execution of a program. Both C and C++ support various kinds of literal constants, and they do have any memory location. For example, 123, 12.34, 037, 0X2, etc. are the literal constants.

DIWAKAR EDUCATION HUB

A simple example to understand the concept of identifiers.

```
#include <iostream>

using namespace std;

int main()

{

int a;

int A;

cout<<"Enter the values of 'a' and 'A'";

cin>>a;

cin>>A;

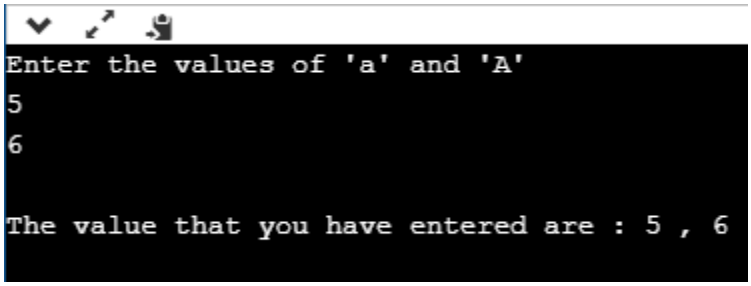
cout<<"\nThe values that you have entered are : "<<a<<" , "<<A;

return 0;

}
```

In the above code, we declare two variables 'a' and 'A'. Both the letters are same but they will behave as different identifiers. As we know that the identifiers are the case-sensitive so both the identifiers will have different memory locations.

Output



What are the keywords?

DIWAKAR EDUCATION HUB

Keywords are the reserved words that have a special meaning to the compiler. They are reserved for a special purpose, which cannot be used as the identifiers. For example, 'for', 'break', 'while', 'if', 'else', etc. are the predefined words where predefined words are those words whose meaning is already known by the compiler. Whereas, the identifiers are the names which are defined by the programmer to the program elements such as variables, functions, arrays, objects, classes.

Differences between Identifiers and Keywords

The following is the list of differences between identifiers and keywords:

Identifiers	Keywords
Identifiers are the names defined by the programmer to the basic elements of a program.	Keywords are the reserved words whose meaning is known by the compiler.
It is used to identify the name of the variable.	It is used to specify the type of entity.
It can consist of letters, digits, and underscore.	It contains only letters.
It can use both lowercase and uppercase letters.	It uses only lowercase letters.
No special character can be used except the underscore.	It cannot contain any special character.

The starting letter of identifiers can be lowercase, uppercase or underscore.	It can be started only with the lowercase letter.
It can be classified as internal and external identifiers.	It cannot be further classified.
Examples are test, result, sum, power, etc.	Examples are 'for', 'if', 'else', 'break', etc.

C++ Variable

A variable is a name of memory location. It is used to store data. Its value can be changed and it can be reused many times.

It is a way to represent memory location through symbol so that it can be easily identified.

the syntax to declare a variable:

```
type variable_list;
```

The example of declaring variable is given below:

```
int x;
float y;
char z;
```

Here, x, y, z are variables and int, float, char are data types.

We can also provide values while declaring the variables as given below:

```
int x=5,b=10;  //declaring 2 variable of integer type
```

DIWAKAR EDUCATION

```
float f=30.8;

char c='A';
```

Rules for defining variables

A variable can have alphabets, digits and underscore.

A variable name can start with alphabet and underscore only. It can't start with digit.

No white space is allowed within variable name.

A variable name must not be any reserved word or keyword e.g. char, float etc.

Valid variable names:

```
int a;

int _ab;

int a30;
```

Invalid variable names:

```
int 4;

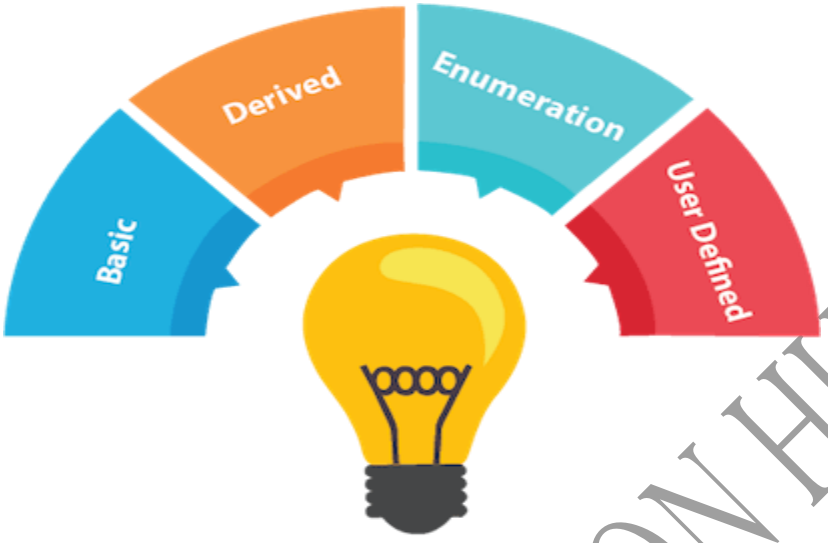
int x y;

int double;
```

C++ Data Types

A data type specifies the type of data that a variable can store such as integer, floating, character etc.

DIWAKAR EDUCATION HUB



Data Types in C++

There are 4 types of data types in C++ language.

Types	Data Types
Basic Data Type	int, char, float, double, etc
Derived Data Type	array, pointer, etc
Enumeration Data Type	enum
User Defined Data Type	structure

The memory size of basic data types may change according to 32 or 64 bit operating system.

the basic data types. It size is given according to 32 bit OS.

Data Types	Memory Size	Range
char	1 byte	-128 to 127
signed char	1 byte	-128 to 127
unsigned char	1 byte	0 to 127
short	2 byte	-32,768 to 32,767
signed short	2 byte	-32,768 to 32,767
unsigned short	2 byte	0 to 32,767
int	2 byte	-32,768 to 32,767
signed int	2 byte	-32,768 to 32,767
unsigned int	2 byte	0 to 32,767
short int	2 byte	-32,768 to 32,767
signed short int	2 byte	-32,768 to 32,767
unsigned short int	2 byte	0 to 32,767
long int	4 byte	
signed long int	4 byte	
unsigned long int	4 byte	

float	4 byte	
double	8 byte	
long double	10 byte	

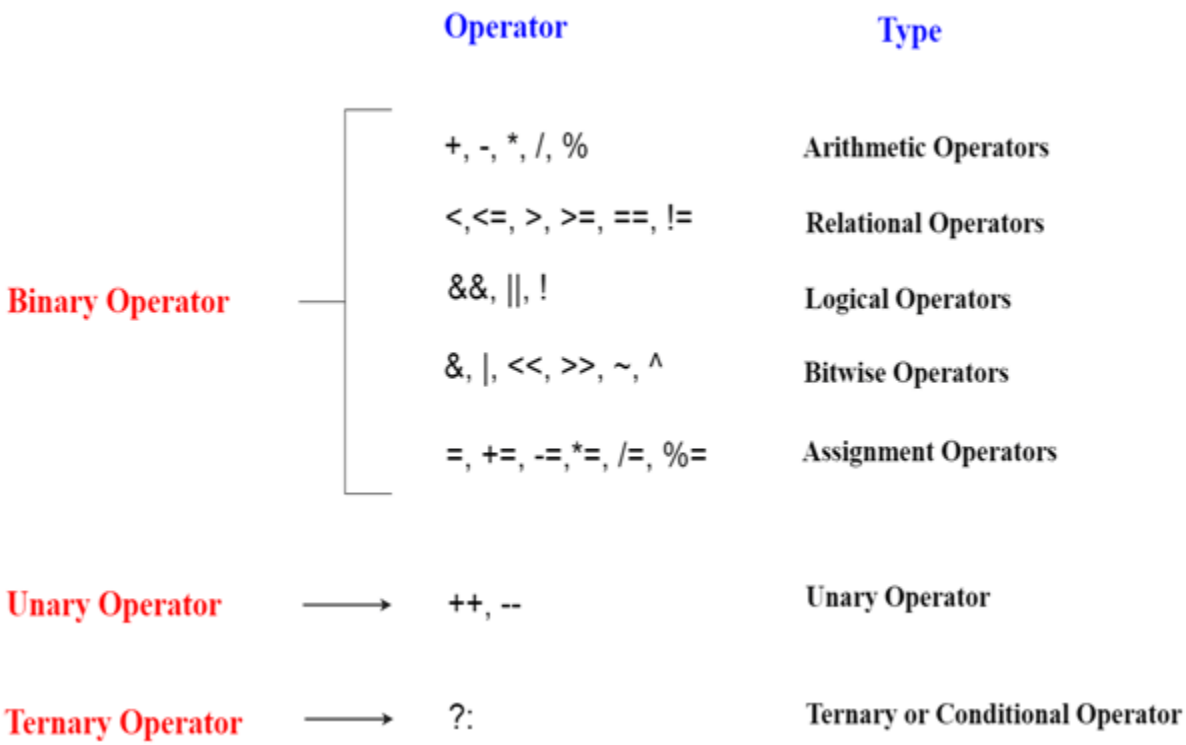
C++ Operators

An operator is simply a symbol that is used to perform operations. There can be many types of operations like arithmetic, logical, bitwise etc.

There are following types of operators to perform different types of operations in C language.

- Arithmetic Operators
- Relational Operators
- Logical Operators
- Bitwise Operators
- Assignment Operator
- Unary operator
- Ternary or Conditional Operator
- Misc Operator

DIWAKAR EDUCATION HUB



Precedence of Operators in C++

The precedence of operator species that which operator will be evaluated first and next. The associativity specifies the operators direction to be evaluated, it may be left to right or right to left.

the precedence by the example given below:

1. **int** data=5+10*10;

The "data" variable will contain 105 because * (multiplicative operator) is evaluated before + (additive operator).

The precedence and associativity of C++ operators is given below:

Category	Operator	Associativity

DIWAKAR EDUC

Postfix	() [] -> . ++ - -	Left to right
Unary	+ - ! ~ ++ - - (type)* & sizeof	Right to left
Multiplicative	* / %	Left to right
Additive	+ -	Right to left
Shift	<< >>	Left to right
Relational	< <= > >=	Left to right
Equality	== !=/td>	Right to left
Bitwise AND	&	Left to right
Bitwise XOR	^	Left to right
Bitwise OR	|	Right to left
Logical AND	&&	Left to right
Logical OR	||	Left to right
Conditional	?:	Right to left
Assignment	= += -= *= /= %=>>= <<= &= ^= |=	Right to left
Comma	,	Left to right

Decision making in C++ - if, else and else if

Decision making is about deciding the order of execution of statements based on certain conditions or repeat a group of statements until certain specified conditions are met. C++ handles decision-making by supporting the following statements,

- *if* statement
- *switch* statement
- conditional operator statement
- *goto* statement

Decision making with if statement

The if statement may be implemented in different forms depending on the complexity of conditions to be tested. The different forms are,

1. Simple *if* statement
2. *if....else* statement
3. Nested *if....else* statement
4. *else if* statement

Simple if statement

The general form of a simple *if* statement is,

```
if(expression)
{
   statement-inside;
}
   statement-outside;
```

If the **expression** is true, then 'statement-inside' will be executed, otherwise 'statement-inside' is skipped and only 'statement-outside' will be executed.

DIWAKAR EDUCATION HUB

Example:

```
#include< iostream.h>

int main( )

{

   int x,y;

   x=15;

   y=13;

   if (x > y )

   {

      cout << "x is greater than y";

   }

}
```

Output

```
x is greater than y

y is greater than x
```

Nested if....else statement

The general form of a nested *if...else* statement is,

```
if(expression)

{

   if(expression1)

   {
```

DIWAKAR EDUCATION HUB

```
    statement-block1;
  }
  else
  {
    statement-block2;
  }
}
else
{
  statement-block3;
}
```

if 'expression' is **false** or returns **false**, then the 'statement-block3' will be executed, otherwise execution will enter the if condition and check for 'expression 1'. Then if the 'expression 1' is **true** or returns **true**, then the 'statement-block1' will be executed otherwise 'statement-block2' will be executed.

Example:

```
void main( )
{
  int a,b,c;
  cout << "enter 3 number";
  cin >> a >> b >> c;
```

DIWAKAR EDUCATION HUB

```
if(a > b)
{
   if( a > c)
   {
      cout << "a is greatest";
   }
   else
   {
      cout << "c is greatest";
   }
}
else
{
   if( b> c)
   {
      cout << "b is greatest";
   }
   else
   {
      cout << "c is greatest";
   }
```

DIWAKAR EDUCATION HUB

```
   }
}
```

The above code will print different statements based on the values of a, b and c variables.

else-if Ladder

The general form of *else-if* ladder is,

```
if(expression 1)
{
   statement-block1;
}
else if(expression 2)
{
   statement-block2;
}
else if(expression 3 )
{
   statement-block3;
}
else
   default-statement;
```

The expression is tested from the top(of the ladder) downwards. As soon as the true condition is found, the statement associated with it is executed.

DIWAKAR EDUCATION HUB

Example:

```
void main( )
{
   int a;
   cout << "enter a number";
   cin >> a;
   if( a%5==0 && a%8==0)
   {
      cout << "divisible by both 5 and 8";
   }
   else if( a%8==0 )
   {
      cout << "divisible by 8";
   }
   else if(a%5==0)
   {
      cout << "divisible by 5";
   }
   else
   {
      cout << "divisible by none";
```

DIWAKAR EDUCATION HUB

```
   }

}
```

If We enter value **40** for the variable a, then the output will be:

Output

```
divisible by both 5 and 8
```

Points to Remember

In if statement, a single statement can be included without enclosing it into curly braces { }.

```
int a = 5;

if(a > 4)

cout << "success";
```

Output

```
success
```

No curly braces are required in the above case, but if we have more than one statement inside if condition, then we must enclose them inside curly braces otherwise only the first statement after the if condition will be considered.

```
int a = 2;

if(a > 4)

   cout << "success";

   // below statement is outside the if condition

   cout << "Not inside the if condition"
```

Output

DIWAKAR EDUCATION HUB

Not inside the if condition

1. == must be used for comparison in the expression of if condition, if you use = the expression will always return **true**, because it performs assignment not comparison.
2. Other than **0(zero)**, all other positive numeric values are considered as **true**.
3. if(27)
4. cout << "hello";\

Output

hello

C++ Functions

The function in C++ language is also known as procedure or subroutine in other programming languages.

To perform any task, we can create function. A function can be called many times. It provides modularity and code reusability.

Advantage of functions in C

There are many advantages of functions.

1) Code Reusability

By creating functions in C++, we can call it many times. So we don't need to write the same code again and again.

2) Code optimization

It makes the code optimized, we don't need to write much code.

DIWAKAR EDUCATION HUB

Suppose, we have to check 3 numbers (531, 883 and 781) whether it is prime number or not. Without using function, we need to write the prime number logic 3 times. So, there is repetition of code.

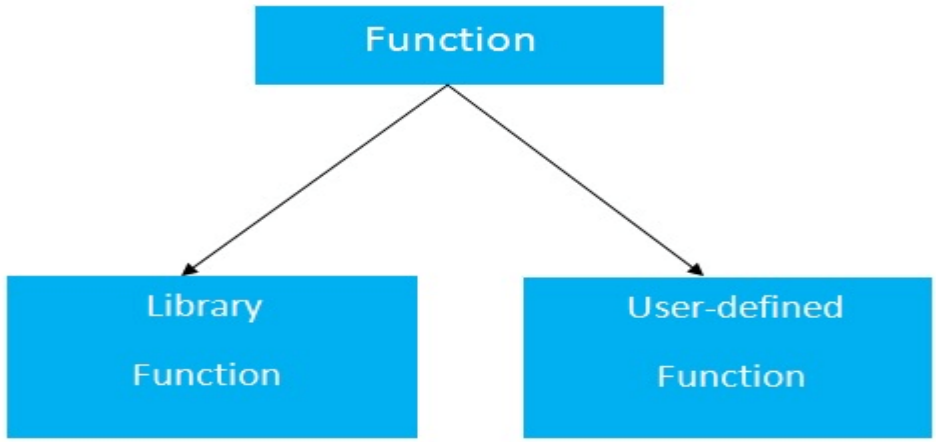
But if we use functions, we need to write the logic only once and we can reuse it several times.

Types of Functions

There are two types of functions in C programming:

1. Library Functions: are the functions which are declared in the C++ header files such as ceil(x), cos(x), exp(x), etc.

2. User-defined functions: are the functions which are created by the C++ programmer, so that he/she can use it many times. It reduces complexity of a big program and optimizes the code.



Declaration of a function

The syntax of creating function in C++ language is given below:

```
return_type function_name(data_type parameter...)
{
//code to be executed
```

```
}
```

C++ Function Example

the simple example of C++ function.

```
#include <iostream>
using namespace std;
void func() {
  static int i=0; //static variable
  int j=0; //local variable
  i++;
  j++;
  cout<<"i=" << i<<" and j=" <<j<<endl;
}
int main()
{
 func();
 func();
 func();
}
```

DIWAKAR EDUCATION HUB

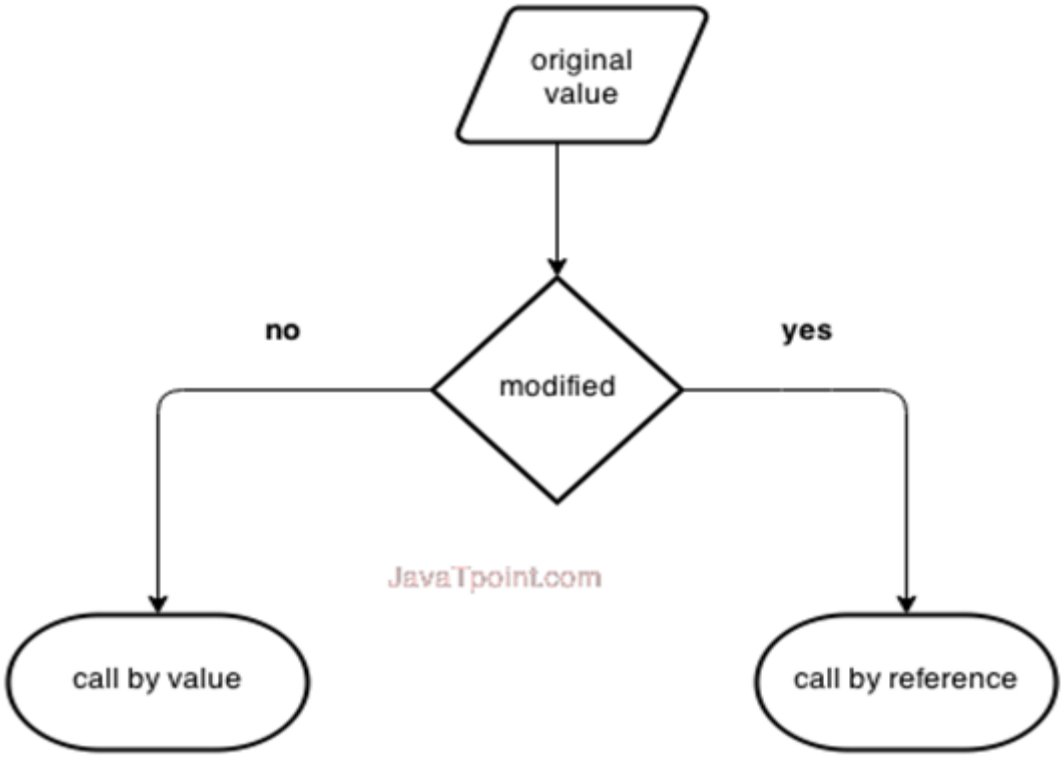
Output:

```
i= 1 and j= 1
i= 2 and j= 1
```

i= 3 and j= 1

Call by value and call by reference in C++

There are two ways to pass value or data to function in C language: call by value and call by reference. Original value is not modified in call by value but it is modified in call by reference.



Understand call by value and call by reference in C++ language one by one.

Call by value in C++

In call by value, **original value is not modified.**

In call by value, value being passed to the function is locally stored by the function parameter in stack memory location. If We change the value of function parameter, it is changed for the current function only. It will not change the value of variable inside the caller method such as main().

The concept of call by value in C++ language by the example given below:

```
#include <iostream>

using namespace std;

void change(int data);

int main()

{

int data = 3;

change(data);

cout << "Value of the data is: " << data<< endl;

return 0;

}

void change(int data)

{

data = 5;

}
```

Output:

```
Value of the data is: 3
```

Call by reference in C++

In call by reference, original value is modified because we pass reference (address).

Here, address of the value is passed in the function, so actual and formal arguments share the same address space. Hence, value changed inside the function, is reflected inside as well as outside the function.

DIWAKAR EDUCATION HUB

Note: To understand the call by reference, We must have the basic knowledge of pointers.

the concept of call by reference in C++ language by the example given below:

```
#include<iostream>
using namespace std;
void swap(int *x, int *y)
{
 int swap;
 swap=*x;
 *x=*y;
 *y=swap;
}
int main()
{
 int x=500, y=100;
 swap(&x, &y);  // passing value to function
 cout<<"Value of x is: "<<x<<endl;
 cout<<"Value of y is: "<<y<<endl;
 return 0;
}
```

Output:

DIWAKAR EDUCATION HUB

```
Value of x is: 100

Value of y is: 500
```

Difference between call by value and call by reference in C++

No.	Call by value	Call by reference
1	A copy of value is passed to the function	An address of value is passed to the function
2	Changes made inside the function is not reflected on other functions	Changes made inside the function is reflected outside the function also
3	Actual and formal arguments will be created in different memory location	Actual and formal arguments will be created in same memory location

DIWAKAR

C++ virtual function

- A C++ virtual function is a member function in the base class that We redefine in a derived class. It is declared using the virtual keyword.
- It is used to tell the compiler to perform dynamic linkage or late binding on the function.
- There is a necessity to use the single pointer to refer to all the objects of the different classes. So, we create the pointer to the base class that refers to all the derived objects. But, when base class pointer contains the address of the derived class object, always executes the base class function. This issue can only be resolved by using the 'virtual' function.

- A 'virtual' is a keyword preceding the normal declaration of a function.
- When the function is made virtual, C++ determines which function is to be invoked at the runtime based on the type of the object pointed by the base class pointer.

Late binding or Dynamic linkage

In late binding function call is resolved during runtime. Therefore compiler determines the type of object at runtime, and then binds the function call.

Rules of Virtual Function

- Virtual functions must be members of some class.
- Virtual functions cannot be static members.
- They are accessed through object pointers.
- They can be a friend of another class.
- A virtual function must be defined in the base class, even though it is not used.
- The prototypes of a virtual function of the base class and all the derived classes must be identical. If the two functions with the same name but different prototypes, C++ will consider them as the overloaded functions.
- We cannot have a virtual constructor, but we can have a virtual destructor.
- Consider the situation when we don't use the virtual keyword.

```
#include <iostream>

using namespace std;

class A

{

  int x=5;

   public:
```

DIWAKAR EDUCATION HUB

```
    void display()
    {
        std::cout << "Value of x is : " << x<<std::endl;
    }
};
class B: public A
{
    int y = 10;
    public:
    void display()
    {
        std::cout << "Value of y is : " <<y<< std::endl;
    }
};
int main()
{
    A *a;
    B b;
    a = &b;
   a->display();
    return 0;
```

DIWAKAR EDUCATION HUB

```
}
```

Output:

Value of x is : 5

In the above example, * a is the base class pointer. The pointer can only access the base class members but not the members of the derived class. Although C++ permits the base pointer to point to any object derived from the base class, it cannot directly access the members of the derived class. Therefore, there is a need for virtual function which allows the base pointer to access the members of the derived class.

C++ virtual function Example

the simple example of C++ virtual function used to invoked the derived class in a program.

```
#include <iostream>
{
 public:
 virtual void display()
 {
  cout << "Base class is invoked"<<endl;
 }
};
class B:public A
{
 public:
```

DIWAKAR EDUCATION HUB

```
void display()
{
 cout << "Derived Class is invoked"<<endl;
}
};
int main()
{
 A* a;   //pointer of base class
 B b;    //object of derived class
 a = &b;
 a->display();   //Late Binding occurs
}
```

Output:

```
Derived Class is invoked
```

Pure Virtual Function

- A virtual function is not used for performing any task. It only serves as a placeholder.
- When the function has no definition, such function is known as "**do-nothing**" function.
- The "**do-nothing**" function is known as a **pure virtual function**. A pure virtual function is a function declared in the base class that has no definition relative to the base class.
- A class containing the pure virtual function cannot be used to declare the objects of its own, such classes are known as abstract base classes.

DIWAKAR EDUCATION HUB

- The main objective of the base class is to provide the traits to the derived classes and to create the base pointer used for achieving the runtime polymorphism.

Pure virtual function can be defined as:

1. **virtual void** display() = 0;

A simple example:

```
#include <iostream>

using namespace std;

class Base

{

   public:

   virtual void show() = 0;

};

class Derived : public Base

{

   public:

   void show()

   {

      std::cout << "Derived class is derived from the base class." << std::endl;

   }

};

int main()
```

DIWAKAR EDUCATION HUB

```
{
   Base *bptr;
   //Base b;
   Derived d;
   bptr = &d;
   bptr->show();
   return 0;
}
```

Output:

Derived class is derived from the base class.

In the above example, the base class contains the pure virtual function. Therefore, the base class is an abstract base class. We cannot create the object of the base class.

C++ Inheritance

In C++, inheritance is a process in which one object acquires all the properties and behaviors of its parent object automatically. In such way, we can reuse, extend or modify the attributes and behaviors which are defined in other class.

In C++, the class which inherits the members of another class is called derived class and the class whose members are inherited is called base class. The derived class is the specialized class for the base class.

Advantage of C++ Inheritance

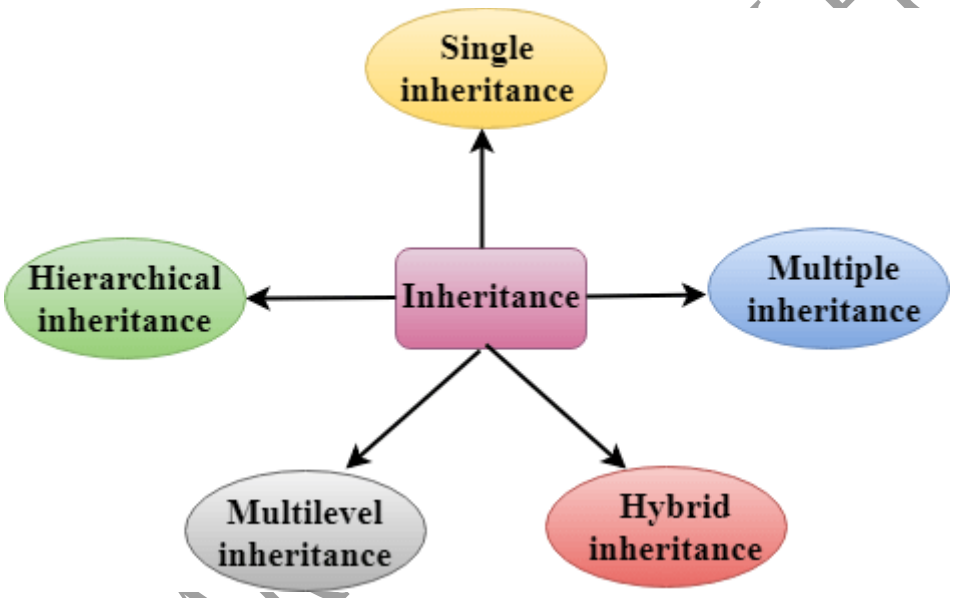
Code reusability: Now we can reuse the members of our parent class. So, there is no need to define the member again. So less code is required in the class.

DIWAKAR EDUCATION HUB

Types Of Inheritance

C++ supports five types of inheritance:

- Single inheritance
- Multiple inheritance
- Hierarchical inheritance
- Multilevel inheritance
- Hybrid inheritance



Derived Classes

A Derived class is defined as the class derived from the base class.

The Syntax of Derived class:

```
class derived_class_name :: visibility-mode base_class_name
{
   // body of the derived class.
}
```

Where,

derived_class_name: It is the name of the derived class.

visibility mode: The visibility mode specifies whether the features of the base class are publicly inherited or privately inherited. It can be public or private.

base_class_name: It is the name of the base class.

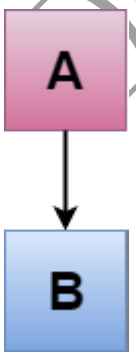
- When the base class is privately inherited by the derived class, public members of the base class becomes the private members of the derived class. Therefore, the public members of the base class are not accessible by the objects of the derived class only by the member functions of the derived class.
- When the base class is publicly inherited by the derived class, public members of the base class also become the public members of the derived class. Therefore, the public members of the base class are accessible by the objects of the derived class as well as by the member functions of the base class.

Note:

- In C++, the default mode of visibility is private.
- The private members of the base class are never inherited.

C++ Single Inheritance

Single inheritance is defined as the inheritance in which a derived class is inherited from the only one base class.



DIWAKAR EDUCATION HUB

Where 'A' is the base class, and 'B' is the derived class.

C++ Single Level Inheritance Example: Inheriting Fields

When one class inherits another class, it is known as single level inheritance. the example of single level inheritance which inherits the fields only.

```
#include <iostream>

using namespace std;

 class Account {

   public:

   float salary = 60000;

 };

   class Programmer: public Account {

   public:

   float bonus = 5000;

   };

int main(void) {

    Programmer p1;

    cout<<"Salary: "<<p1.salary<<endl;

    cout<<"Bonus: "<<p1.bonus<<endl;

    return 0;

}
```

DIWAKAR EDUCATION HUB

Output:

```
Salary: 60000

Bonus: 5000
```

In the above example, Employee is the **base** class and Programmer is the **derived** class.

C++ Single Level Inheritance Example: Inheriting Methods

another example of inheritance in C++ which inherits methods only.

```
#include <iostream>

using namespace std;

 class Animal {

   public:

 void eat() {

    cout<<"Eating..."<<endl;

 }

   };

   class Dog: public Animal

   {

     public:

    void bark(){

    cout<<"Barking...";

     }

   };
```

DIWAKAR EDUCATION HUB

```
int main(void) {
   Dog d1;
   d1.eat();
   d1.bark();
   return 0;
}
```

Output:

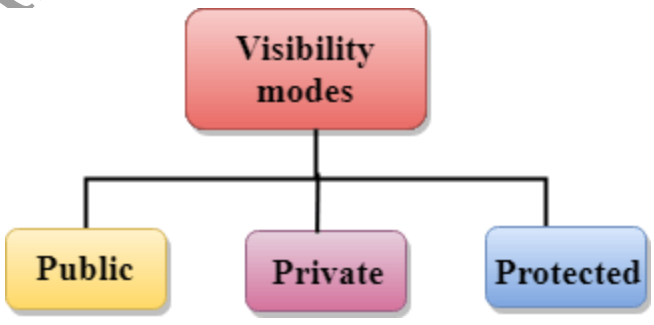
```
Eating...
Barking...
```

How to make a Private Member Inheritable

The private member is not inheritable. If we modify the visibility mode by making it public, but this takes away the advantage of data hiding.

C++ introduces a third visibility modifier, i.e., **protected**. The member which is declared as protected will be accessible to all the member functions within the class as well as the class immediately derived from it.

Visibility modes can be classified into three categories:



- **Public**: When the member is declared as public, it is accessible to all the functions of the program.

DIWAKAR EDUCATION HUB

- **Private**: When the member is declared as private, it is accessible within the class only.
- **Protected**: When the member is declared as protected, it is accessible within its own class as well as the class immediately derived from it.

Visibility of Inherited Members

Base class visibility	Derived class visibility		
	Public	Private	Protected
Private	Not Inherited	Not Inherited	Not Inherited
Protected	Protected	Private	Protected
Public	Public	Private	Protected

C++ Multilevel Inheritance

Multilevel inheritance is a process of deriving a class from another derived class.

DIWAKAR



C++ Multi Level Inheritance Example

When one class inherits another class which is further inherited by another class, it is known as multi level inheritance in C++. Inheritance is transitive so the last derived class acquires all the members of all its base classes.

the example of multi level inheritance in C++.

```
#include <iostream>
using namespace std;
 class Animal {
   public:
 void eat() {
    cout<<"Eating..."<<endl;
 }
   };
   class Dog: public Animal
```

DIWAKAR EDUCATION HUB

```
  {
     public:
    void bark(){
   cout<<"Barking..."<<endl;
    }
  };
  class BabyDog: public Dog
  {
     public:
    void weep() {
   cout<<"Weeping...";
    }
  };
int main(void) {
   BabyDog d1;
   d1.eat();
   d1.bark();
   d1.weep();
    return 0;
}
```

Output:

DIWAKAR EDUCATION HUB

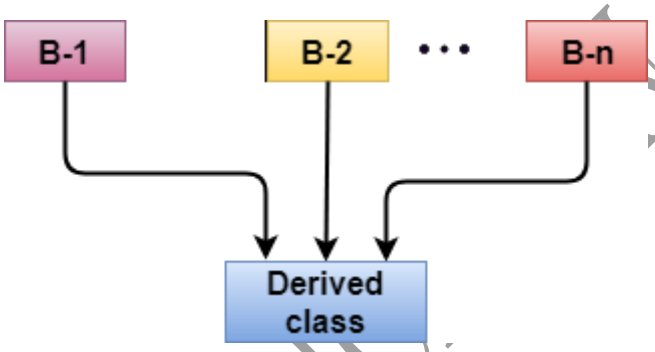
```
Eating...

Barking...

Weeping...
```

C++ Multiple Inheritance

Multiple inheritance is the process of deriving a new class that inherits the attributes from two or more classes.



Syntax of the Derived class:

```
class D : visibility B-1, visibility B-2, ?
{
   // Body of the class;
}
```

Ambiguity Resolution in Inheritance

Ambiguity can be occurred in using the multiple inheritance when a function with the same name occurs in more than one base class.

- The above issue can be resolved by using the class resolution operator with the function. In the above example, the derived class code can be rewritten as:

```
class C : public A, public B
```

DIWAKAR EDUCATION HUB

```
{
   void view()
   {
      A :: display();        // Calling the display() function of class A.
      B :: display();        // Calling the display() function of class B.

   }
};
```

An ambiguity can also occur in single inheritance.

Consider the following situation:

```
class A
{
  public:
void display()
{
  cout<<?Class A?;
}
};
class B
{
 public:
```

DIWAKAR EDUCATION HUB

```
 void display()
{
 cout<<?Class B?;
}
} ;
```

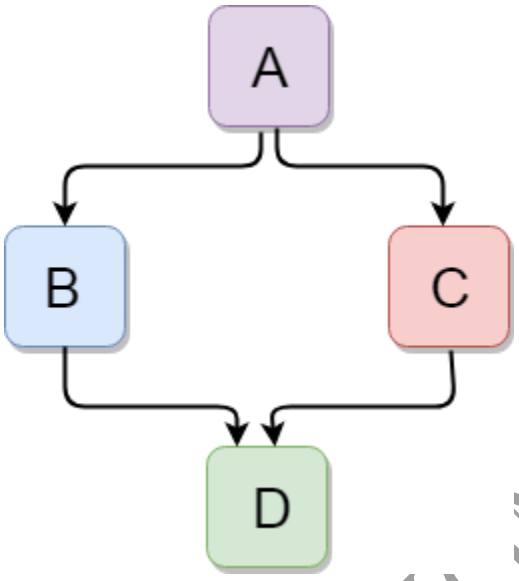
In the above case, the function of the derived class overrides the method of the base class. Therefore, call to the display() function will simply call the function defined in the derived class. If we want to invoke the base class function, we can use the class resolution operator.

```
int main()
{
   B b;
  b.display();              // Calling the display() function of B class.
  b.B :: display();       // Calling the display() function defined in B class.
}
```

C++ Hybrid Inheritance

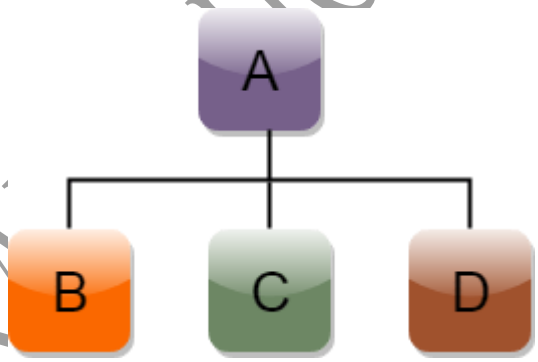
Hybrid inheritance is a combination of more than one type of inheritance.

DIWAKAR EDUCATION HUB



C++ Hierarchical Inheritance

Hierarchical inheritance is defined as the process of deriving more than one class from a base class.



Syntax of Hierarchical inheritance:

```
class A
{
   // body of the class A.
}
class B : public A
```

DIWAKAR EDUCATION HUB

```
{
   // body of class B.
}
class C : public A
{
   // body of class C.
}
class D : public A
{
   // body of class D.
}
```

C++ Object and Class

Since C++ is an object-oriented language, program is designed using objects and classes in C++.

C++ Object

In C++, Object is a real world entity, for example, chair, car, pen, mobile, laptop etc.

In other words, object is an entity that has state and behavior. Here, state means data and behavior means functionality.

Object is a runtime entity, it is created at runtime.

Object is an instance of a class. All the members of the class can be accessed through object.

DIWAKAR EDUCATION HUB

An example to create object of student class using s1 as the reference variable.

```
Student s1;  //creating an object of Student
```

In this example, Student is the type and s1 is the reference variable that refers to the instance of Student class.

C++ Class

In C++, object is a group of similar objects. It is a template from which objects are created. It can have fields, methods, constructors etc.

An example of C++ class that has three fields only.

```
class Student
{
   public:
   int id;  //field or data member
   float salary; //field or data member
   String name;//field or data member
}
```

C++ Constructor

In C++, constructor is a special method which is invoked automatically at the time of object creation. It is used to initialize the data members of new object generally. The constructor in C++ has the same name as class or structure.

There can be two types of constructors in C++.

- Default constructor
- Parameterized constructor

C++ Default Constructor

DIWAKAR EDUCATION HUB

A constructor which has no argument is known as default constructor. It is invoked at the time of creating object.

the simple example of C++ default Constructor.

```
#include <iostream>
using namespace std;
class Employee
 {
   public:
       Employee()
       {
          cout<<"Default Constructor Invoked"<<endl;
       }
};
int main(void)
{
   Employee e1; //creating an object of Employee
   Employee e2;
   return 0;
}
```

Output:

```
Default Constructor Invoked
```

DIWAKAR EDUCATION HUB

Default Constructor Invoked

C++ Parameterized Constructor

A constructor which has parameters is called parameterized constructor. It is used to provide different values to distinct objects.

C++ Destructor

A destructor works opposite to constructor; it destructs the objects of classes. It can be defined only once in a class. Like constructors, it is invoked automatically.

A destructor is defined like constructor. It must have same name as class. But it is prefixed with a tilde sign (~).

Note: C++ destructor cannot have parameters. Moreover, modifiers can't be applied on destructors.

C++ Constructor and Destructor Example

An example of constructor and destructor in C++ which is called automatically

```
#include <iostream>
using namespace std;
class Employee
 {
   public:
     Employee()
     {
        cout<<"Constructor Invoked"<<endl;
     }
```

DIWAKAR EDUCATION HUB

```
    ~Employee()
    {
       cout<<"Destructor Invoked"<<endl;
    }
};
int main(void)
{
   Employee e1; //creating an object of Employee
   Employee e2; //creating an object of Employee
   return 0;
}
```

Output:

```
Constructor Invoked
Constructor Invoked
Destructor Invoked
Destructor Invoked
```

C++ Overloading (Operator and Function)

C++ allows us to specify more than one definition for a **function** name or an **operator** in the same scope, which is called **function overloading** and **operator overloading** respectively.

An overloaded declaration is a declaration that is declared with the same name as a previously declared declaration in the same scope, except that both

DIWAKAR EDUCATION HUB

declarations have different arguments and obviously different definition (implementation).

When we call an overloaded **function** or **operator**, the compiler determines the most appropriate definition to use, by comparing the argument types We have used to call the function or operator with the parameter types specified in the definitions. The process of selecting the most appropriate overloaded function or operator is called **overload resolution**.

Function Overloading in C++

we can have multiple definitions for the same function name in the same scope. The definition of the function must differ from each other by the types and/or the number of arguments in the argument list. We cannot overload function declarations that differ only by return type.

Following is the example where same function **print()** is being used to print different data types −

```
#include <iostream>

using namespace std;

class printData {

  public:

    void print(int i) {

      cout << "Printing int: " << i << endl;

    }

    void print(double  f) {

      cout << "Printing float: " << f << endl;

    }
```

DIWAKAR EDUCATION HUB

```
    void print(char* c) {
      cout << "Printing character: " << c << endl;
    }
};
int main(void) {
  printData pd;
   // Call print to print integer
  pd.print(5);
  // Call print to print float
  pd.print(500.263);
  // Call print to print character
  pd.print("Hello C++");

  return 0;
}
```

When the above code is compiled and executed, it produces the following result −

Printing int: 5

Printing float: 500.263

Printing character: Hello C++

Operators Overloading in C++

DIWAKAR EDUCATION HUB

We can redefine or overload most of the built-in operators available in C++. Thus, a programmer can use operators with user-defined types as well.

Overloaded operators are functions with special names: the keyword "operator" followed by the symbol for the operator being defined. Like any other function, an overloaded operator has a return type and a parameter list.

Box operator+(const Box&);

declares the addition operator that can be used to **add** two Box objects and returns final Box object. Most overloaded operators may be defined as ordinary non-member functions or as class member functions. In case we define above function as non-member function of a class then we would have to pass two arguments for each operand as follows −

Box operator+(const Box&, const Box&);

Following is the list of operators which can be overloaded −

+	-	*	/	%	^
&	|	~	!	,	=
<	>	<=	>=	++	--
<<	>>	==	!=	&&	||
+=	-=	/=	%=	^=	&=
|=	*=	<<=	>>=	[]	()
->	->*	new	new []	delete	delete []

Following is the list of operators, which can not be overloaded −

DIWAKAR EDUCATION HUB

::	.*	.	?:

Operator Overloading Examples

Here are various operator overloading examples to help us in understanding the concept.

Sr.No	Operators & Example
1	Unary Operators Overloading
2	Binary Operators Overloading
3	Relational Operators Overloading
4	Input/Output Operators Overloading
5	++ and -- Operators Overloading
6	Assignment Operators Overloading
7	Function call () Operator Overloading
8	Subscripting [] Operator Overloading
9	Class Member Access Operator -> Overloading

DIWAKAR EDUCATION HUB

Templates in C++

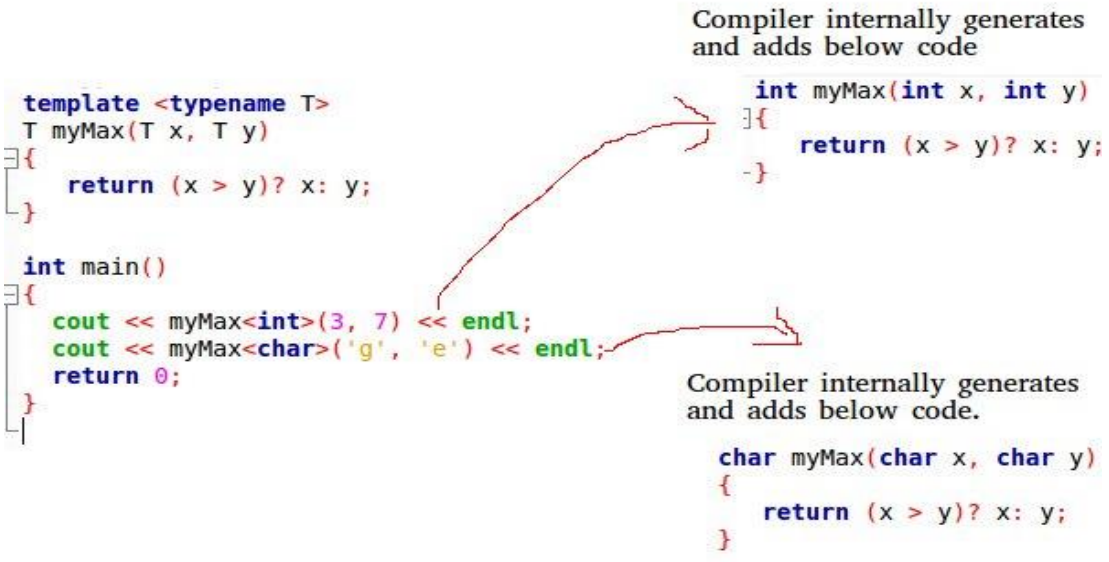
A template is a simple and yet very powerful tool in C++. The simple idea is to pass data type as a parameter so that we don't need to write the same code for different data types. For example, a software company may need sort() for

different data types. Rather than writing and maintaining the multiple codes, we can write one sort() and pass data type as a parameter.

C++ adds two new keywords to support templates: *'template'* and *'typename'*. The second keyword can always be replaced by keyword 'class'.

How templates work?

Templates are expanded at compiler time. This is like macros. The difference is, compiler does type checking before template expansion. The idea is simple, source code contains only function/class, but compiled code may contain multiple copies of same function/class.



Function Templates We write a generic function that can be used for different data types. Examples of function templates are sort(), max(), min(), printArray().

```
#include <iostream>

using namespace std;

// One function works for all data types.  This would work

// even for user defined types if operator '>' is overloaded

template <typename T>

T myMax(T x, T y)

{

  return (x > y)? x: y;

}

int main()

{

 cout << myMax<int>(3, 7) << endl;  // Call myMax for int

 cout << myMax<double>(3.0, 7.0) << endl; // call myMax for double

 cout << myMax<char>('g', 'e') << endl;   // call myMax for char

 return 0;

}
```

Output:

```
7

7

G
```

DIWAKAR EDUCATION HUB

Class Templates Like function templates, class templates are useful when a class defines something that is independent of the data type. Can be useful for classes like LinkedList, BinaryTree, Stack, Queue, Array, etc.

C++ Exception Handling

An exception is a problem that arises during the execution of a program. A C++ exception is a response to an exceptional circumstance that arises while a program is running, such as an attempt to divide by zero.

Exceptions provide a way to transfer control from one part of a program to another. C++ exception handling is built upon three keywords: **try, catch,** and **throw**.

- **throw** − A program throws an exception when a problem shows up. This is done using a **throw** keyword.
- **catch** − A program catches an exception with an exception handler at the place in a program where We want to handle the problem. The **catch** keyword indicates the catching of an exception.
- **try** − A **try** block identifies a block of code for which particular exceptions will be activated. It's followed by one or more catch blocks.

Assuming a block will raise an exception, a method catches an exception using a combination of the **try** and **catch** keywords. A try/catch block is placed around the code that might generate an exception. Code within a try/catch block is referred to as protected code, and the syntax for using try/catch as follows −

```
try {
  // protected code
} catch( ExceptionName e1 ) {
  // catch block
} catch( ExceptionName e2 ) {
```

DIWAKAR EDUCATION HUB

```
  // catch block
} catch( ExceptionName eN ) {
  // catch block
}
```

We can list down multiple **catch** statements to catch different type of exceptions in case our **try** block raises more than one exception in different situations.

Throwing Exceptions

Exceptions can be thrown anywhere within a code block using **throw** statement. The operand of the throw statement determines a type for the exception and can be any expression and the type of the result of the expression determines the type of exception thrown.

Following is an example of throwing an exception when dividing by zero condition occurs –

```
double division(int a, int b) {
  if( b == 0 ) {
     throw "Division by zero condition!";
  }
  return (a/b);
}
```

Catching Exceptions

The **catch** block following the **try** block catches any exception. We can specify what type of exception We want to catch and this is determined by the exception declaration that appears in parentheses following the keyword catch.

DIWAKAR EDUCATION HUB

```
try {
  // protected code
} catch( ExceptionName e ) {
  // code to handle ExceptionName exception
}
```

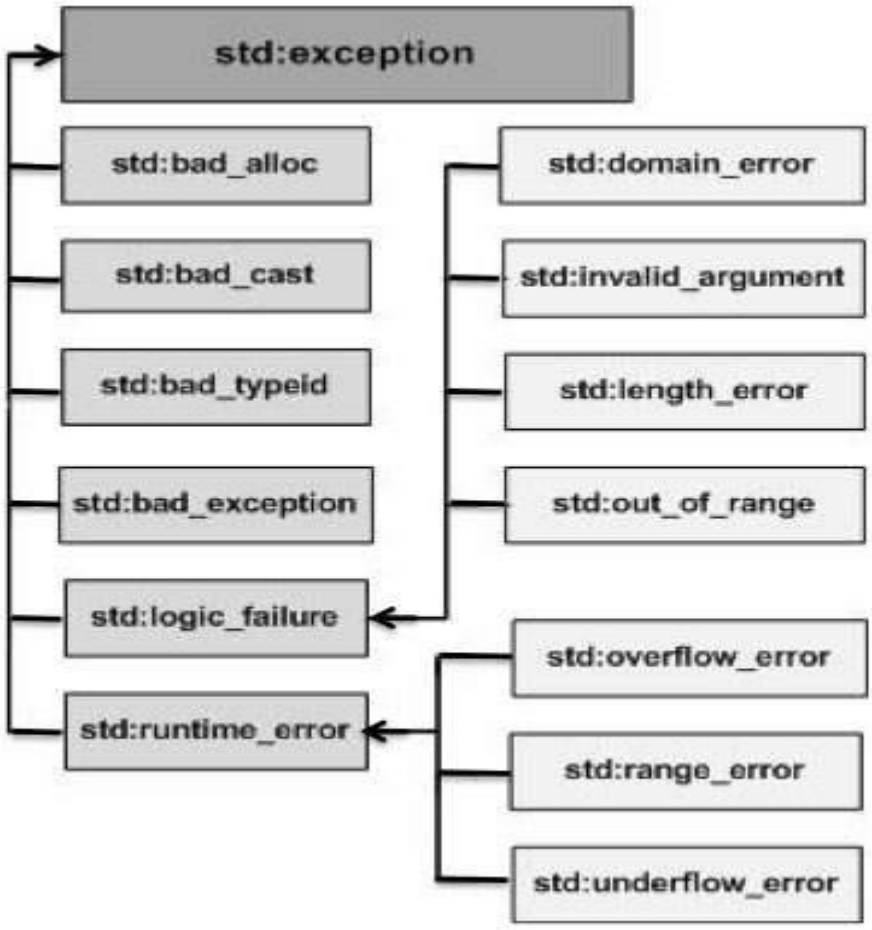
Above code will catch an exception of **ExceptionName** type. If We want to specify that a catch block should handle any type of exception that is thrown in a try block, We must put an ellipsis, ..., between the parentheses enclosing the exception declaration as follows −

```
try {
  // protected code
} catch(...) {
  // code to handle any exception
}
```

C++ Standard Exceptions

C++ provides a list of standard exceptions defined in **<exception>** which we can use in our programs. These are arranged in a parent-child class hierarchy shown below −

DIWAKAR EDUCATION HUB



Here is the small description of each exception mentioned in the above hierarchy –

Sr.No	Exception & Description
1	**std::exception** An exception and parent class of all the standard C++ exceptions.
2	**std::bad_alloc** This can be thrown by **new**.

3	**std::bad_cast** This can be thrown by **dynamic_cast**.
4	**std::bad_exception** This is useful device to handle unexpected exceptions in a C++ program.
5	**std::bad_typeid** This can be thrown by **typeid**.
6	**std::logic_error** An exception that theoretically can be detected by reading the code.
7	**std::domain_error** This is an exception thrown when a mathematically invalid domain is used.
8	**std::invalid_argument** This is thrown due to invalid arguments.
9	**std::length_error** This is thrown when a too big std::string is created.
10	**std::out_of_range** This can be thrown by the 'at' method, for example a std::vector and std::bitset<>::operator[]().

DIWAKAR EDUCATION HUB

11	**std::runtime_error** An exception that theoretically cannot be detected by reading the code.
12	**std::overflow_error** This is thrown if a mathematical overflow occurs.
13	**std::range_error** This is occurred when We try to store a value which is out of range.
14	**std::underflow_error** This is thrown if a mathematical underflow occurs.

C++ Files and Streams

another standard C++ library called **fstream**, which defines three new data types −

Sr.No	Data Type & Description
1	**ofstream** This data type represents the output file stream and is used to create files and to write information to files.
2	**ifstream** This data type represents the input file stream and is used to read information from files.

DIWAKAR EDUCATION HUB

3	**fstream** This data type represents the file stream generally, and has the capabilities of both ofstream and ifstream which means it can create files, write information to files, and read information from files.

To perform file processing in C++, header files <iostream> and <fstream> must be included in our C++ source file.

Opening a File

A file must be opened before We can read from it or write to it.
Either **ofstream** or **fstream** object may be used to open a file for writing. And ifstream object is used to open a file for reading purpose only.

Following is the standard syntax for open() function, which is a member of fstream, ifstream, and ofstream objects.

void open(const char *filename, ios::openmode mode);

Here, the first argument specifies the name and location of the file to be opened and the second argument of the **open()** member function defines the mode in which the file should be opened.

Sr.No	Mode Flag & Description
1	**ios::app** Append mode. All output to that file to be appended to the end.
2	**ios::ate** Open a file for output and move the read/write control to the end of the file.

DIWAKAR EDUCATION HUB

3	**ios::in** Open a file for reading.
4	**ios::out** Open a file for writing.
5	**ios::trunc** If the file already exists, its contents will be truncated before opening the file.

we can combine two or more of these values by **OR**ing them together. For example if We want to open a file in write mode and want to truncate it in case that already exists, following will be the syntax –

```
ofstream outfile;

outfile.open("file.dat", ios::out | ios::trunc );
```

Similar way, We can open a file for reading and writing purpose as follows –

```
fstream  afile;

afile.open("file.dat", ios::out | ios::in );
```

Closing a File

When a C++ program terminates it automatically flushes all the streams, release all the allocated memory and close all the opened files. But it is always a good practice that a programmer should close all the opened files before program termination.

Following is the standard syntax for close() function, which is a member of fstream, ifstream, and ofstream objects.

```
void close();
```

DIWAKAR EDUCATION HUB

Writing to a File

While doing C++ programming, We write information to a file from our program using the stream insertion operator (<<) just as We use that operator to output information to the screen. The only difference is that We use an **ofstream** or **fstream** object instead of the **cout** object.

Reading from a File

We read information from a file into our program using the stream extraction operator (>>) just as We use that operator to input information from the keyboard. The only difference is that We use an **ifstream** or **fstream** object instead of the **cin** object.

File Position Pointers

Both **istream** and **ostream** provide member functions for repositioning the file-position pointer. These member functions are **seekg** ("seek get") for istream and **seekp** ("seek put") for ostream.

The argument to seekg and seekp normally is a long integer. A second argument can be specified to indicate the seek direction. The seek direction can be **ios::beg** (the default) for positioning relative to the beginning of a stream, **ios::cur** for positioning relative to the current position in a stream or **ios::end** for positioning relative to the end of a stream.

The file-position pointer is an integer value that specifies the location in the file as a number of bytes from the file's starting location. Some examples of positioning the "get" file-position pointer are −

// position to the nth byte of fileObject (assumes ios::beg)

fileObject.seekg(n);

// position n bytes forward in fileObject

DWAKAR EDUCATION HUB

```
fileObject.seekg( n, ios::cur );


// position n bytes back from end of fileObject
fileObject.seekg( n, ios::end );


// position at end of fileObject
fileObject.seekg( 0, ios::end );
```

C++ Exceptions

C++ Exceptions

When executing C++ code, different errors can occur: coding errors made by the programmer, errors due to wrong input, or other unforeseeable things.

When an error occurs, C++ will normally stop and generate an error message. The technical term for this is: C++ will throw an **exception** (throw an error).

C++ try and catch

Exception handling in C++ consist of three keywords: try, throw and catch:

The try statement allows We to define a block of code to be tested for errors while it is being executed.

The throw keyword throws an exception when a problem is detected, which lets us create a custom error.

The catch statement allows We to define a block of code to be executed, if an error occurs in the try block.

The try and catch keywords come in pairs:

Example

DIWAKAR EDUCATION HUB

```
try {
  // Block of code to try
  throw exception; // Throw an exception when a problem arise
}
catch () {
  // Block of code to handle errors
}
```

Web Programming

Web programming refers to the writing, markup and coding involved in Web development, which includes Web content, Web client and server scripting and network security. The most common languages used for Web programming are XML, HTML, JavaScript, Perl 5 and PHP. Web programming is different from just programming, which requires interdisciplinary knowledge on the application area, client and server scripting, and database technology.

Web programming can be briefly categorized into client and server coding. The client side needs programming related to accessing data from users and providing information. It also needs to ensure there are enough plug ins to enrich user experience in a graphic user interface, including security measures.

1. To improve user experience and related functionalities on the client side, JavaScript is usually used. It is an excellent client-side platform for designing and implementing Web applications.
2. HTML5 and CSS3 supports most of the client-side functionality provided by other application frameworks.

The server side needs programming mostly related to data retrieval, security and performance. Some of the tools used here include ASP, Lotus Notes, PHP, Java and MySQL. There are certain tools/platforms that aid in both client- and server-side programming. Some examples of these are Opa and Tersus.

Hypertext Markup Language (HTML)

DIWAKAR EDUCATION HUB

Hypertext markup language (HTML) is the major markup language used to display Web pages on the Internet. In other words, Web pages are composed of HTML, which is used to display text, images or other resources through a Web browser.

All HTML is plain text, meaning it is not compiled and may be read by humans. The file extension for an HTML file is .htm or .html.

New Web developers may mistake HTML for a programming language when it is actually a markup language. HTML is used with other technologies because all HTML really does is organize documents. On the client side, JavaScript (JS) is used to provide interactivity. On the server side, a Web development platform like Ruby, PHP or ASP.NET is used.

When a Web developer builds an application, the work is performed on the server, and raw HTML is sent out to the user. The line between server-side development and client side development is blurry with technologies like AJAX.

HTML was never designed for the Web that exists today, as it is just a markup language with severe limitations, in terms of control and design. Numerous technologies have been used to work around this issue - the most significant being cascading style sheet (CSS).

The long term solution is (or hopefully will be) HTML5, which is the next generation of HTML and allows for more control and interactivity. As with any development on the Web, the move to standards is a slow and arduous process, and Web developers and designers have to make due with current and supported technologies, which means that basic HTML will continue to be used for some time.

Applications of HTML

As mentioned before, HTML is one of the most widely used language over the web. I'm going to list few of them here:

DIWAKAR EDUCATION HUB

- **Web pages development** - HTML is used to create pages which are rendered over the web. Almost every page of web is having html tags in it to render its details in browser.
- **Internet Navigation** - HTML provides tags which are used to navigate from one page to another and is heavily used in internet navigation.
- **Responsive UI** - HTML pages now-a-days works well on all platform, mobile, tabs, desktop or laptops owing to responsive design strategy.
- **Offline support** HTML pages once loaded can be made available offline on the machine without any need of internet.
- **Game development**- HTML5 has native support for rich experience and is now useful in gaming developent arena as well.

Basic HTML Document

In its simplest form, following is an example of an HTML document –

```
<!DOCTYPE html>
<html>
  <head>
    <title>This is document title</title>
  </head>
  <body>
    <h1>This is a heading</h1>
    <p>Document content goes here.....</p>
  </body>
</html>
```

DIWAKAR EDUCATION HUB

HTML Tags

As told earlier, HTML is a markup language and makes use of various tags to format the content. These tags are enclosed within angle braces **<Tag Name>**. Except few tags, most of the tags have their corresponding closing tags. For example, **<html>** has its closing tag **</html>** and **<body>** tag has its closing tag **</body>** tag etc.

Above example of HTML document uses the following tags –

Sr.No	Tag & Description
1	**<!DOCTYPE...>** This tag defines the document type and HTML version.
2	**<html>** This tag encloses the complete HTML document and mainly comprises of document header which is represented by <head>...</head> and document body which is represented by <body>...</body> tags.
3	**<head>** This tag represents the document's header which can keep other HTML tags like <title>, <link> etc.
4	**<title>** The <title> tag is used inside the <head> tag to mention the document title.
5	**<body>** This tag represents the document's body which keeps other HTML tags

DIWAKAR EDUCATION HUB

	like <h1>, , <p> etc.
6	**<h1>** This tag represents the heading.
7	**<p>** This tag represents a paragraph.

To learn HTML, we will need to study various tags and understand how they behave, while formatting a textual document. Learning HTML is simple as users have to learn the usage of different tags in order to format the text or images to make a beautiful webpage.

World Wide Web Consortium (W3C) recommends to use lowercase tags starting from HTML 4.

HTML Document Structure

A typical HTML document will have the following structure –

```
<html>
  <head>
    Document header related tags
  </head>
  <body>
    Document body related tags
  </body>
</html>
```

DIWAKAR EDUCATION HUB

The <!DOCTYPE> Declaration

The <!DOCTYPE> declaration tag is used by the web browser to understand the version of the HTML used in the document. Current version of HTML is 5 and it makes use of the following declaration −

<!DOCTYPE html>

There are many other declaration types which can be used in HTML document depending on what version of HTML is being used.

Extensible Markup Language (XML)

Extensible Markup Language (XML) is a universal format, maintained by the W3C, used for representation and transfer of structured data on the web or between different applications.

The language uses a structured representation by allowing users to create custom defined tags according to XML Document Type Definition (DTD) standards. The structure of an XML document can be represented in the form of a tree known as a Document Object Model (DOM).

XML is designed to describe data but is not concerned with the data's visualization. The tags created in XML are self explanatory and the user is free to define their own tags — hence the "extensible."

XML support is provided by many programming language platforms to create and process XML data. Simplicity, portability, platform independence, and usability are some of the key features that have resulted in the increasing popularity of the use of XML-based standards.

XML has been the main source of motivation for development of service oriented architecture (SOA) platforms such as web services that are not tied to any particular language and collaborate effectively in heterogeneous environments. RSS, Atom, SOAP and XHTML are some of the standards influenced by XML.

DIWAKAR EDUCATION HUB

JavaScript (JS)

Javascript (JS) is a scripting languages, primarily used on the Web. It is used to enhance HTML pages and is commonly found embedded in HTML code. JavaScript is an interpreted language. Thus, it doesn't need to be compiled. JavaScript renders web pages in an interactive and dynamic fashion. This allowing the pages to react to events, exhibit special effects, accept variable text, validate data, create cookies, detect a user's browser, etc.

HTML pages are fine for displaying static content, e.g. a simple image or text. However, most pages nowadays are rarely static. Many of today's pages have menus, forms, slideshows and even images that provide user interaction. Javascript is the language employed by web developers to provide such interaction. Since JavaScript works with HTML pages, a developer needs to know HTML to harness this scripting language's full potential. While there are other languages that can be used for scripting on the Web, in practice it is essentially all Javascript.

There are two ways to use JavaScript in an HTML file. The first one involves embedding all the JavaScript code in the HTML code, while the second method makes use of a separate JavaScript file that's called from within a Script element, i.e., enclosed by Script tags. JavaScript files are identified by the .js extension. Although JavaScript is mostly used to interact with HTML objects, it can also be made to interact with other non-HTML objects such as browser plugins, CSS (Cascading Style Sheets) properties, the current date, or the browser itself. To write JavaScript code, all We need is a basic text editor like Notepad in Windows, Gimp in Linux, or BBEdit. Some text editors, like BBEdit feature syntax highlighting for JavaScript. This will allow We easily identify elements of JavaScript code. The latest versions of Internet Explorer, Firefox, and Opera all support JavaScript.

Java

Java is a programming language that produces software for multiple platforms. When a programmer writes a Java application, the compiled code (known as

DIWAKAR EDUCATION HU

bytecode) runs on most operating systems (OS), including Windows, Linux and Mac OS. Java derives much of its syntax from the C and C++ programming languages.

Java was developed in the mid-1990s by James A. Gosling, a former computer scientist with Sun Microsystems.

Java produces applets (browser-run programs), which facilitate graphical user interface (GUI) and object interaction by Internet users. Prior to Java applets, Web pages were typically static and non-interactive. Java applets have diminished in popularity with the release of competing products, such as Adobe Flash and Microsoft Silverlight.

Java applets run in a Web browser with Java Virtual Machine (JVM), which translates Java bytecode into native processor instructions and allows indirect OS or platform program execution. JVM provides the majority of components needed to run bytecode, which is usually smaller than executable programs written through other programming languages. Bytecode cannot run if a system lacks required JVM.

Java program development requires a Java software development kit (SDK) that typically includes a compiler, interpreter, documentation generator and other tools used to produce a complete application.

Development time may be accelerated through the use of integrated development environments (IDE) - such as JBuilder, Netbeans, Eclipse or JCreator. IDEs facilitate the development of GUIs, which include buttons, text boxes, panels, frames, scrollbars and other objects via drag-and-drop and point-and-click actions.

Java programs are found in desktops, servers, mobile devices, smart cards and Blu-ray Discs (BD).

DIWAKAR EDUCATION HUB

Java is –

- **Object Oriented** − In Java, everything is an Object. Java can be easily extended since it is based on the Object model.
- **Platform Independent** − Unlike many other programming languages including C and C++, when Java is compiled, it is not compiled into platform specific machine, rather into platform independent byte code. This byte code is distributed over the web and interpreted by the Virtual Machine (JVM) on whichever platform it is being run on.
- **Simple** − Java is designed to be easy to learn. If We understand the basic concept of OOP Java, it would be easy to master.
- **Secure** − With Java's secure feature it enables to develop virus-free, tamper-free systems. Authentication techniques are based on public-key encryption.
- **Architecture-neutral** − Java compiler generates an architecture-neutral object file format, which makes the compiled code executable on many processors, with the presence of Java runtime system.
- **Portable** − Being architecture-neutral and having no implementation dependent aspects of the specification makes Java portable. Compiler in Java is written in ANSI C with a clean portability boundary, which is a POSIX subset.
- **Robust** − Java makes an effort to eliminate error prone situations by emphasizing mainly on compile time error checking and runtime checking.
- **Multithreaded** − With Java's multithreaded feature it is possible to write programs that can perform many tasks simultaneously. This design feature allows the developers to construct interactive applications that can run smoothly.
- **Interpreted** − Java byte code is translated on the fly to native machine instructions and is not stored anywhere. The development process is more rapid and analytical since the linking is an incremental and light-weight process.

DIWAKAR EDUCATION HUB

- **High Performance** – With the use of Just-In-Time compilers, Java enables high performance.
- **Distributed** – Java is designed for the distributed environment of the internet.
- **Dynamic** – Java is considered to be more dynamic than C or C++ since it is designed to adapt to an evolving environment. Java programs can carry extensive amount of run-time information that can be used to verify and resolve accesses to objects on run-time.

Dynamic HyperText Markup Language (DHTML)

Dynamic HyerText Markup Language (DHTML) is a combination of Web development technologies used to create dynamically changing websites. Web pages may include animation, dynamic menus and text effects. The technologies used include a combination of HTML, JavaScript or VB Script,
CSS and the document object model (DOM).

Designed to enhance a Web user's experience, DHTML includes the following features:

- Dynamic content, which allows the user to dynamically change Web page content
- Dynamic positioning of Web page elements
- Dynamic style, which allows the user to change the Web page's color, font, size or content

While DHTML enhances the website user's experience, the technology may also be frustrating for users when it is used incorrectly. For example, a website menu with flashy DHTML animations can easily confuse user navigation. Another DHTML issue occurs when Web developers attempt to create cross-browser DHTML, which is very difficult.

For Web developers, DHTML poses the following problems:

DIWAKAR EDUCATION HUB

- It can be difficult to develop and debug because of lack of Web browser and technological support.
- DHTML scripts may not work correctly in various Web browsers.
- The Web page layout may not display correctly when it is developed to display in different screen size combinations and in different browsers.

As a result of these problems, Web developers must determine whether DHTML enhances the user experience in any given context. Most Web developers abandon complex DHTML and use simple cross-browser routines to improve user experience, as opposed to integrating excessive DHTML visual effects.

Components of Dynamic HTML

DHTML consists of the following four components or languages:

- HTML 4.0
- CSS
- JavaScript
- DOM.

HTML 4.0

HTML is a client-side markup language, which is a core component of the DHTML. It defines the structure of a web page with various defined basic elements or tags.

CSS

CSS stands for Cascading Style Sheet, which allows the web users or developers for controlling the style and layout of the HTML elements on the web pages.

JavaScript

JavaScript is a scripting language which is done on a client-side. The various browser supports JavaScript technology. DHTML uses the JavaScript technology for accessing, controlling, and manipulating the HTML elements. The statements in JavaScript are the commands which tell the browser for performing an action.

DIWAKAR EDUCATION HUB

DOM

DOM is the document object model. It is a w3c standard, which is a standard interface of programming for HTML. It is mainly used for defining the objects and properties of all elements in HTML.

Uses of DHTML

Following are the uses of DHTML (Dynamic HTML):

- It is used for designing the animated and interactive web pages that are developed in real-time.
- DHTML helps users by animating the text and images in their documents.
- It allows the authors for adding the effects on their pages.
- It also allows the page authors for including the drop-down menus or rollover buttons.
- This term is also used to create various browser-based action games.
- It is also used to add the ticker on various websites, which needs to refresh their content automatically.

Features of DHTML

Following are the various characteristics or features of DHTML (Dynamic HTML):

- Its simplest and main feature is that we can create the web page dynamically.
- **Dynamic Style** is a feature, that allows the users to alter the font, size, color, and content of a web page.
- It provides the facility for using the events, methods, and properties. And, also provides the feature of code reusability.
- It also provides the feature in browsers for data binding.

DIWAKAR EDUCATION HUB

- Using DHTML, users can easily create dynamic fonts for their web sites or web pages.
- With the help of DHTML, users can easily change the tags and their properties.
- The web page functionality is enhanced because the DHTML uses low-bandwidth effect.

Difference between HTML and DHTML

Following table describes the differences between HTML and DHTML:

HTML (Hypertext Markup language)	DHTML (Dynamic Hypertext Markup language)
1. HTML is simply a markup language.	1. DHTML is not a language, but it is a set of technologies of web development.
2. It is used for developing and creating web pages.	2. It is used for creating and designing the animated and interactive web sites or pages.
3. This markup language creates static web pages.	3. This concept creates dynamic web pages.
4. It does not contain any server-side scripting code.	4. It may contain the code of server-side scripting.
5. The files of HTML are stored with the .html or .htm extension in a	5. The files of DHTML are stored with the .dhtm extension in a system.

system.	
6. A simple page which is created by a user without using the scripts or styles called as an HTML page.	6. A page which is created by a user using the HTML, CSS, DOM, and JavaScript technologies called a DHTML page.
7. This markup language does not need database connectivity.	7. This concept needs database connectivity because it interacts with users.

DHTML JavaScript

JavaScript can be included in HTML pages, which creates the content of the page as dynamic. We can easily type the JavaScript code within the <head> or <body> tag of a HTML page. If we want to add the external source file of JavaScript, we can easily add using the <src> attribute.

Following are the various examples, which describes how to use the JavaScript technology with the DHTML:

Scripting Language

A scripting language is a language that uses a sophisticated method to bring codes to a runtime environment. In key ways, scripting languages are made for specific runtime environments, and they automate some of the code implementation.

In that sense, they are modernizations of a system that previously used compilers to interpret inputs.

Examples of scripting language implementation involve their use in operating system shells and web browser technologies, and elsewhere, where the interpreter can enhance how the language is used.

DIWAKAR EDUCATION

Example: Python

A basic comparison of scripting languages in programming evolution involves Python, one of the most popular languages used for many new kinds of projects involving machine learning.

Python is known as a scripting language in its modular and automated build, compared to legacy programming languages like COBOL and BASIC, which are known as compiled languages.

Interpreted Languages

In many cases, a scripting language uses an interpreter instead of a compiler, and that's how We can tell whether a language is a scripting language or not.

Compiled languages use a compiler to make code into assembly language or machine language. By contrast, scripting languages and other interpreted languages use an interpreter.

The interpreter is responsible to interpret the source code for program execution. We could say that in a scripting language with an interpreter, the code is the language itself, and it gets interpreted relatively on-the-fly. Other kinds of systems like just-in-time compiling can also apply.

Hybrids

An example of the complexity of compiler languages and interpreted languages such as scripting languages is evident in the evolution of Sun Microsystems and its Java computer programming language set, which has been so much a fundamental part of computer science for so many years.

Classic Java is commonly known as a compiler language. It uses the traditional compiler system to convert code into machine language, as do C++ and other object oriented programming methods of its time.

However, languages like JavaScript are known as interpreter languages, where instead of using a compiler, an interpreter tool is used to change the way that the

DIWAKAR EDUCATION HUB

code syntax is used and implemented. If We, for example, have a virtual machine friendly language like ByteCode interpreting Java script for a compiler, that would be an interpreted language.

Benefits and Disadvantages of Scripting and Interpreted Languages

There are various benefits and disadvantages associated with the use of interpreted languages over compiler languages. The uniqueness of domain-specific scripting languages and their use in various runtime environments has been discussed. There's also the idea that interpreted language systems can help when distributed systems have different machine languages in play that make it difficult for compiled languages to bridge these cross-platform gaps.

On the other hand, some experts talk about latency with interpreted programs, just because the code has to run through an interpreter instead of being traditionally compiled. Experts have to assess these sorts of trade-offs as they consider whether using a scripting language makes sense in a given project environment.

Generally, though, the ability to abstract programming in this way is an attractive part of modernizing our codebase tools.

Java Servlet

Java Servlets are server-side Java program modules that process and answer client requests and implement the servlet interface. It helps in enhancing Web server functionality with minimal overhead, maintenance and support.

A servlet acts as an intermediary between the client and the server. As servlet modules run on the server, they can receive and respond to requests made by the client. Request and response objects of the servlet offer a convenient way to handle HTTP requests and send text data back to the client.

Since a servlet is integrated with the Java language, it also possesses all the Java features such as high portability, platform independence, security and Java database connectivity.

DIWAKAR EDUCATION HUB

There are two Java Servlet types: Basic and HTTP.

HTTP servlets are used as follows:

- *When an HTML form is submitted, the servlet processes and stores the data.*
- *When a client supplies a database query, the results are provided to the client by the servlet.*
- *In most cases, the server uses the common gateway interface (CGI).*

However, Java Servlets have many advantages over CGI, including:

- *A servlet runs in the same process, eliminating the need to create a new process for every request.*
- *The CGI program must be reloaded for each CGI request. A servlet, however, does not require reloading and remains in the memory between requests.*
- *A servlet answers multiple requests simultaneously by using one instance, saving memory and easily managing persistent data.*
- *The servlet engine runs in a sandbox or restricted environment, protecting the server from potentially harmful servlets.*

Servlet Life Cycle

The servlet life cycle is the Java servlet processing event sequence that occurs from servlet instance creation to destruction. The servlet life cycle is controlled by the container that deploys the servlet.

The servlet life cycle is made up of four stages:

- Instantiation
- Initialization
- Client request handling
- Destruction

DIWAKAR EDUCATION HUB

When a servlet request is mapped, the servlet container checks for the existence of a servlet class instance. If an instance does not exist, the Web container loads the servlet class, creates an instance of this class and initializes this instance by calling the init() method.

The initialization process is completed prior to client request handling. The container does not call the init() method again, unless a servlet is reloaded. After the instantiation and initialization are completed, the servlet container calls the service() method to respond to the request. When the servlet is no longer needed, the container destroys the servlet with the destroy() method. This method is also executed only one time.

Applications of Servlet

- Read the explicit data sent by the clients (browsers). This includes an HTML form on a Web page or it could also come from an applet or a custom HTTP client program.
- Read the implicit HTTP request data sent by the clients (browsers). This includes cookies, media types and compression schemes the browser understands, and so forth.
- Process the data and generate the results. This process may require talking to a database, executing an RMI or CORBA call, invoking a Web service, or computing the response directly.
- Send the explicit data (i.e., the document) to the clients (browsers). This document can be sent in a variety of formats, including text (HTML or XML), binary (GIF images), Excel, etc.
- Send the implicit HTTP response to the clients (browsers). This includes telling the browsers or other clients what type of document is being returned (e.g., HTML), setting cookies and caching parameters, and other such tasks.

Java Applet

DIWAKAR EDUCATION HUB

A Java applet is a small dynamic Java program that can be transferred via the Internet and run by a Java-compatible Web browser. The main difference between Java-based applications and applets is that applets are typically executed in an AppletViewer or Java-compatible Web browser. All applets import the java.awt package.

The following are two issues with Java applets:

- Security: Java resolves the security issue by restricting applets to Java's execution environment and preventing access to system resources.
- Portability: Portability is defined as the applet's ability to run on different computers and operating systems.

Any browser with Java Virtual Machine (JVM) can execute bytecode, which is the output of a Java compiler and can be run only in a JVM. Bytecode is Java's solution for security and portability.

Application Conversion to Applets

It is easy to convert a graphical Java application (that is, an application that uses the AWT and that We can start with the Java program launcher) into an applet that We can embed in a web page.

Following are the specific steps for converting an application to an applet.

- Make an HTML page with the appropriate tag to load the applet code.
- Supply a subclass of the JApplet class. Make this class public. Otherwise, the applet cannot be loaded.
- Eliminate the main method in the application. Do not construct a frame window for the application. Our application will be displayed inside the browser.
- Move any initialization code from the frame window constructor to the init method of the applet. We don't need to explicitly construct the applet object. The browser instantiates it for We and calls the init method.

DIWAKAR EDUCATION HUB

- Remove the call to setSize; for applets, sizing is done with the width and height parameters in the HTML file.
- Remove the call to setDefaultCloseOperation. An applet cannot be closed; it terminates when the browser exits.
- If the application calls setTitle, eliminate the call to the method. Applets cannot have title bars. (You can, of course, title the web page itself, using the HTML title tag.)
- Don't call setVisible(true). The applet is displayed automatically.

Computer Graphics

Computer Graphics involves technology to access. The Process transforms and presents information in a visual form. The role of computer graphics insensible. In today life, computer graphics has now become a common element in user interfaces, T.V. commercial motion pictures.

Computer Graphics is the creation of pictures with the help of a computer. The end product of the computer graphics is a picture it may be a business graph, drawing, and engineering.

In computer graphics, two or three-dimensional pictures can be created that are used for research. Many hardware devices algorithm has been developing for improving the speed of picture generation with the passes of time. It includes the creation storage of models and image of objects. These models for various fields like engineering, mathematical and so on.

Today computer graphics is entirely different from the earlier one. It is not possible. It is an interactive user can control the structure of an object of various input devices.

Definition of Computer Graphics:

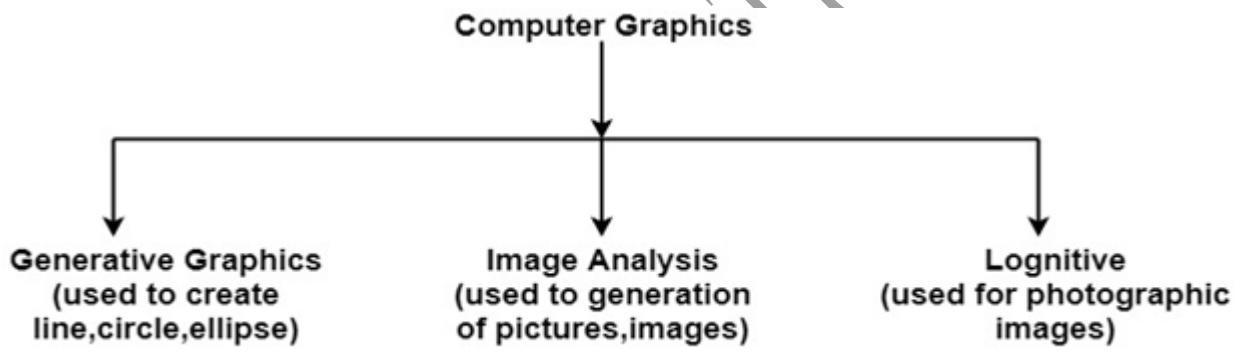
It is the use of computers to create and manipulate pictures on a display device. It comprises of software techniques to create, store, modify, represents pictures.

DIWAKAR EDUCATION HUB

Why computer graphics used?

Suppose a shoe manufacturing company want to show the sale of shoes for five years. For this vast amount of information is to store. So a lot of time and memory will be needed. This method will be tough to understand by a common man. In this situation graphics is a better alternative. Graphics tools are charts and graphs. Using graphs, data can be represented in pictorial form. A picture can be understood easily just with a single look.

Interactive computer graphics work using the concept of two-way communication between computer users. The computer will receive signals from the input device, and the picture is modified accordingly. Picture will be changed quickly when we apply command.



Display Devices in Computer Graphics

The display device is an output device used to represent the information in the form of images (visual form). Display systems are mostly called a **video monitor** or **Video display unit (VDU).**

Display devices are designed to model, display, view, or display information. The purpose of display technology is to simplify information sharing.

Today, the demand for high-quality displays is increasing.

There are some display devices given below:

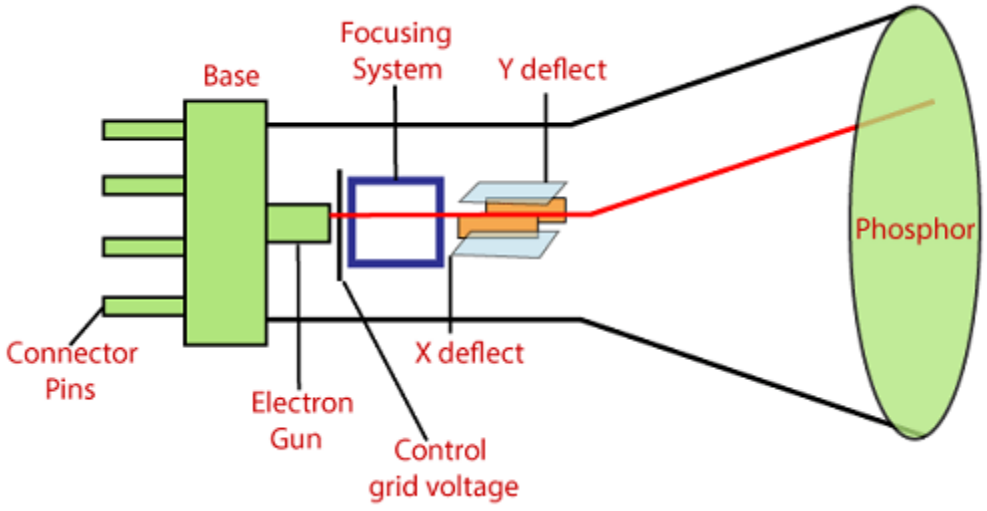
1. Cathode-Ray Tube(CRT)

DIWAKAR TION HUB

2. Color CRT Monitor
3. Liquid crystal display(LCD)
4. Light Emitting Diode(LED)
5. Direct View Storage Tubes(DVST)
6. Plasma Display
7. 3D Display

1. **Cathode-ray Tube (CRT):** Here, CRT stands for Cathode ray tube. It is a technology which is used in traditional computer monitor and television.

Cathode ray tube is a particular type of vacuum tube that displays images when an electron beam collides on the radiant surface.



Component of CRT:

- **Electron Gun:** The electron gun is made up of several elements, mainly a heating filament (heater) and a cathode.

The electron gun is a source of electrons focused on a narrow beam facing the CRT.

- **Focusing & Accelerating Anodes:** These anodes are used to produce a narrow and sharply focused beam of electrons.
- **Horizontal & Vertical Deflection Plates:** These plates are used to guide the path of the electron the beam. The plates produce an electromagnetic field that bends the electron beam through the area as it travels.
- **Phosphorus-coated Screen:** The phosphorus coated screen is used to produce bright spots when the high-velocity electron beam hits it.

There are two ways to represent an object on the screen:

1. **Raster Scan:** It is a scanning technique in which the electron beam moves along the screen. It moves from top to bottom, covering one line at a time.

A raster scan is based on pixel intensity control display as a rectangular box on the screen called a **raster**.

Picture description is stored in the memory area called as **Refresh buffer, or Frame Buffer.**

Frame buffer is also known as **Raster or Bitmap.** Raster scan provides the refresh rate of 60 to 80 frames per second.

For Example: Television

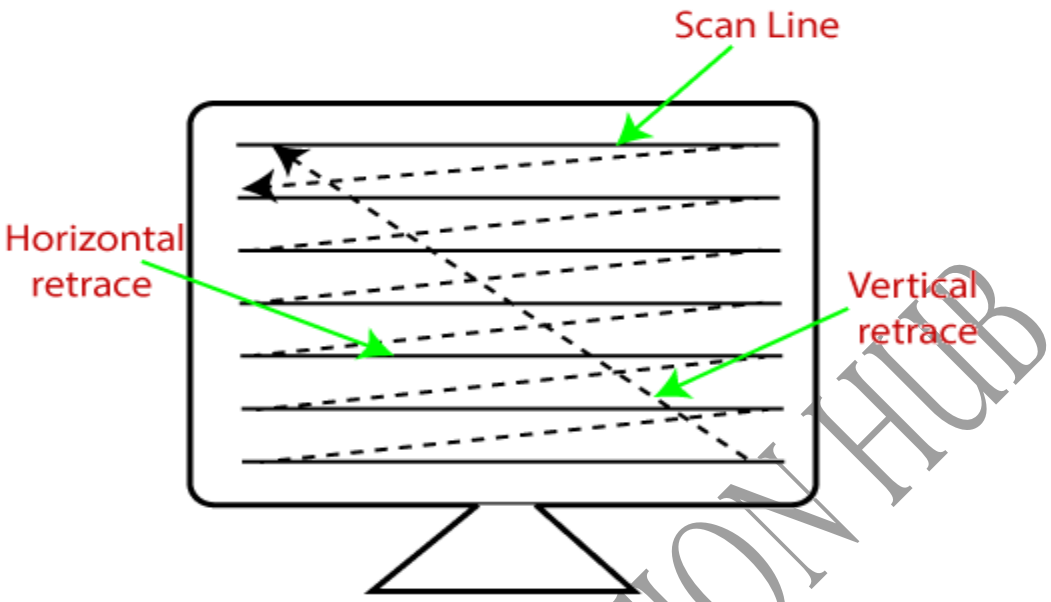
The beam refreshing has two types:

1. Horizontal Retracing
2. Vertical Retracing

When the beam starts from the top left corner and reaches bottom right, and again return to the top left, it is called the **vertical retrace**.

It will call back from top to bottom more horizontally as a horizontal reversal.

DIWAKAR EDUCATION HUB



Advantages:

1. Real image
2. Many colors to be produced
3. Dark scenes can be pictured

Disadvantages:

1. Less resolution
2. Display picture line by line
3. More costly

2. Random Scan (Vector scan): It is also known as stroke-writing display or calligraphic display. In this, the electron beam points only to the area in which the picture is to be drawn.

It uses an electron beam like a pencil to make a line image on the screen. The image is constructed from a sequence of straight-line segments. On the screen, each line segment is drawn by the beam to pass from one point on the screen to the other, where its x & y coordinates define each point.

DIWAKAR EDUCATION HUB

After compilation of picture drawing, the system cycle back to the first line and create all the lines of picture 30 to 60 times per second.

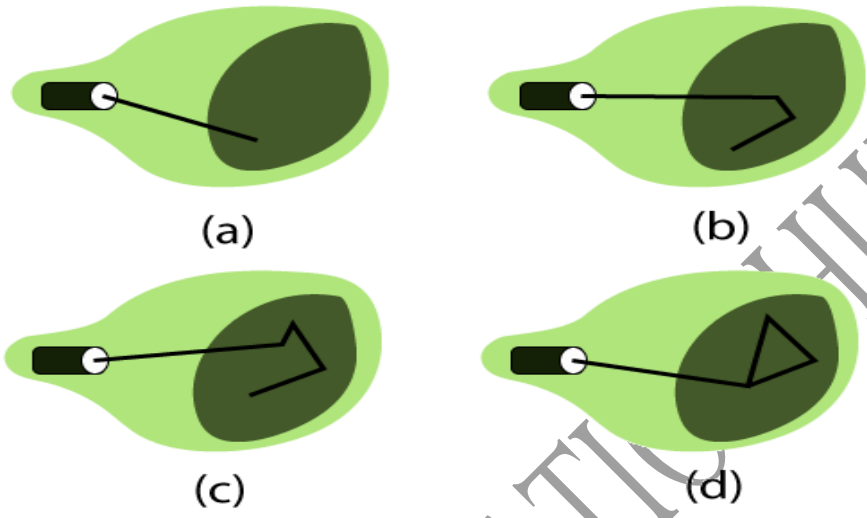


Fig: A Random Scan display draws the lines of an object in a specific order

Advantages:

1. High Resolution
2. Draw smooth line Drawing

Disadvantages:

1. It does only the wireframe.
2. It creates complex scenes due to flicker.

2.Color CRT Monitor: It is similar to a CRT monitor.

The basic idea behind the color CRT monitor is to combine three basic colors- Red, Green, and Blue. By using these three colors, we can produce

millions of different colors.

The two basic color display producing techniques are:

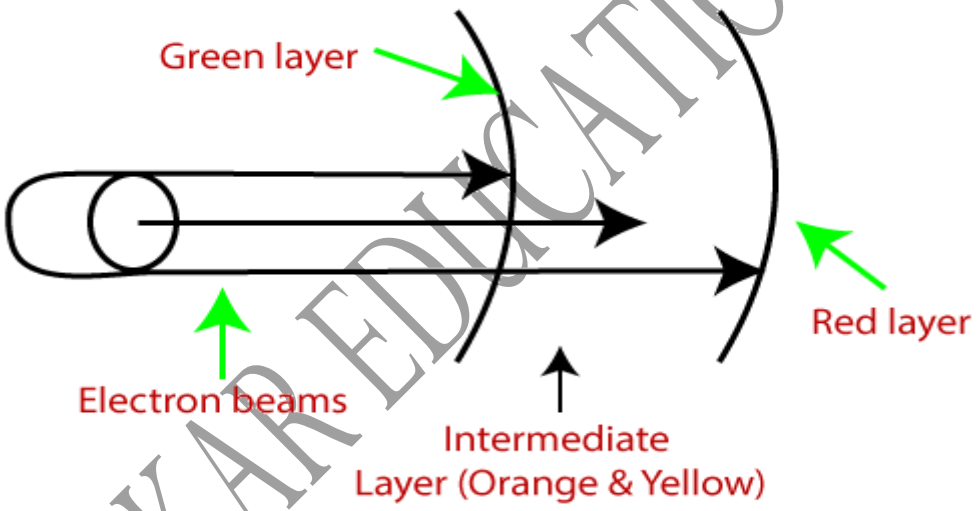
DIWAKAR EDUCATION HUB

1. **Beam–Penetration Method:** It is used with a **random scan** monitor for displaying pictures. There are two phosphorus layers- Red and Green are coated inside the screen. The color shown depends on how far the electron beam penetrates the phosphorus surface.

A powerful electron beam penetrates the CRT, it passes through the red layer and excites the green layer within.

A beam with slow electrons excites only the red layer.

A beam with the medium speed of electrons, a mixture of red and green light is emitted to display two more colors- orange and yellow.



DIWAKAR EDUCATION HUB

Advantages:

1. Better Resolution
2. Half cost
3. Inexpensive

Disadvantages:

1. Only four possible colors
2. Time Consuming

2.Shadow–Mask Method: It is used with a **raster scan** monitor for displaying pictures. It has more range of color than the beam penetration method. It is used in television sets and monitors.

Structure:

1. It has three phosphorus color dots at each position of the pixel.

First Dot: **Red** color

Second Dot: **Green** color

Third Dot: **Blue** color

1. It has three different guns. Each for one color.
2. It has a metal screen or plate just before the phosphorus screen, named **"Shadow-Mask."**
3. It also has a shadow grid just behind the phosphorus coated screen with tiny holes in a triangular shape.

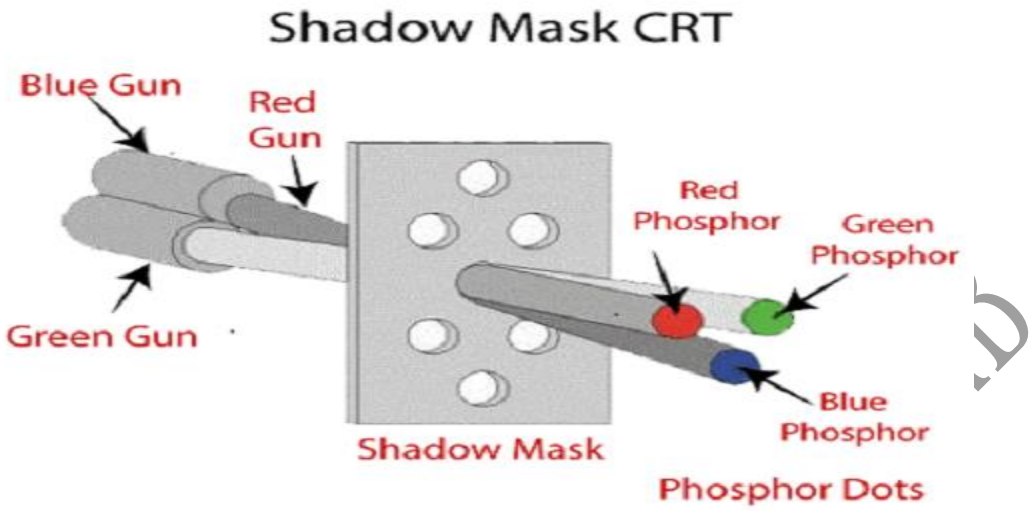
Working: A Shadow Mask is a metal plate with tiny holes present inside a color monitor.

A Shadow Mask directs the beam by consuming the electrons so that the beam hits only the desired point and displays a resulting picture.

It has three different guns. These guns direct their beams to shadow mask, which allows them to pass. It is a task of a shadow mask to direct the beam on its particular dot on the screen and produce a picture on the screen.

A Shadow Mask can display a wider range of pictures than beam penetration.

DIWAKAR EDUCATION HUB



Advantages:

1. Display a wider range picture.
2. Display realistic images.
3. In-line arrangement of RGB color.

Disadvantages:

Difficult to cover all three beams on the same hole.

Poor Resolution.

3.Liquid crystal display (LCD): The LCD depends upon the light modulating properties of liquid crystals.

LCD is used in watches and portable computers. LCD requires an AC power supply instead of DC, so it is difficult to use it in circuits.

It generally works on flat panel display technology. LCD consumes less power than LED. The LCD screen uses the liquid crystal to turn pixels on or off.

Liquid Crystals are a mixture of solid and liquid. When the current flows inside it, its position changes into the desired color.

DIWAKAR EDUCATION

For Example: TFT(Thin Film Transistor)

Advantages:

1. Produce a bright image
2. Energy efficient
3. Completely flat screen

Disadvantages:

Fixed aspect ratio & Resolution

Lower Contrast

More Expensive

4.Light Emitting Diode (LED): LED is a device which emits when current passes through it. It is a semiconductor device.

The size of the LED is small, so we can easily make any display unit by arranging a large number of LEDs.

LED consumes more power compared to LCD. LED is used on TV, smartphones, motor vehicles, traffic light, etc.

LEDs are powerful in structure, so they are capable of withstanding mechanical pressure. LED also works at high temperatures.

Advantages:

1. The Intensity of light can be controlled.
2. Low operational Voltage.
3. Capable of handling the high temperature.

Disadvantages:

More Power Consuming than LCD.

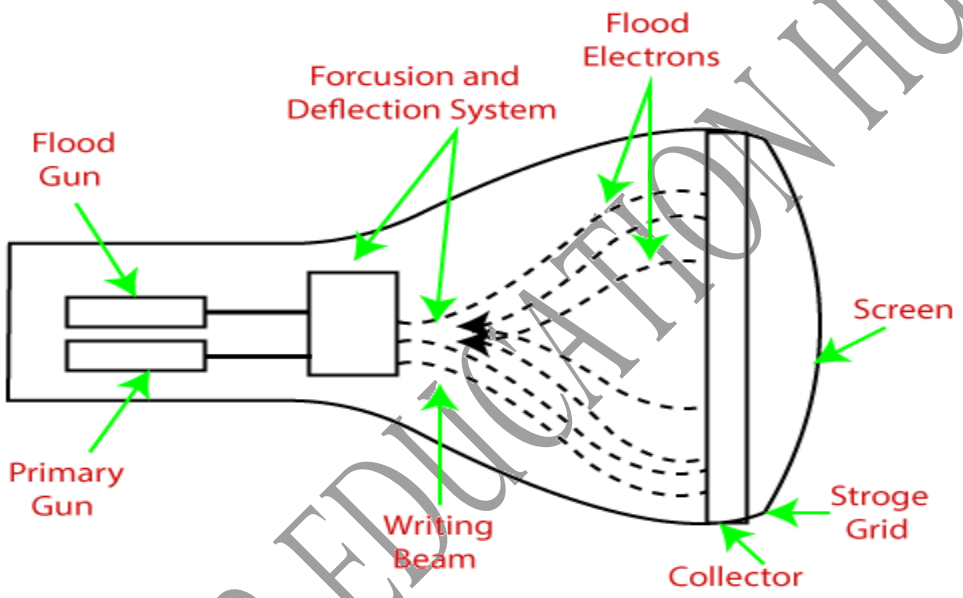
DIWAKAR EDUCATION HUB

5.Direct View Storage Tube (DVST): It is used to store the picture information as a charge distribution behind the phosphor-coated screen.

There are two guns used in DVST:

1.Primary Gun: It is used to store the picture information.

2.Flood / Secondary Gun: It is used to display a picture on the screen.



Advantages:

1. Less Time Consuming
2. No Refreshing Required
3. High-Resolution
4. Less Cost

Disadvantages:

The specific part of the image cannot be erased.

They do not display color.

DIWAKAR EDUCATION HUB

6.Plasma Display: It is a type of flat panel display which uses tiny plasma cells. It is also known as **the Gas-Discharge display**.

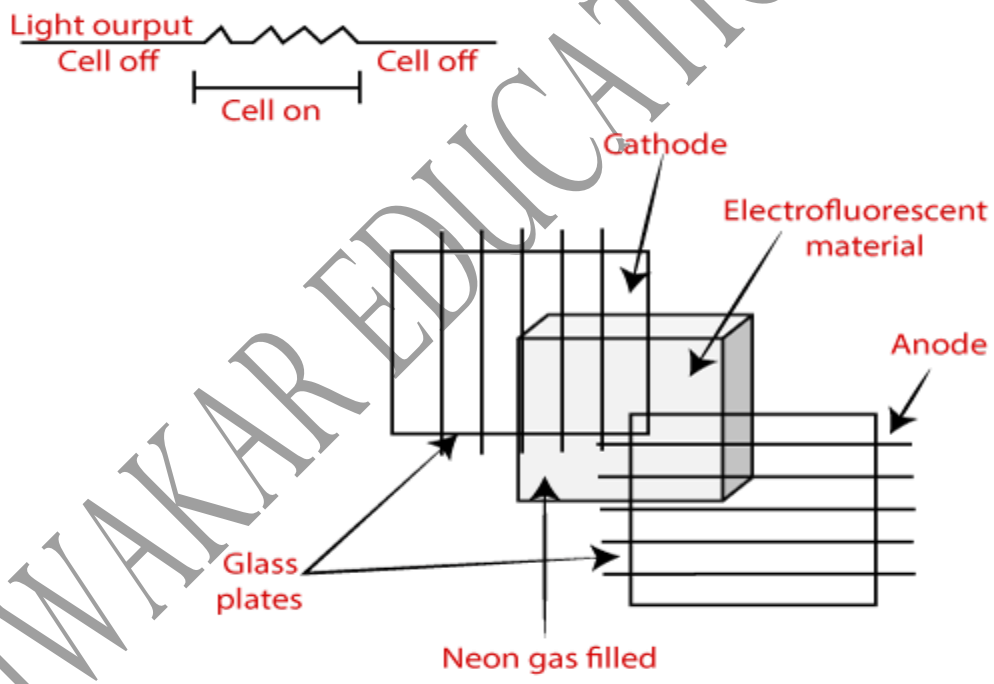
Components of plasma display:

1.Anode: It is used to deliver a positive voltage. It also has the line wires.

2.Cathode: It is used to provide negative voltage to gas cells. It also has fine wires.

3.Gas Plates: These plates work as capacitors. When we pass the voltage, the cell lights regularly.

4.Fluorescent cells: It contains small pockets of gas liquids when the voltage is passed to this neon gas. It emits light.



DIWAKAR EDUCATION HUB

Advantages:

1. Wall Mounted
2. Slim
3. Wider angle

Disadvantages:

Phosphorus loses luminosity over time.

It consumes more electricity than LCD.

Large Size

7.3D Display: It is also called stereoscope display technology. This technology is capable of bringing depth perception to the viewer.

It is used for 3D gaming and 3D TVs.

For Example: Fog Display, Holographic Display, Retina Display Etc.

Advantages:

- Impressive Picture Quality
- High Definition
- Motion Communicates

Disadvantage:

- Expensive
- Binocular Fusion

Raster Scan

In a raster scan system, the electron beam is swept across the screen, one row at a time from top to bottom. As the electron beam moves across each row, the beam intensity is turned on and off to create a pattern of illuminated spots.

A Raster Scan Display is based on intensity control of pixels in the form of a rectangular box called Raster on the screen. Information of on and off pixels is stored in refresh buffer or Frame buffer. Televisions in our house are based on Raster Scan Method. The raster scan system can store information of each pixel

DIWAKAR EDUCATION HUB

position, so it is suitable for realistic display of objects. Raster Scan provides a refresh rate of 60 to 80 frames per second.

Frame Buffer is also known as Raster or bit map. In Frame Buffer the positions are called picture elements or pixels. Beam refreshing is of two types. First is horizontal retracing and second is vertical retracing. When the beam starts from the top left corner and reaches the bottom right scale, it will again return to the top left side called at vertical retrace. Then it will again more horizontally from top to bottom call as horizontal retracing shown in fig:

Types of Scanning or travelling of beam in Raster Scan

1. Interlaced Scanning
2. Non-Interlaced Scanning

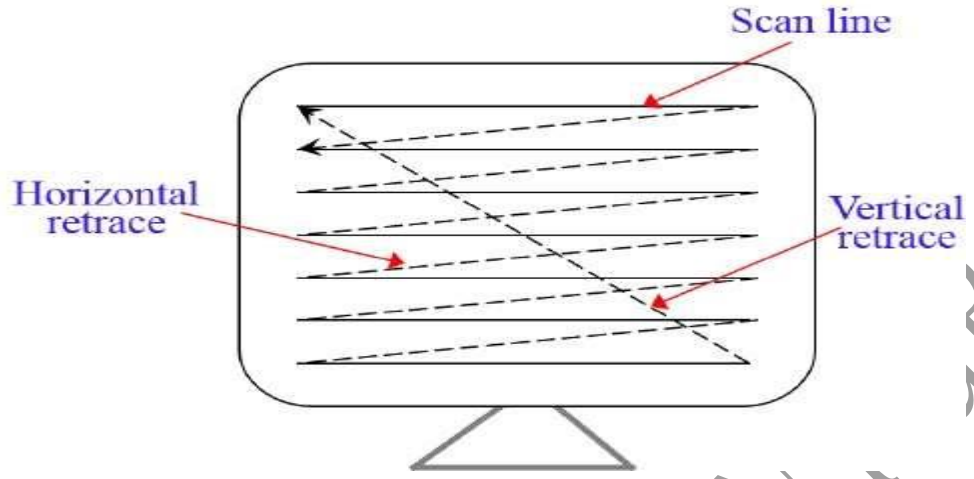
In Interlaced scanning, each horizontal line of the screen is traced from top to bottom. Due to which fading of display of object may occur. This problem can be solved by Non-Interlaced scanning. In this first of all odd numbered lines are traced or visited by an electron beam, then in the next circle, even number of lines are located.

For non-interlaced display refresh rate of 30 frames per second used. But it gives flickers. For interlaced display refresh rate of 60 frames per second is used.

Picture definition is stored in memory area called the **Refresh Buffer** or **Frame Buffer**. This memory area holds the set of intensity values for all the screen points. Stored intensity values are then retrieved from the refresh buffer and "painted" on the screen one row scanlinescanline at a time as shown in the following illustration.

Each screen point is referred to as a **pixel** pictureelementpictureelement or **pel**. At the end of each scan line, the electron beam returns to the left side of the screen to begin displaying the next scan line.

DIWAKAR EDUCATION HUB



Advantages:

1. Realistic image
2. Million Different colors to be generated
3. Shadow Scenes are possible.

Disadvantages:

1. Low Resolution
2. Expensive

Random Scan VectorScanVectorScan

In this technique, the electron beam is directed only to the part of the screen where the picture is to be drawn rather than scanning from left to right and top to bottom as in raster scan. It is also called **vector display, stroke-writing display,** or **calligraphic display**.

Random Scan System uses an electron beam which operates like a pencil to create a line image on the CRT screen. The picture is constructed out of a sequence of straight-line segments. Each line segment is drawn on the screen by directing the beam to move from one point on the screen to the next, where its x & y coordinates define each point. After drawing the picture. The system cycles

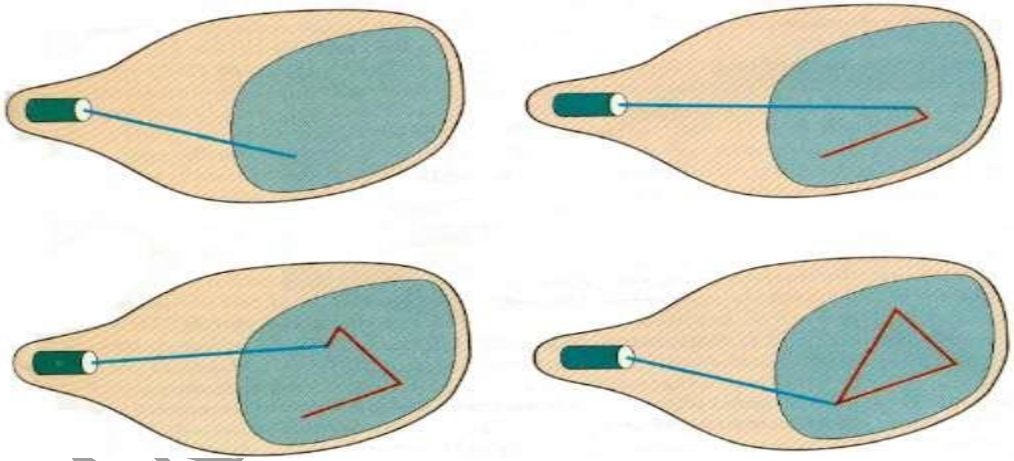
DIWAKAR EDUCATION

back to the first line and design all the lines of the image 30 to 60 time each second. The process is shown in fig:

Random-scan monitors are also known as vector displays or stroke-writing displays or calligraphic displays.

Picture definition is stored as a set of line-drawing commands in an area of memory referred to as the **refresh display file**. To display a specified picture, the system cycles through the set of commands in the display file, drawing each component line in turn. After all the line-drawing commands are processed, the system cycles back to the first line command in the list.

Random-scan displays are designed to draw all the component lines of a picture 30 to 60 times each second.



Advantages:

1. A CRT has the electron beam directed only to the parts of the screen where an image is to be drawn.
2. Produce smooth line drawings.
3. High Resolution

Disadvantages:

1. Random-Scan monitors cannot display realistic shades scenes.

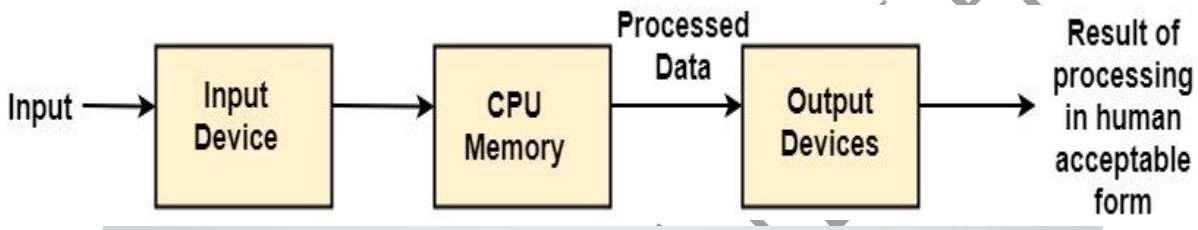
Differentiate between Random and Raster Scan Display:

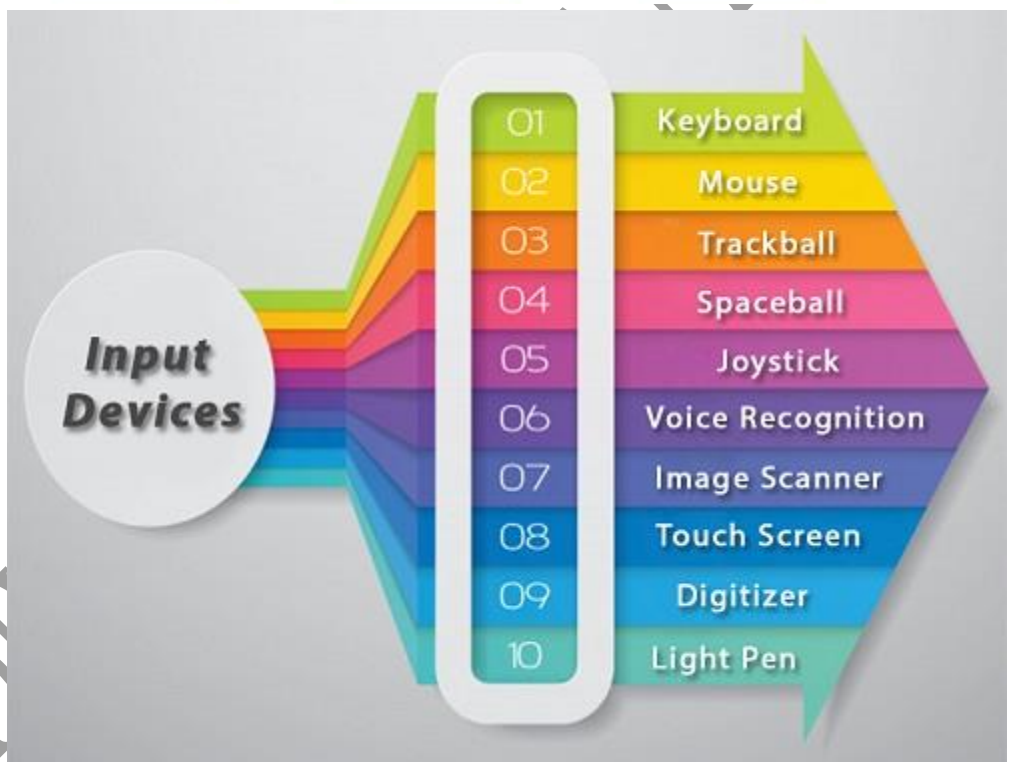
Random Scan	Raster Scan
1. It has high Resolution	1. Its resolution is low.
2. It is more expensive	2. It is less expensive
3. Any modification if needed is easy	3.Modification is tough
4. Solid pattern is tough to fill	4.Solid pattern is easy to fill
5. Refresh rate depends or resolution	5. Refresh rate does not depend on the picture.
6. Only screen with view on an area is displayed.	6. Whole screen is scanned.
7. Beam Penetration technology come under it.	7. Shadow mark technology came under this.
8. It does not use interlacing method.	8. It uses interlacing
9. It is restricted to line drawing	9. It is suitable for realistic display.

applications	

Input Devices

The Input Devices are the hardware that is used to transfer transfers input to the computer. The data can be in the form of text, graphics, sound, and text. Output device display data from the memory of the computer. Output can be text, numeric data, line, polygon, and other objects.





These Devices include:

1. Keyboard
2. Mouse

3. Trackball
4. Spaceball
5. Joystick
6. Light Pen
7. Digitizer
8. Touch Panels
9. Voice Recognition
10. Image Scanner

Keyboard:

The most commonly used input device is a keyboard. The data is entered by pressing the set of keys. All keys are labeled. A keyboard with 101 keys is called a QWERTY keyboard.

The keyboard has alphabetic as well as numeric keys. Some special keys are also available.

1. **Numeric Keys:** 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
2. **Alphabetic keys:** a to z (lower case), A to Z (upper case)
3. **Special Control keys:** Ctrl, Shift, Alt
4. **Special Symbol Keys:** ; , " ? @ ~ ? :
5. **Cursor Control Keys:** ↑ → ← ↓
6. **Function Keys:** F1 F2 F3....F9.
7. **Numeric Keyboard:** It is on the right-hand side of the keyboard and used for fast entry of numeric data.

Function of Keyboard:

DIWAKAR EDUCATION HUB

1. Alphanumeric Keyboards are used in CAD. (Computer Aided Drafting)
2. Keyboards are available with special features line screen co-ordinates entry, Menu selection or graphics functions, etc.
3. Special purpose keyboards are available having buttons, dials, and switches. Dials are used to enter scalar values. Dials also enter real numbers. Buttons and switches are used to enter predefined function values.

Advantage:

1. Suitable for entering numeric data.
2. Function keys are a fast and effective method of using commands, with fewer errors.

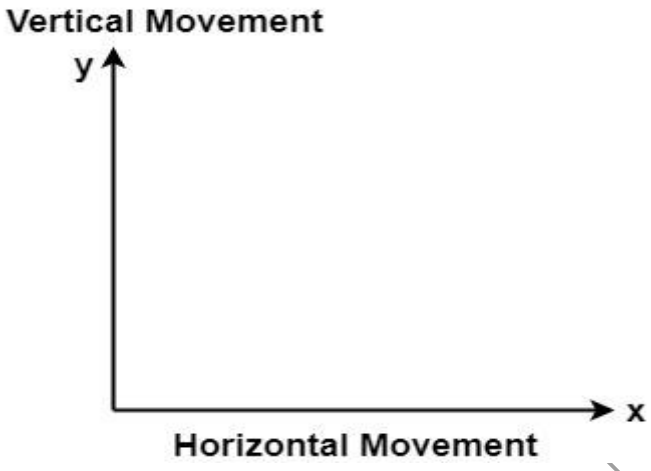
Disadvantage:

1. Keyboard is not suitable for graphics input.

Mouse:

A Mouse is a pointing device and used to position the pointer on the screen. It is a small palm size box. There are two or three depression switches on the top. The movement of the mouse along the x-axis helps in the horizontal movement of the cursor and the movement along the y-axis helps in the vertical movement of the cursor on the screen. The mouse cannot be used to enter text. Therefore, they are used in conjunction with a keyboard.

DIWAKAR EDUCATION HUB

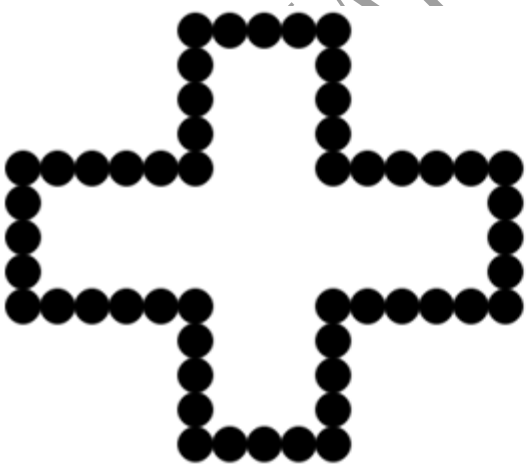


Advantage:

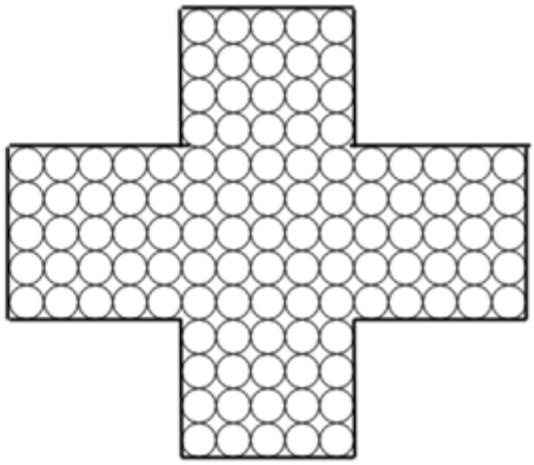
1. Easy to use
2. Not very expensive

Filled Area Primitives:

Region filling is the process of filling image or region. Filling can be of boundary or interior region as shown in fig. Boundary Fill algorithms are used to fill the boundary and flood-fill algorithm are used to fill the interior.



Boundary Filled Region



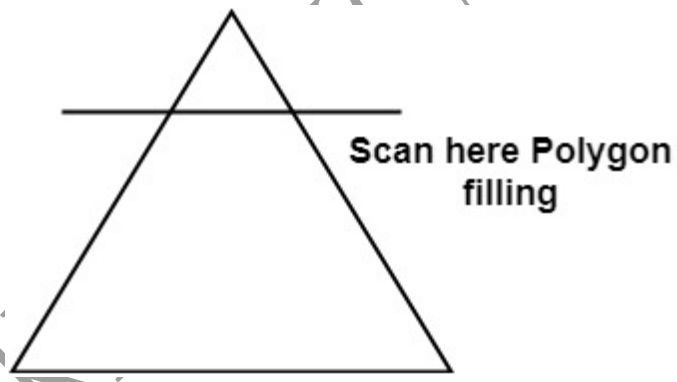
Interior or Flood Filled Region

REDUCATION HUB

Scan Line Polygon Fill Algorithm:

This algorithm lines interior points of a polygon on the scan line and these points are done on or off according to requirement. The polygon is filled with various colors by coloring various pixels.

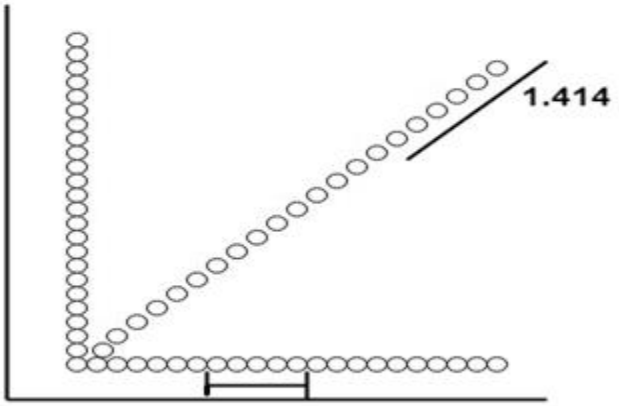
In above figure polygon and a line cutting polygon in shown. First of all, scanning is done. Scanning is done using raster scanning concept on display device. The beam starts scanning from the top left corner of the screen and goes toward the bottom right corner as the endpoint. The algorithms find points of intersection of the line with polygon while moving from left to right and top to bottom. The various points of intersection are stored in the frame buffer. The intensities of such points is keep high. Concept of coherence property is used. According to this property if a pixel is inside the polygon, then its next pixel will be inside the polygon.



Side effects of Scan Conversion:

1. Staircase or Jagged: Staircase like appearance is seen while the scan was converting line or circle.

2. Unequal Intensity: It deals with unequal appearance of the brightness of different lines. An inclined line appears less bright as compared to the horizontal and vertical line.

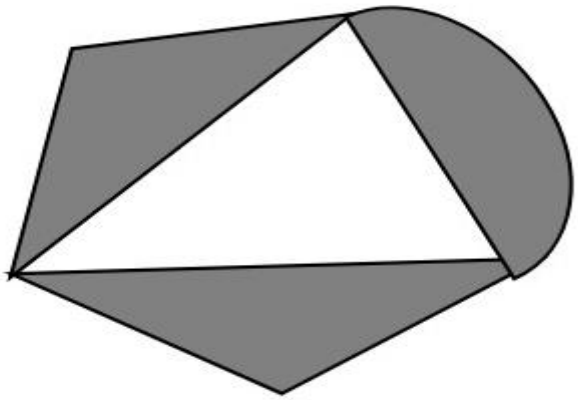


Pixels along with horizontal line are 1 unit apart and vertical. Pixels along diagonal line are 1.414 units.

Flood Fill Algorithm:

In this method, a point or seed which is inside region is selected. This point is called a seed point. Then four connected approaches or eight connected approaches is used to fill with specified color.

The flood fill algorithm has many characters similar to boundary fill. But this method is more suitable for filling multiple colors boundary. When boundary is of many colors and interior is to be filled with one color we use this algorithm.



In fill algorithm, we start from a specified interior point (x, y) and reassign all pixel values are currently set to a given interior color with the desired color. Using

either a 4-connected or 8-connected approaches, we then step through pixel positions until all interior points have been repainted.

Disadvantage:

1. Very slow algorithm
2. May be fail for large polygons
3. Initial pixel required more knowledge about surrounding pixels.

Algorithm:

```
Procedure floodfill (x, y,fill_ color, old_color: integer)
   If (getpixel (x, y)=old_color)
  {
   setpixel (x, y, fill_color);
   fill (x+1, y, fill_color, old_color);
    fill (x-1, y, fill_color, old_color);
   fill (x, y+1, fill_color, old_color);
   fill (x, y-1, fill_color, old_color);
    }
}
```

Program1: To implement 4-connected flood fill algorithm:

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
```

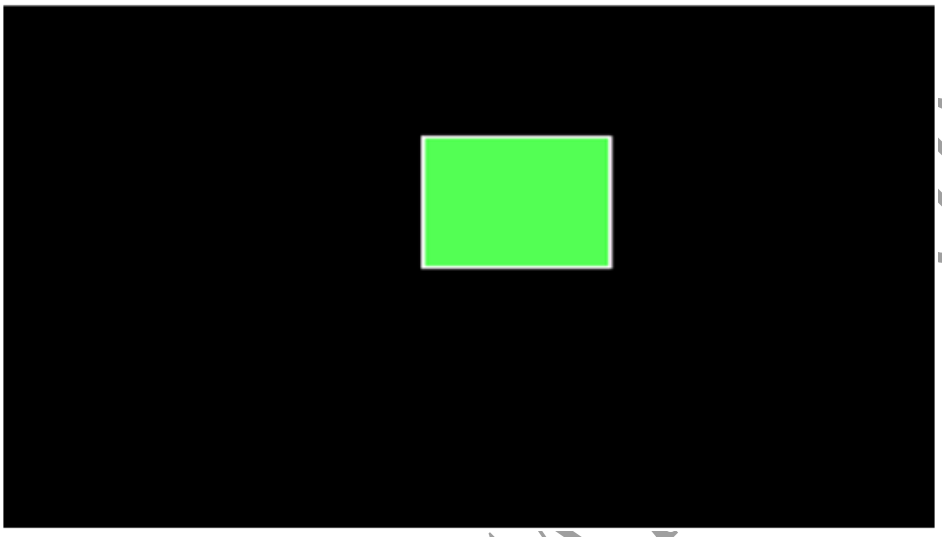
DIWAKAR EDUCATION HUB

```
#include<dos.h>
void flood(int,int,int,int);
void main()
{
   intgd=DETECT,gm;
   initgraph(&gd,&gm,"C:/TURBOC3/bgi");
   rectangle(50,50,250,250);
   flood(55,55,10,0);
   getch();
}
void flood(intx,inty,intfillColor, intdefaultColor)
{
   if(getpixel(x,y)==defaultColor)
   {
      delay(1);
      putpixel(x,y,fillColor);
      flood(x+1,y,fillColor,defaultColor);
      flood(x-1,y,fillColor,defaultColor);
      flood(x,y+1,fillColor,defaultColor);
      flood(x,y-1,fillColor,defaultColor);
   }
```

DIWAKAR EDUCATION HUB

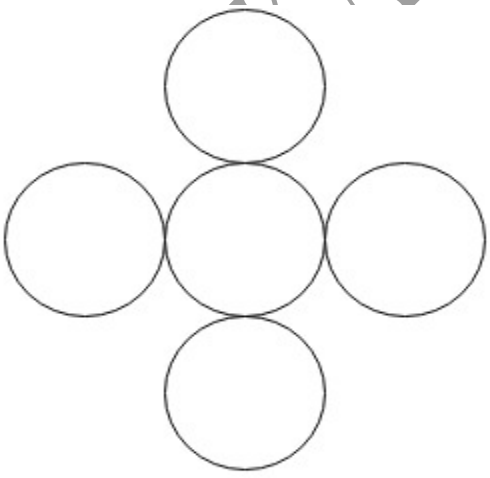
```
}
```

Output:

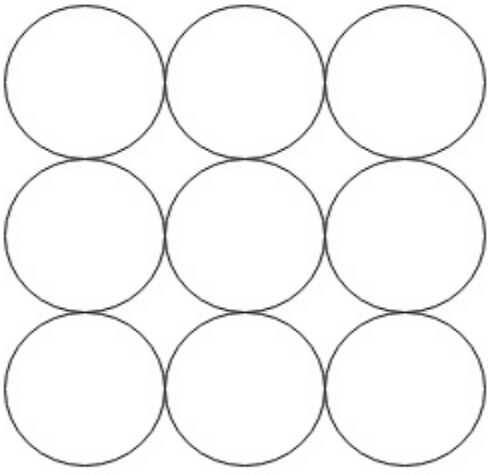


Boundary Filled Algorithm:

This algorithm uses the recursive method. First of all, a starting pixel called as the seed is considered. The algorithm checks boundary pixel or adjacent pixels are colored or not. If the adjacent pixel is already filled or colored then leave it, otherwise fill it. The filling is done using four connected or eight connected approaches.



Four Connected



Eight Connected

Four connected approaches is more suitable than the eight connected approaches.

1. Four connected approaches: In this approach, left, right, above, below pixels are tested.

2. Eight connected approaches: In this approach, left, right, above, below and four diagonals are selected.

Boundary can be checked by seeing pixels from left and right first. Then pixels are checked by seeing pixels from top to bottom. The algorithm takes time and memory because some recursive calls are needed.

Problem with recursive boundary fill algorithm:

It may not fill regions sometimes correctly when some interior pixel is already filled with color. The algorithm will check this boundary pixel for filling and will found already filled so recursive process will terminate. This may vary because of another interior pixel unfilled.

So check all pixels color before applying the algorithm.

Algorithm:

```
Procedure fill (x, y, color, color1: integer)
int c;
c=getpixel (x, y);
if (c!=color) (c!=color1)
{
   setpixel (x, y, color)
   fill (x+1, y, color, color 1);
    fill (x-1, y, color, color 1);
```

DIWAKAR EDUCATION HUB

```
    fill (x, y+1, color, color 1);

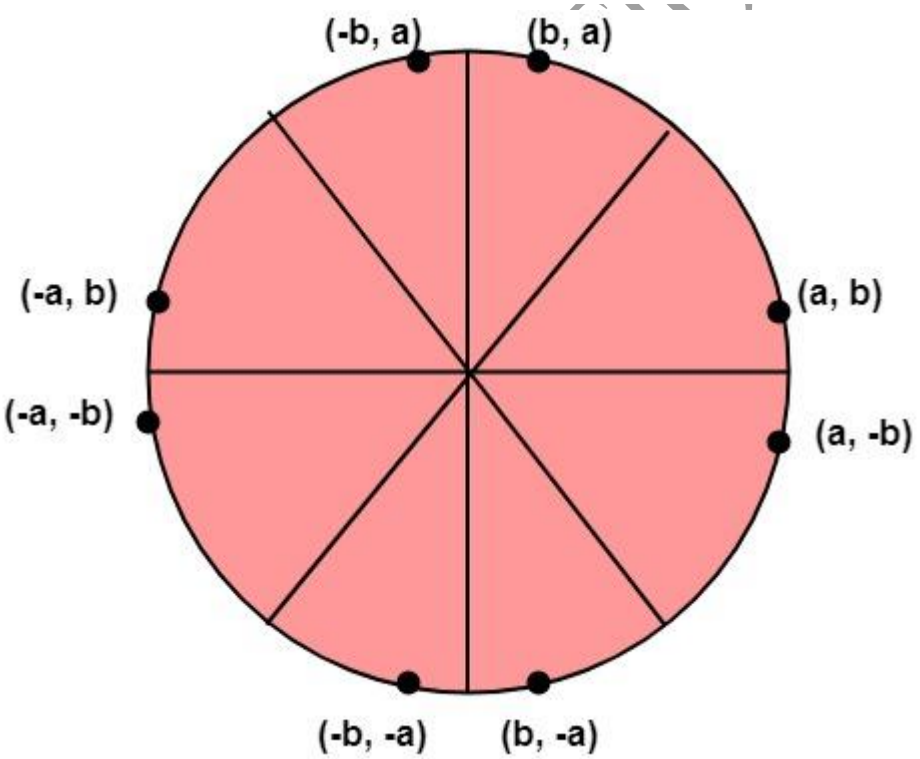
    fill (x, y-1, color, color 1);

}
```

MidPoint Circle Algorithm

It is based on the following function for testing the spatial relationship between the arbitrary point (x, y) and a circle of radius r centered at the origin:

$$f(x, y) = x^2 + y^2 - r^2 \quad \begin{bmatrix} < 0 \text{ for } (x,y) \text{inside the circle} \\ = 0 \text{ for } (x,y) \text{on the circle} \\ > 0 \text{ for } (x,y) \text{outside the circle} \end{bmatrix} \text{.....equation 1}$$



Now, consider the coordinates of the point halfway between pixel T and pixel S

This is called midpoint $(x_{i+1}, y_i - \frac{1}{2})$ and we use it to define a decision parameter:

$P_i = f(x_{i+1}, y_i - \frac{1}{2}) = (x_{i+1})^2 + (y_i - \frac{1}{2})^2 - r^2$ .....equation 2

If P_i is -ve $\Longrightarrow$midpoint is inside the circle and we choose pixel T

If P_i is+ve $\Longrightarrow$midpoint is outside the circle (or on the circle)and we choose pixel S.

The decision parameter for the next step is:

$P_{i+1}=(x_{i+1}+1)^2+(y_{i+1}-\frac{1}{2})^2- r^2$.....equation 3

Since $x_{i+1}=x_{i+1}$, we have

$$P_{i+1} - P_i = ((x_i + 1) + 1)^2-(x_i + 1)^2+(y_{i+1} - \tfrac{1}{2})^2-(y_i - \tfrac{1}{2})^2$$

$$= x_i^2 + 4 + 4x_i - x_i^2 + 1 - 2x_i + y_{i+1}^2 + \tfrac{1}{4} - y_{i+1} - y_i^2 - \tfrac{1}{4} - y_i$$

$$= 2(xi+1) +1+ (y_{i+1}^2 - y_i^2)-(y_{i+1} - y_i)$$

$$P_{i+1} = P_i + 2(x_i + 1) + 1 + (y_{i+1}^2 - y_i^2) - (y_{i+1} - y_i)$$.....equation 4

If pixel T is choosen $\Longrightarrow P_i<0$

We have $y_{i+1}=y_i$

If pixel S is choosen $\Longrightarrow P_i\geq 0$

We have $y_{i+1}=y_i-1$

Thus, $$P_{i+1} = \begin{bmatrix} P_i + 2(x_i + 1) + 1, & \text{if } P_i < 0 \\ P_i + 2(x_i + 1) + 1 - 2(y_i - 1), & \text{if } P_i \geq 0 \end{bmatrix}$$.....equation 5

We can continue to simplify this in n terms of (x_i,y_i) and get

$$P_{i+1} = \begin{bmatrix} P_i + 2x_i + 3, & \text{if } P_i < 0 \\ P_i + 2(x_i - y_i) + 5, & \text{if } P_i \geq 0 \end{bmatrix}$$.....equation 6

Now, initial value of P_i (0,r)from equation 2

$$P_1 = (0 + 1)^2+(r - \tfrac{1}{2})^2-r^2$$

$$= 1 + \tfrac{1}{4} - r^2 = \tfrac{5}{4} - r$$

We can put $\frac{5}{4} \cong 1$

∴r is an integer

So, P_1=1-r

Algorithm:

Step1: Put x =0, y =r in equation 2

We have p=1-r

Step2: Repeat steps while x ≤ y

Plot (x, y)

If (p<0)

Then set p = p + 2x + 3

Else

p = p + 2(x-y)+5

y =y - 1 (end if)

x =x+1 (end loop)

Step3: End

Program to draw a circle using Midpoint Algorithm:

```
#include <graphics.h>

#include <stdlib.h>

#include <math.h>

#include <stdio.h>

#include <conio.h>

#include <iostream.h>


class bresen
```

DIWAKAR EDUCATION HUB

```
{
   float x, y,a, b, r, p;
   public:
   void get ();
   void cal ();
};
   void main ()
   {
   bresen b;
   b.get ();
   b.cal ();
   getch ();
  }
   Void bresen :: get ()
  {
   cout<<"ENTER CENTER AND RADIUS";
   cout<< "ENTER (a, b)";
   cin>>a>>b;
   cout<<"ENTER r";
   cin>>r;
}
```

DIWAKAR EDUCATION HUB

```
void bresen ::cal ()
{
   /* request auto detection */
   int gdriver = DETECT,gmode, errorcode;
   int midx, midy, i;
   /* initialize graphics and local variables */
   initgraph (&gdriver, &gmode, " ");
   /* read result of initialization */
   errorcode = graphresult ();
   if (errorcode ! = grOK)    /*an error occurred */
   {
      printf("Graphics error: %s \n", grapherrormsg (errorcode);
      printf ("Press any key to halt:");
      getch ();
      exit (1); /* terminate with an error code */
   }
   x=0;
   y=r;
   putpixel (a, b+r, RED);
   putpixel (a, b-r, RED);
   putpixel (a-r, b, RED);
```

DIWAKAR EDUCATION HUB

```
putpixel (a+r, b, RED);
p=5/4)-r;
while (x<=y)
{
   If (p<0)
   p+= (4*x)+6;
   else
   {
      p+=(2*(x-y))+5;
      y--;
   }
   x++;
   putpixel (a+x, b+y, RED);
   putpixel (a-x, b+y, RED);
   putpixel (a+x, b-y, RED);
   putpixel (a+x, b-y, RED);
   putpixel (a+x, b+y, RED);
   putpixel (a+x, b-y, RED);
   putpixel (a-x, b+y, RED);
   putpixel (a-x, b-y, RED);
}
```

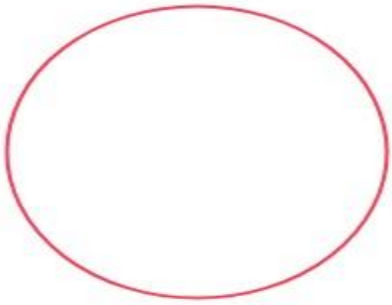
DIWAKAR EDUCATION HUB

```
}
```

Output:

```
ENTER CENTER AND RADIUS
ENTER (a, b) 319, 239

ENTER  r 100
```



Bresenham's Line Algorithm

This algorithm is used for scan converting a line. It was developed by Bresenham. It is an efficient method because it involves only integer addition, subtractions, and multiplication operations. These operations can be performed very rapidly so lines can be generated quickly.

In this method, next pixel selected is that one who has the least distance from true line.

The method works as follows:

Assume a pixel $P_1'(x_1',y_1')$,then select subsequent pixels as we work our may to the night, one pixel position at a time in the horizontal direction toward $P_2'(x_2',y_2')$.

Once a pixel in choose at any step

The next pixel is

1. Either the one to its right (lower-bound for the line)

DIWAKAR EDUCA

2. One top its right and up (upper-bound for the line)

The line is best approximated by those pixels that fall the least distance from the path between P_1',P_2'.

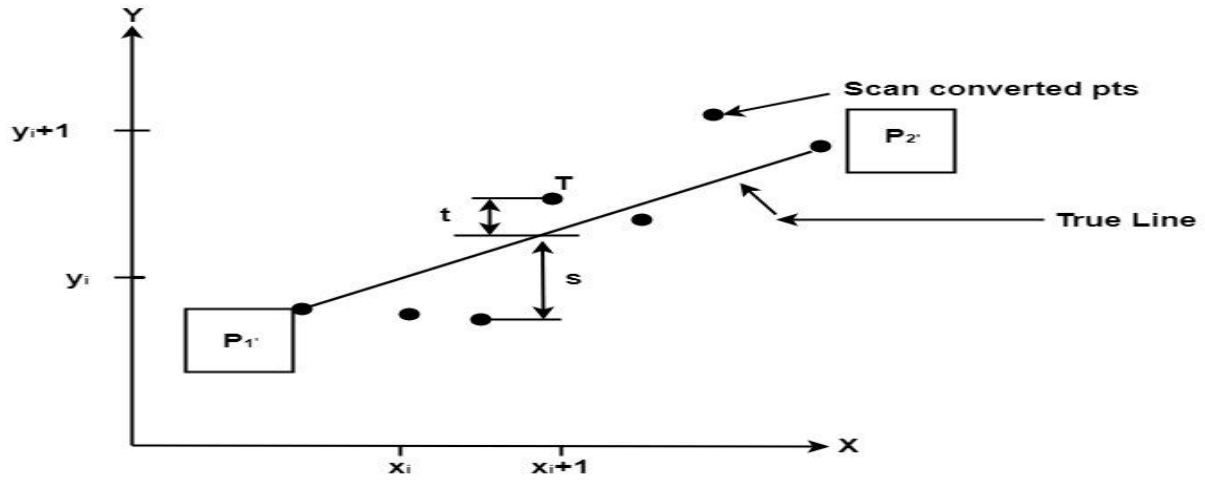


Fig: Scan Converting a line.

To chooses the next one between the bottom pixel S and top pixel T.

If S is chosen

We have $x_{i+1}=x_i+1$ and $y_{i+1}=y_i$

If T is chosen

We have $x_{i+1}=x_i+1$ and $y_{i+1}=y_i+1$

The actual y coordinates of the line at $x = x_{i+1}$is

$y=mx_{i+1}+b$

$$y = m(x_i + 1) + b$$

The distance from S to the actual line in y direction

$s = y-y_i$

The distance from T to the actual line in y direction

$t = (y_i+1)-y$

Now consider the difference between these 2 distance values

s - t

When (s-t) <0 ⟹ s < t

The closest pixel is S

When (s-t) ≥0 ⟹ s < t

The closest pixel is T

This difference is

s-t = (y-yi)-[(yi+1)-y]

= 2y - 2yi -1

$$s - t = 2m(x_i + 1) + 2b - 2y_i - 1$$ [Putting the value of (1)]

Substituting m by $\frac{\Delta y}{\Delta x}$ and introducing decision variable

$d_i=\Delta x\ (s-t)$

$d_i=\Delta x\ (2\ \frac{\Delta y}{\Delta x}\ (x_i+1)+2b-2y_i-1)$

$=2\Delta x y_i-2\Delta y-1\Delta x.2b-2y_i\Delta x-\Delta x$

$d_i=2\Delta y.x_i-2\Delta x.y_i+c$

Where $c= 2\Delta y+\Delta x\ (2b-1)$

We can write the decision variable d_{i+1} for the next slip on

$d_{i+1}=2\Delta y.x_{i+1}-2\Delta x.y_{i+1}+c$

$d_{i+1}-d_i=2\Delta y.(x_{i+1}-x_i)- 2\Delta x(y_{i+1}-y_i)$

Since x_(i+1)=x_i+1,we have

$d_{i+1}+d_i=2\Delta y.(x_i+1-x_i)- 2\Delta x(y_{i+1}-y_i)$

Special Cases

DIWAKAR EDUCATION HUB

If chosen pixel is at the top pixel T (i.e., $d_i \geq 0) \Longrightarrow y_{i+1}=y_i+1$

$d_{i+1}=d_i+2\Delta y-2\Delta x$

If chosen pixel is at the bottom pixel T (i.e., $d_i<0) \Longrightarrow y_{i+1}=y_i$

$d_{i+1}=d_i+2\Delta y$

Finally, we calculate d_1

$d_1=\Delta x[2m(x_1+1)+2b-2y_1-1]$

$d_1=\Delta x[2(mx_1+b-y_1)+2m-1]$

Since $mx_1+b-y_i=0$ and $m = \frac{\Delta y}{\Delta x}$, we have

$d_1=2\Delta y-\Delta x$

Advantage:

1. It involves only integer arithmetic, so it is simple.

2. It avoids the generation of duplicate points.

3. It can be implemented using hardware because it does not use multiplication and division.

4. It is faster as compared to DDA (Digital Differential Analyzer) because it does not involve floating point calculations like DDA Algorithm.

Disadvantage:

1. This algorithm is meant for basic line drawing only Initializing is not a part of Bresenham's line algorithm. So to draw smooth lines, We should want to look into a different algorithm.

Bresenham's Line Algorithm:

Step1: Start Algorithm

Step2: Declare variable $x_1,x_2,y_1,y_2,d,i_1,i_2,dx,dy$

DIWAKAR EDUCATION HUB

Step3: Enter value of x_1,y_1,x_2,y_2

Where $x_1,y_1$are coordinates of starting point

And x_2,y_2 are coordinates of Ending point

Step4: Calculate $dx = x_2-x_1$

Calculate $dy = y_2-y_1$

Calculate $i_1=2*dy$

Calculate $i_2=2*(dy-dx)$

Calculate $d=i_1-dx$

Step5: Consider (x, y) as starting point and x_{end}as maximum possible value of x.

If $dx < 0$

Then $x = x_2$

$y = y_2$

$x_{end}=x_1$

If $dx > 0$

Then $x = x_1$

$y = y_1$

$x_{end}=x_2$

Step6: Generate point at (x,y)coordinates.

Step7: Check if whole line is generated.

If $x >= x_{end}$

Stop.

Step8: Calculate co-ordinates of the next pixel

If $d < 0$

Then $d = d + i_1$

If $d \geq 0$

Then $d = d + i_2$

Increment $y = y + 1$

Step9: Increment $x = x + 1$

DIWAKAR EDUCATION HUB

Step10: Draw a point of latest (x, y) coordinates

Step11: Go to step 7

Step12: End of Algorithm

Example: Starting and Ending position of the line are (1, 1) and (8, 5). Find intermediate points.

Solution: $x_1=1$

$y_1=1$

$x_2=8$

$y_2=5$

$dx= x_2-x_1=8-1=7$

$dy=y_2-y_1=5-1=4$

$I_1=2* \Delta y=2*4=8$

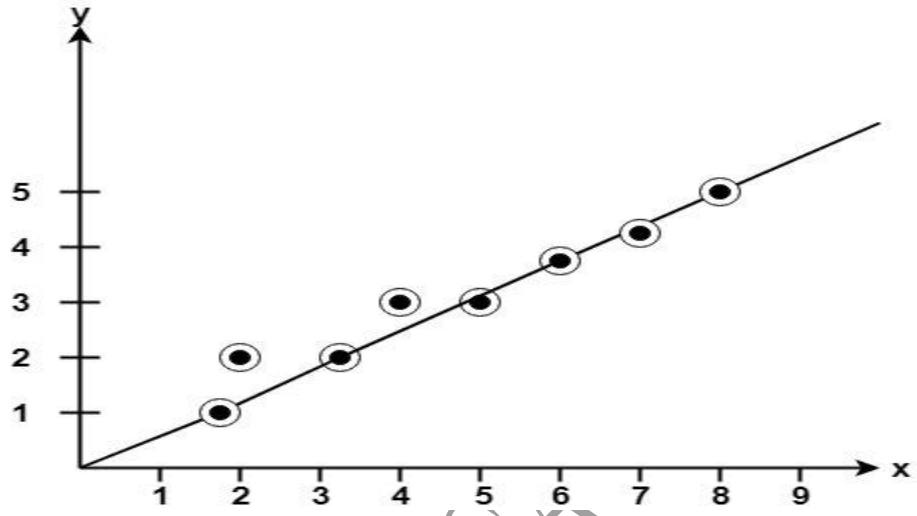
$I_2=2*(\Delta y-\Delta x)=2*(4-7)=-6$

$d = I_1-\Delta x=8-7=1$

x	y	d=d+I_1 or I_2
1	1	d+I_2=1+(-6)=-5
2	2	d+I_1=-5+8=3
3	2	d+I_2=3+(-6)=-3
4	3	d+I_1=-3+8=5
5	3	d+I_2=5+(-6)=-1
6	4	d+I_1=-1+8=7

DUCATION HUB

7	4	$d+I_2=7+(-6)=1$
8	5	



Program to implement Bresenham's Line Drawing Algorithm:

```
#include<stdio.h>
#include<graphics.h>
void drawline(int x0, int y0, int x1, int y1)
{
   int dx, dy, p, x, y;
   dx=x1-x0;
   dy=y1-y0;
   x=x0;
   y=y0;
   p=2*dy-dx;
   while(x<x1)
```

DIWAKAR EDUCATION

```
    {
        if(p>=0)
        {
            putpixel(x,y,7);
            y=y+1;
            p=p+2*dy-2*dx;
        }
        else
        {
            putpixel(x,y,7);
            p=p+2*dy;}
            x=x+1;
        }
}
int main()
{
    int gdriver=DETECT, gmode, error, x0, y0, x1, y1;
    initgraph(&gdriver, &gmode, "c:\\turboc3\\bgi");
    printf("Enter co-ordinates of first point: ");
    scanf("%d%d", &x0, &y0);
    printf("Enter co-ordinates of second point: ");
```

DIWAKAR EDUCATION HUB

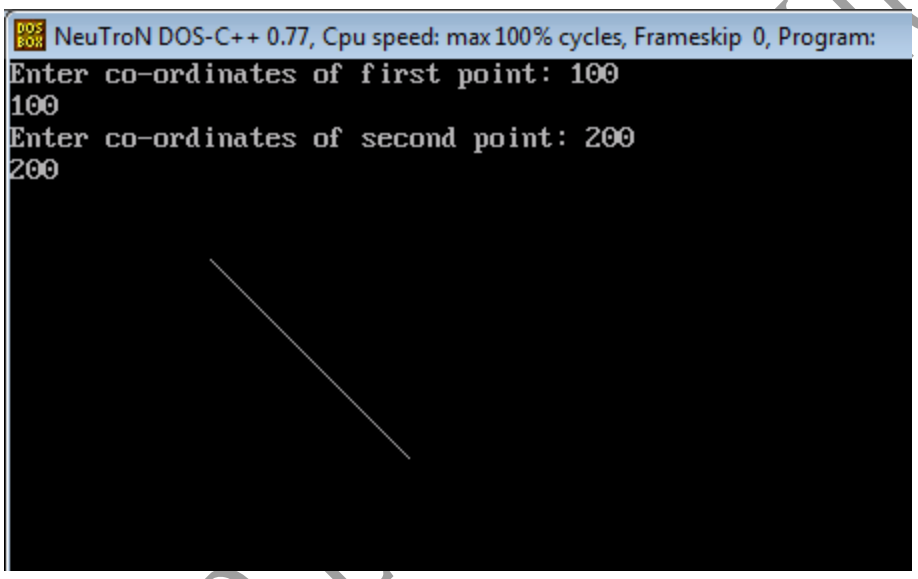
```
    scanf("%d%d", &x1, &y1);

    drawline(x0, y0, x1, y1);

    return 0;

}
```

Output:

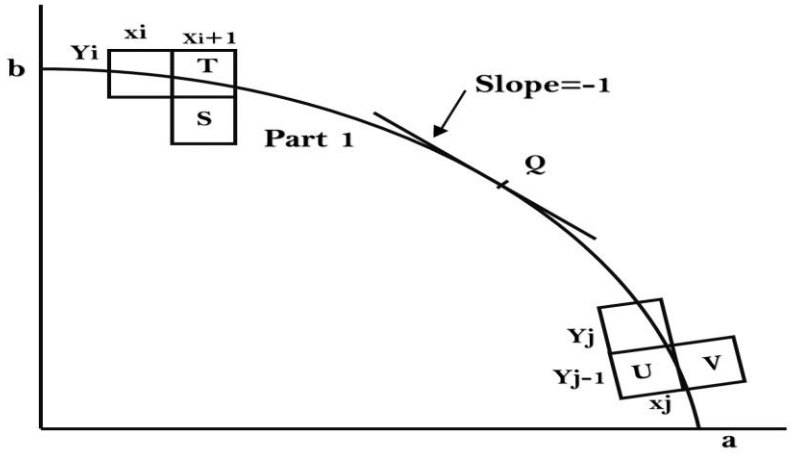


Midpoint Ellipse Algorithm:

This is an incremental method for scan converting an ellipse that is centered at the origin in standard position i.e., with the major and minor axis parallel to coordinate system axis. It is very similar to the midpoint circle algorithm. Because of the four-way symmetry property we need to consider the entire elliptical curve in the first quadrant.

first rewrite the ellipse equation and define the function f that can be used to decide if the midpoint between two candidate pixels is inside or outside the ellipse:

$$f(x, y) = b^2x^2 + a^2y^2 - a^2b^2 = \begin{bmatrix} < 0(x,y)\text{inside} \\ o\ (x,y)\text{on} \\ > 0\ (x,y)\text{outside} \end{bmatrix}$$



Now divide the elliptical curve from (0, b) to (a, 0) into two parts at point Q where the slope of the curve is -1.

Slope of the curve is defined by the f(x, y) = 0 is $\frac{dy}{dx} = -\frac{fx}{fy}$ where fx & fy are partial derivatives of f(x, y) with respect to x & y.

We have $fx = 2b^2 x$, $fy = 2a^2 y$ & $\frac{dy}{dx} = -\frac{2b^2x}{2a^2y}$ Hence we can monitor the slope value during the scan conversion process to detect Q. Our starting point is (0, b)

Suppose that the coordinates of the last scan converted pixel upon entering step i are (x_i, y_i). We are to select either T $(x_{i+1}), y_i)$ or S (x_{i+1}, y_{i-1}) to be the next pixel. The midpoint of T & S is used to define the following decision parameter.

$pi = f(x_{i+1}), y_i - \frac{1}{2})$

$pi = b^2 (x_{i+1})^2 + a^2 (y_i - \frac{1}{2})^2 - a^2 b^2$

If pi<0, the midpoint is inside the curve and we choose pixel T.

DIWAKAR EDUCA

If pi>0, the midpoint is outside or on the curve and we choose pixel S.

Decision parameter for the next step is:

$p_{i+1}=f(x_{i+1}+1,y_{i+1}-\frac{1}{2})$

$= b^2 (x_{i+1}+1)^2+a^2 (y_{i+1}-\frac{1}{2})^2-a^2 b^2$

Since $x_{i+1}=xi+1$,we have

$p_{i+1}-pi=b^2[((x_{i+1}+1)^2+a^2 (y_{i+1}-\frac{1}{2})^2-(y_i-\frac{1}{2})^2]$

$p_{i+1}= pi+2b^2 x_{i+1}+b^2+a^2 [(y_{i+1}-\frac{1}{2})^2-(y_i-\frac{1}{2})^2]$

If T is chosen pixel (pi<0), we have $y_{i+1}=yi$.

If S is chosen pixel (pi>0) we have $y_{i+1}=yi-1$. Thus we can express

p_{i+1}in terms of pi and (x_{i+1},y_{i+1}): $p_{i+1}= pi+2b^2 x_{i+1}+b^2$ if pi<0 = $pi+2b^2 x_{i+1}+b^2-2a^2 y_{i+1}$ if pi>0

The initial value for the recursive expression can be obtained by the evaluating the original definition of pi with (0, b):

$p1 = (b^2+a^2 (b-\frac{1}{2})^2-a^2 b^2$

$= b^2-a^2 b+a^2/4$

Suppose the pixel $(x_j\ y_j)$ has just been scan converted upon entering step j. The next pixel is either U (x_j ,y_j-1) or V (x_j+1,y_j-1). The midpoint of the horizontal line connecting U & V is used to define the decision parameter:

$q_j=f(x_j+\frac{1}{2},y_j-1)$

$q_j=b^2 (x_j+\frac{1}{2})^2+a^2 (y_j -1)^2-a^2 b^2$

If q_j<0, the midpoint is inside the curve and we choose pixel V.

DIWAKAR EDUCATION HUB

If $q_j \geq 0$, the midpoint is outside the curve and we choose pixel U.Decision parameter for the next step is:

$q_{j+1}=f(x_{j+1}+\frac{1}{2},y_{j+1}-1)$

$= b^2 (x_{j+1}+\frac{1}{2})^2+ a^2 (y_{j+1}-1)^2- a^2 b^2$

Since $y_{j+1}=y_j-1$,we have

$q_{j+1}-q_j=b^2 [(x_{j+1}+\frac{1}{2})^2-(x_j + \frac{1}{2})^2]+a^2 (y_{j+1}-1)^2-(y_{j+1})^2]$

$q_{j+1}=q_j+b^2 [(x_{j+1}+\frac{1}{2})^2-(x_j + \frac{1}{2})^2]-2a^2 y_{j+1}+a^2$

If V is chosen pixel (qj<0), we have $x_{j+1}=xj$.

If U is chosen pixel (pi>0) we have $x_{j+1}=xj$. Thus we can express

q_{j+1}in terms of q_j and (x_{j+1},y_{j+1}):

$q_{j+1}=q_j+2b^2 x_{j+1}-2a^2 y_{j+1}+a^2$ if qj < 0

$=q_j-2a^2 y_{j+1}+a^2$ if qj>0

The initial value for the recursive expression is computed using the original definition of qj. And the coordinates of $(x_k\ y_k)$ of the last pixel choosen for the part 1 of the curve:

$q1 = f(x_k+\frac{1}{2},y_k-1)=b^2 (x_k+\frac{1}{2})^2-a^2 (y_k-1)^2- a^2 b^2$

Algorithm:

int x=0, y=b; [starting point]

int fx=0, fy=$2a^2$ b [initial partial derivatives]

int p = b^2-a^2 b+$a^2/4$

while (fx<="" 1="" {="" set="" pixel="" (x,="" y)="" x++;="" fx="fx" +="" $2b^{2;}$

if (p<0)

DIWAKAR EDUCATION HUB

```
        p = p + fx +b2;
        else
        {
                y--;
                fy=fy-2a2
                p = p + fx +b2-fy;
        }
}
Setpixel (x, y);
p=b2(x+0.5)2+ a2 (y-1)2- a2 b2
while (y>0)
{
        y--;
        fy=fy-2a2;
        if (p>=0)
        p=p-fy+a2
        else
        {
                x++;
                fx=fx+2b2
                p=p+fx-fy+a2;
```

DIWAKAR EDUCATION HUB

```
        }
        Setpixel (x,y);
}
```

Transformations

Computer Graphics provide the facility of viewing object from different angles. The architect can study building from different angles i.e.

1. Front Evaluation
2. Side elevation
3. Top plan

A Cartographer can change the size of charts and topographical maps. So if graphics images are coded as numbers, the numbers can be stored in memory. These numbers are modified by mathematical operations called as Transformation.

The purpose of using computers for drawing is to provide facility to user to view the object from different angles, enlarging or reducing the scale or shape of object called as Transformation.

Two essential aspects of transformation are given below:

1. Each transformation is a single entity. It can be denoted by a unique name or symbol.
2. It is possible to combine two transformations, after connecting a single transformation is obtained, e.g., A is a transformation for translation. The B transformation performs scaling. The combination of two is C=AB. So C is obtained by concatenation property.

There are two complementary points of view for describing object transformation.

DIWAKAR EDUCATION HUB

1. Geometric Transformation: The object itself is transformed relative to the coordinate system or background. The mathematical statement of this viewpoint is defined by geometric transformations applied to each point of the object.
2. Coordinate Transformation: The object is held stationary while the coordinate system is transformed relative to the object. This effect is attained through the application of coordinate transformations.

An example that helps to distinguish these two viewpoints:

The movement of an automobile against a scenic background we can simulate this by

- Moving the automobile while keeping the background fixed-(Geometric Transformation)
- We can keep the car fixed while moving the background scenery-(Coordinate Transformation)

Types of Transformations:

1. Translation
2. Scaling
3. Rotating
4. Reflection
5. Shearing

Note: Translation, Scaling, and Rotation are also called as Basic Transformations.

Translation

DIWAKAR EDUCATION HUB

It is the straight line movement of an object from one position to another is called Translation. Here the object is positioned from one coordinate location to another.

Translation of point:

To translate a point from coordinate position (x, y) to another (x_1 y_1), we add algebraically the translation distances T_x and T_y to original coordinate.

$x_1=x+T_x$

$y_1=y+T_y$

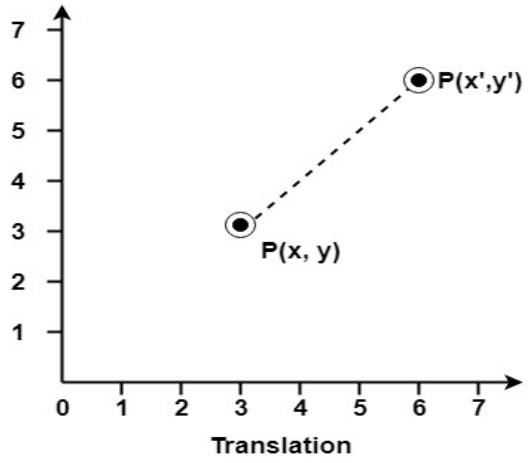
The translation pair (T_x,T_y) is called as shift vector.

Translation is a movement of objects without deformation. Every position or point is translated by the same amount. When the straight line is translated, then it will be drawn using endpoints.

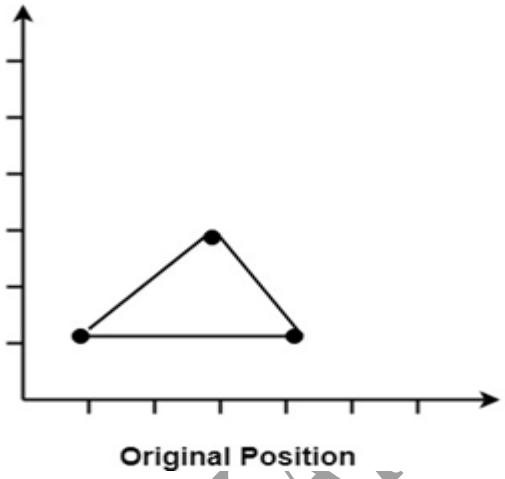
For translating polygon, each vertex of the polygon is converted to a new position. Similarly, curved objects are translated. To change the position of the circle or ellipse its center coordinates are transformed, then the object is drawn using new coordinates.

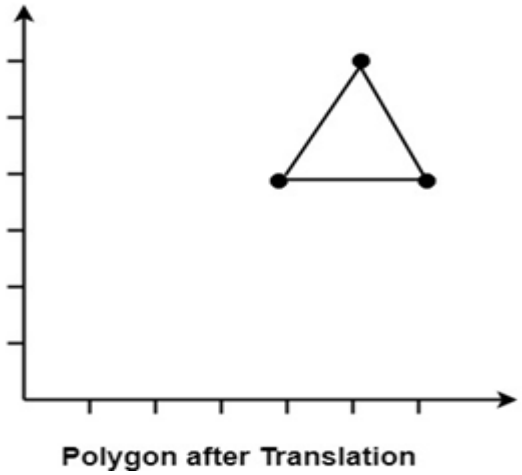
Let P is a point with coordinates (x, y). It will be translated as (x^1 y^1).

DIWAKAR EDUCATION HUB



Translation of Polygon





Matrix for Translation:

$$\begin{vmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{vmatrix} \quad \text{Or} \quad \begin{vmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{vmatrix}$$

Scaling:

It is used to alter or change the size of objects. The change is done using scaling factors. There are two scaling factors, i.e. S_x in x direction S_y in y-direction. If the

original position is x and y. Scaling factors are S_x and S_y then the value of coordinates after scaling will be x^1 and y_1.

If the picture to be enlarged to twice its original size then $S_x = S_y = 2$. If $S_x$and S_y are not equal then scaling will occur but it will elongate or distort the picture.

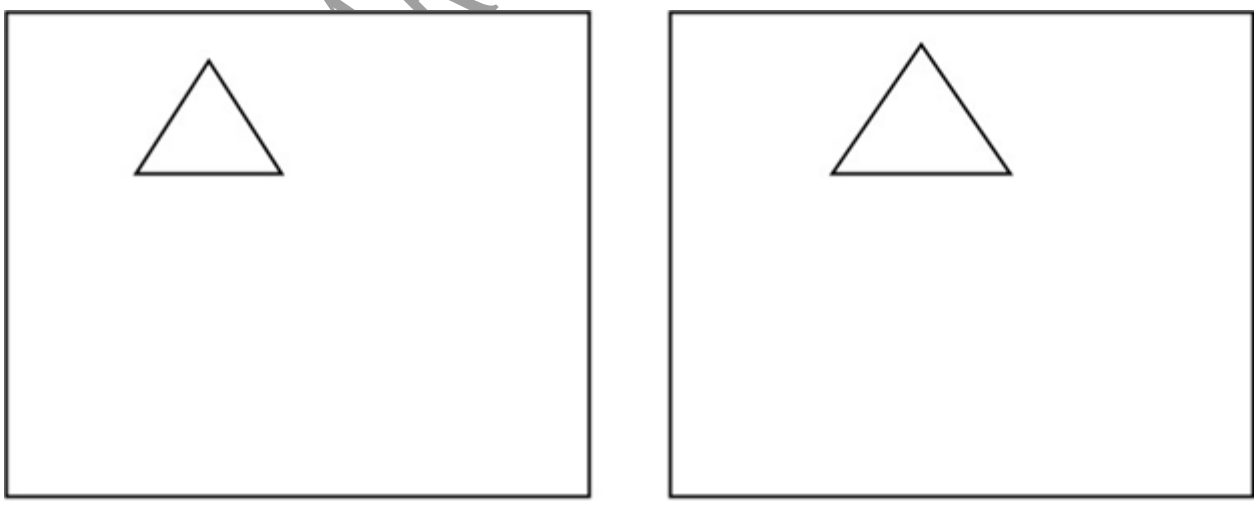
If scaling factors are less than one, then the size of the object will be reduced. If scaling factors are higher than one, then the size of the object will be enlarged.

If $S_x$and $S_y$are equal it is also called as Uniform Scaling. If not equal then called as Differential Scaling. If scaling factors with values less than one will move the object closer to coordinate origin, while a value higher than one will move coordinate position farther from origin.

Enlargement: If $T_1= \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$,If $(x_1\ y_1)$is original position and $T_1$is translation vector then $(x_2\ y_2)$ are coordinated after scaling

$$[x_2 y_2] = [\ x_1 y_1] \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} = [2x_1\ 2y_1]$$

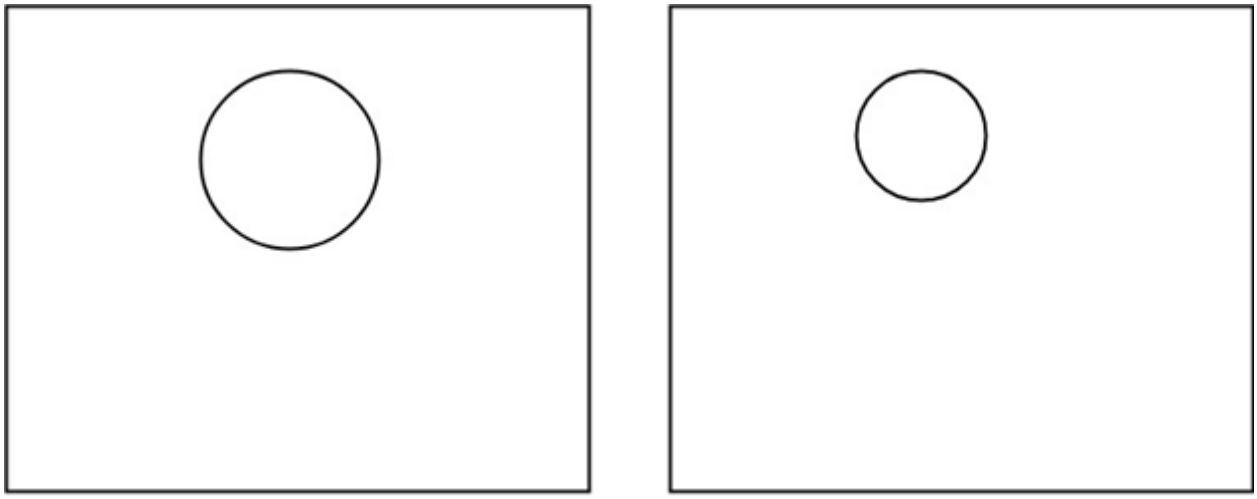
The image will be enlarged two times



Original Image Enlarged Image

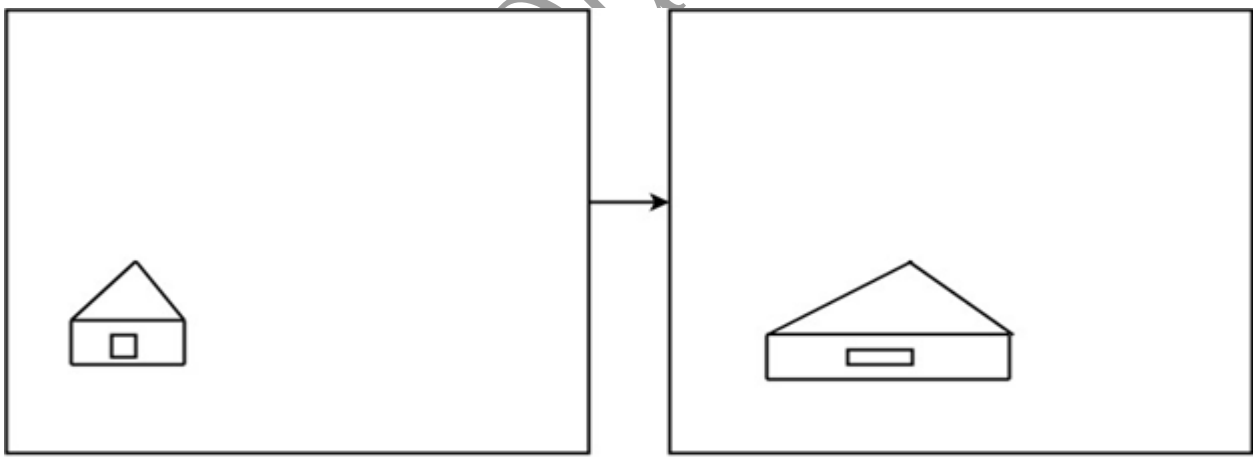
Reduction: If $T_1 = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}$. If $(x_1\ y_1)$ is original position and T_1 is translation vector, then $(x_2\ y_2)$ are coordinates after scaling

$$[x_2 y_2] = [\ x_1 y_1] \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix} = [.5x_1 . 5y_1]$$



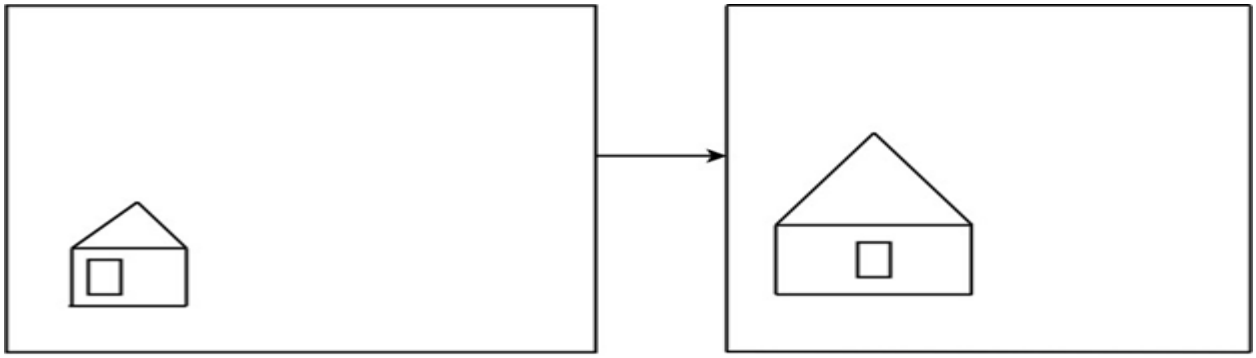
Original Image

Reduction Image



Original Object

Object after scaling in X direction



Original Object

Object after scaling in Y direction

Matrix for Scaling:

$$S = \begin{vmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

Example: Prove that 2D Scaling transformations are commutative i.e, $S_1 S_2 = S_2 S_1$.

Solution: S_1 and S_2 are scaling matrices

DIWAKAR EDUCATION

$$S_1 = \begin{bmatrix} Sx_1 & 0 & 0 \\ 0 & Sy_1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$S_2 = \begin{bmatrix} Sx_2 & 0 & 0 \\ 0 & Sy_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$S_1 * S_2 = \begin{bmatrix} Sx_1 & 0 & 0 \\ 0 & Sy_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} Sx_2 & 0 & 0 \\ 0 & Sy_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} Sx_1Sx_2 & 0 & 0 \\ 0 & Sy_1Sy_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{.....equation 1}$$

$$S_2 * S_1 = \begin{bmatrix} Sx_2 & 0 & 0 \\ 0 & Sy_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} Sx_1 & 0 & 0 \\ 0 & Sy_1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} Sx_2Sx_1 & 0 & 0 \\ 0 & Sy_2Sy_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{.....equation 2}$$

From equation 1 and 2

$$S_1S_2 = S_2S_1. \text{ Hence Proved}$$

Rotation:

It is a process of changing the angle of the object. Rotation can be clockwise or anticlockwise. For rotation, we have to specify the angle of rotation and rotation point. Rotation point is also called a pivot point. It is print about which object is rotated.

Types of Rotation:

1. Anticlockwise
2. Counterclockwise

The positive value of the pivot point (rotation angle) rotates an object in a counter-clockwise (anti-clockwise) direction.

DIWAKAR EL

The negative value of the pivot point (rotation angle) rotates an object in a clockwise direction.

When the object is rotated, then every point of the object is rotated by the same angle.

Straight Line: Straight Line is rotated by the endpoints with the same angle and redrawing the line between new endpoints.

Polygon: Polygon is rotated by shifting every vertex using the same rotational angle.

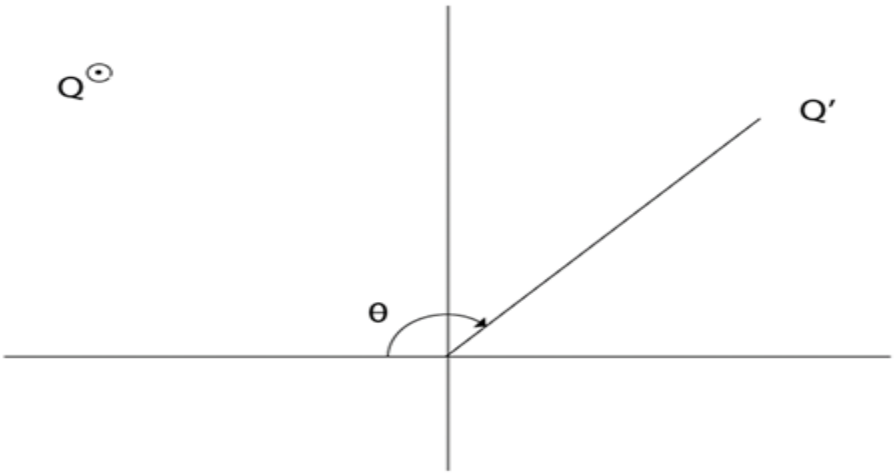
Curved Lines: Curved Lines are rotated by repositioning of all points and drawing of the curve at new positions.

Circle: It can be obtained by center position by the specified angle.

Ellipse: Its rotation can be obtained by rotating major and minor axis of an ellipse by the desired angle.

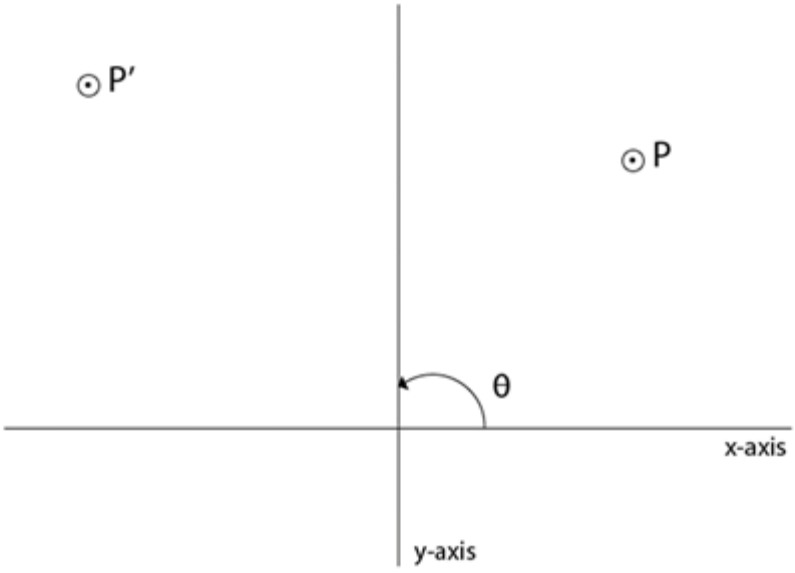
DIWAKAR EDU

Rotation in anticlockwise direction



Q is original position
Q ' is final rotated position

Rotation of P in clockwise direction



P is original Position
P' is final position or position after rotation
where θ is angle of rotation

Matrix for rotation is a clockwise direction.

$$R=\begin{bmatrix}\cos\theta & \sin\theta \\ \sin\theta & \cos\theta\end{bmatrix}$$

Matrix for rotation is an anticlockwise direction.

$$R=\begin{bmatrix}\cos\theta & \sin\theta \\ -\sin\theta & \cos\theta\end{bmatrix}$$

Matrix for homogeneous co-ordinate rotation (clockwise)

$$R=\begin{pmatrix}\cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1\end{pmatrix}$$

Matrix for homogeneous co-ordinate rotation (anticlockwise)

$$R=\begin{pmatrix}\cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1\end{pmatrix}$$

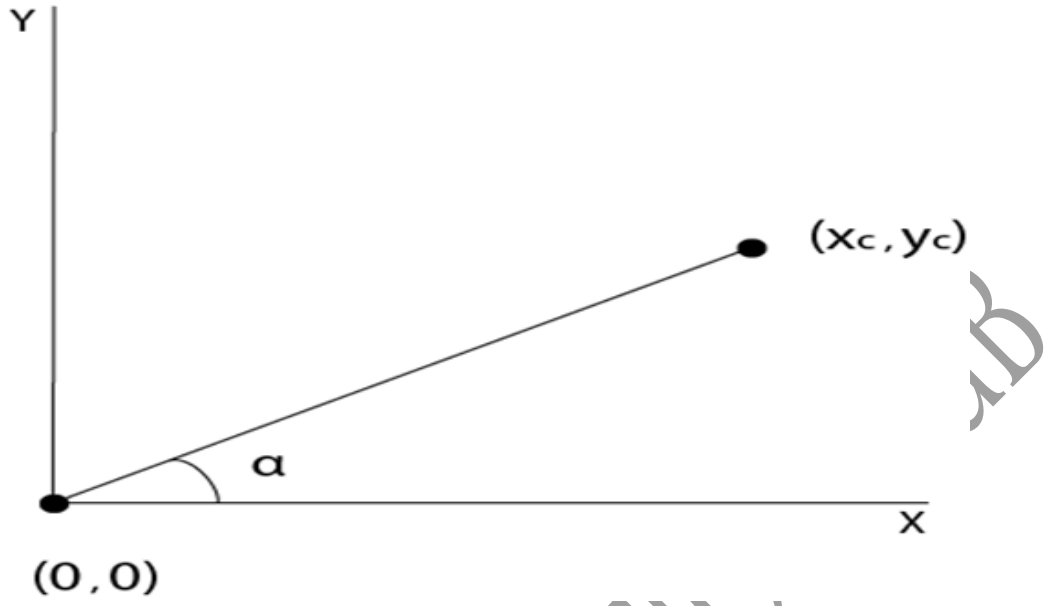
Rotation about an arbitrary point: If we want to rotate an object or point about an arbitrary point, first of all, we translate the point about which we want to rotate to the origin. Then rotate point or object about the origin, and at the end, we again translate it to the original place. We get rotation about an arbitrary point.

Example: The point (x, y) is to be rotated

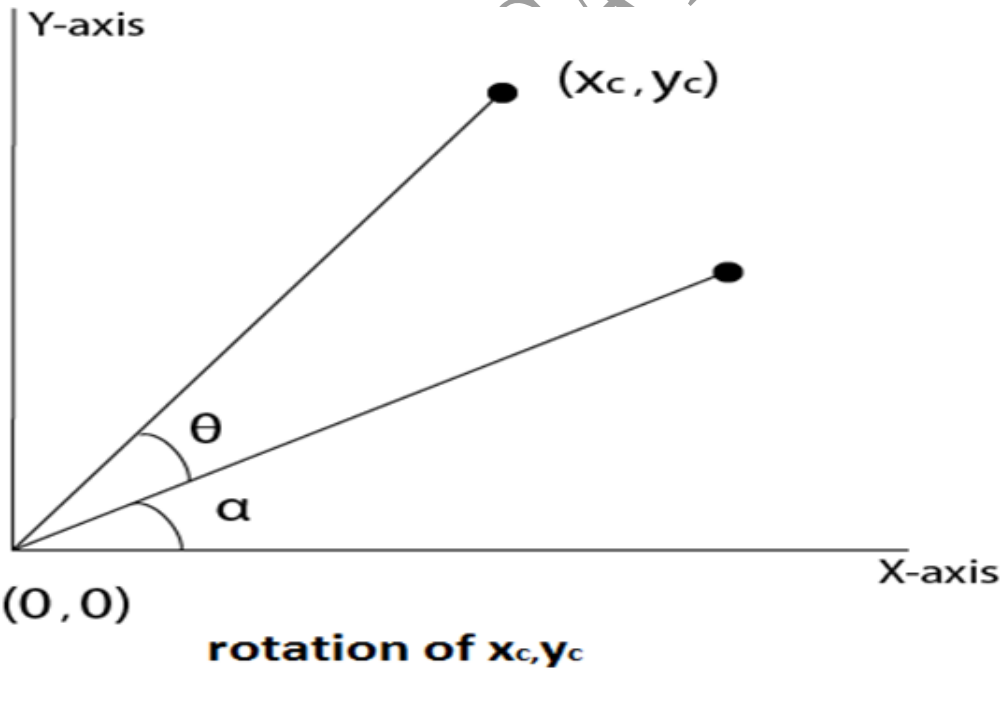
The $(x_c\ y_c)$ is a point about which counterclockwise rotation is done

Step1: Translate point $(x_c\ y_c)$ to origin

EDUCATION HUB

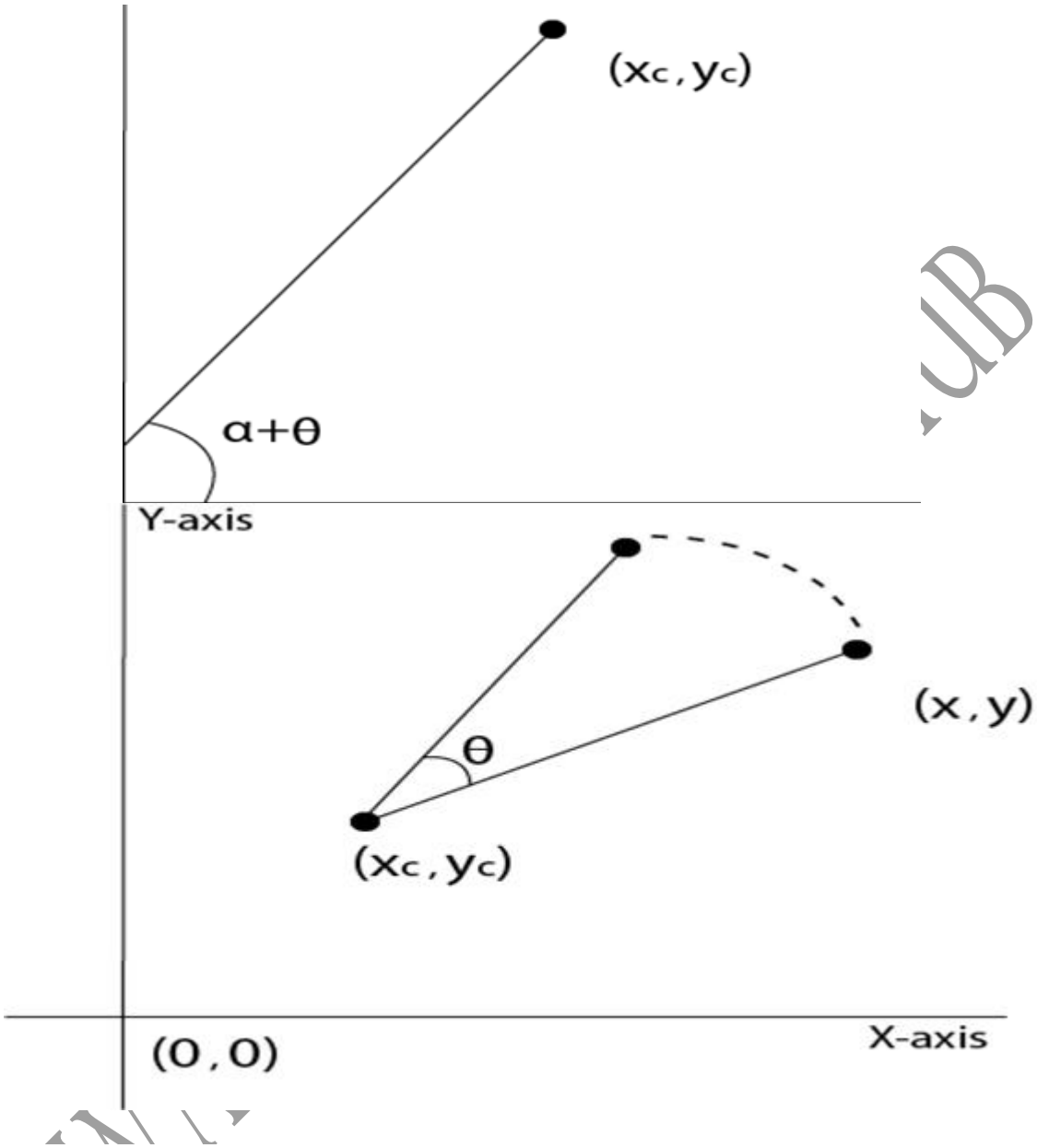


Step2: Rotation of (x, y) about the origin



rotation of x_c,y_c

Step3: Translation of center of rotation back to its original position



Example1: Prove that 2D rotations about the origin are commutative i.e. $R_1 R_2 = R_2 R_1$.

Solution: R_1 and R_2 are rotation matrices

$$R_1 = \begin{bmatrix} \cos\theta_1 & \sin\theta_1 & 0 \\ -\sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R_2 = \begin{bmatrix} \cos\theta_2 & \sin\theta_2 & 0 \\ -\sin\theta_2 & \cos\theta_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R_1 * R_2 = \begin{bmatrix} \cos\theta_1 & \sin\theta_1 & 0 \\ -\sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta_2 & \sin\theta_2 & 0 \\ -\sin\theta_2 & \cos\theta_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos\theta_1 \cos\theta_2 - \sin\theta_1 \sin\theta_2 & \cos\theta_1\sin\theta_2 + \sin\theta_1\cos\theta_2 & 0 \\ -\sin\theta_1\cos\theta_2 - \cos\theta_1 \sin\theta_2 & -\sin\theta_1\sin\theta_2 + \cos\theta_1\cos\theta_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \ldots \text{eq 1}$$

$$R_2 * R_1 = \begin{bmatrix} \cos\theta_2 & \sin\theta_2 & 0 \\ -\sin\theta_2 & \cos\theta_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta_1 & \sin\theta_1 & 0 \\ -\sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos\theta_2 \cos\theta_1 - \sin\theta_2 \sin\theta_2 & \cos\theta_2\sin\theta_1 + \sin\theta_2\cos\theta_1 & 0 \\ -\sin\theta_2\cos\theta_1 - \cos\theta_2 \sin\theta_1 & -\sin\theta_2\sin\theta_2 + \cos\theta_2\cos\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \ldots \text{eq 2}$$

From eq 1 & eq 2

$R_1R_2 = R_2R_1$. Hence Proved.

Example2: Rotate a line CD whose endpoints are (3, 4) and (12, 15) about origin through a 45° anticlockwise direction.

Solution: The point C (3, 4)

$$R = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

$$\theta = 45°$$

Let

$$R = \begin{bmatrix} \cos45° & \sin45° \\ -\sin45° & \cos45° \end{bmatrix}$$

$$R = \begin{bmatrix} 0.707 & 0.707 \\ -0.707 & -0.707 \end{bmatrix}$$

The point A (3, 4) after rotation will be

$$[x, y] = [3, 4] \begin{bmatrix} 0.707 & 0.707 \\ -0.707 & -0.707 \end{bmatrix}$$

$$= [3 * 0.707 - 4 * 0.707 \quad 3 * 0.707 + 4 * 0.707]$$

[2.121-2.8282.21+2.828]

[-.707 4.949]

The rotation of point B (12, 15)

$$R_1 = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

$$= \begin{bmatrix} 0.707 & 0.707 \\ -0.707 & -0.707 \end{bmatrix}$$

The point B (12, 15) will be

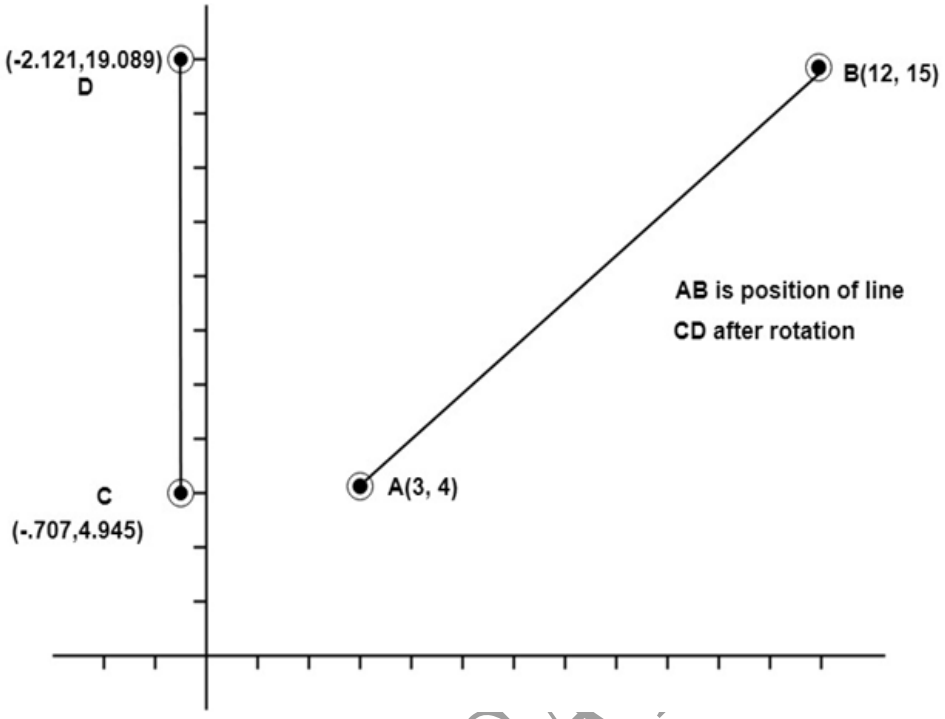
$$[x, y] = [12, 15] \begin{bmatrix} 0.707 & 0.707 \\ -0.707 & -0.707 \end{bmatrix}$$

$$= [12 * 0.707 + 15 * 0.707 \quad 12 * 0.707 + 15 * 0.707]$$

$$= [(8.484\text{-}10.605) \; (8.484 + 10.605)]$$

$$= [-2.121 \quad 19.089]$$

So line AB after rotation at 45°become [.707, 4.945] and [-2.121, 19.089]



Example3: Rotate line AB whose endpoints are A (2, 5) and B (6, 12) about origin through a 30° clockwise direction.

Solution: For rotation in the clockwise direction. The matrix is

$$R = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

Step1: Rotation of point A (2, 5). Take angle 30°

$$R = \begin{bmatrix} \cos 30° & -\sin 30° \\ \sin 30° & \cos 30° \end{bmatrix}$$

$$= [2, 5] \begin{bmatrix} \cos 30° & -\sin 30° \\ -\sin 30° & \cos 30° \end{bmatrix}$$

$$= [2, 5] \begin{bmatrix} .866 & -0.5 \\ .5 & .866 \end{bmatrix}$$

$$= [2 * .866 + 5 * .5 \qquad 2 * (-.5) + 5\ (.866)]$$

$$= [(1.732 + 2.5 \qquad -1 + 4.33]$$

$$= [4.232 \qquad 3.33]$$

A point (2, 5) become (4.232, 3.33)

Step2: Rotation of point B (6, 12)

$$= [6 \;\; 12] \begin{bmatrix} \cos 30° & -\sin 30° \\ \sin 30° & \cos 30° \end{bmatrix}$$

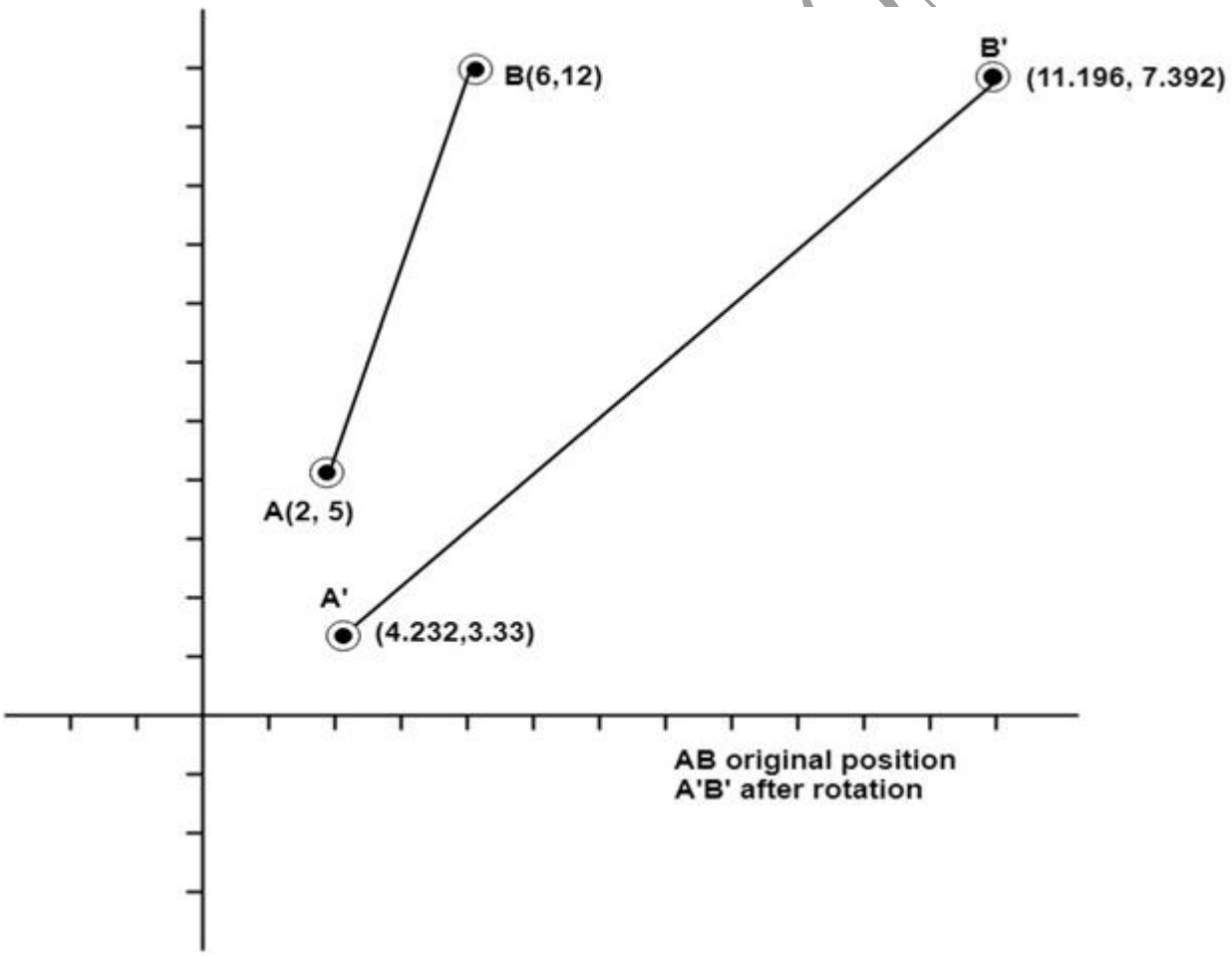
$$= [6 \; 12] \begin{bmatrix} .866 & -0.5 \\ .5 & .866 \end{bmatrix}$$

$$= [6 * .866 + 12 * .5 \qquad 6 * (-.5) + 12 * (.866)]$$

$$= [5.196 + 6 \qquad -3 + 10.392]$$

$$= [11.196 \qquad 7.392]$$

B point (6, 12) becomes (11.46, 7.312) after rotation 30° in clockwise direction.



Reflection:

It is a transformation which produces a mirror image of an object. The mirror image can be either about x-axis or y-axis. The object is rotated by180°.

Types of Reflection:

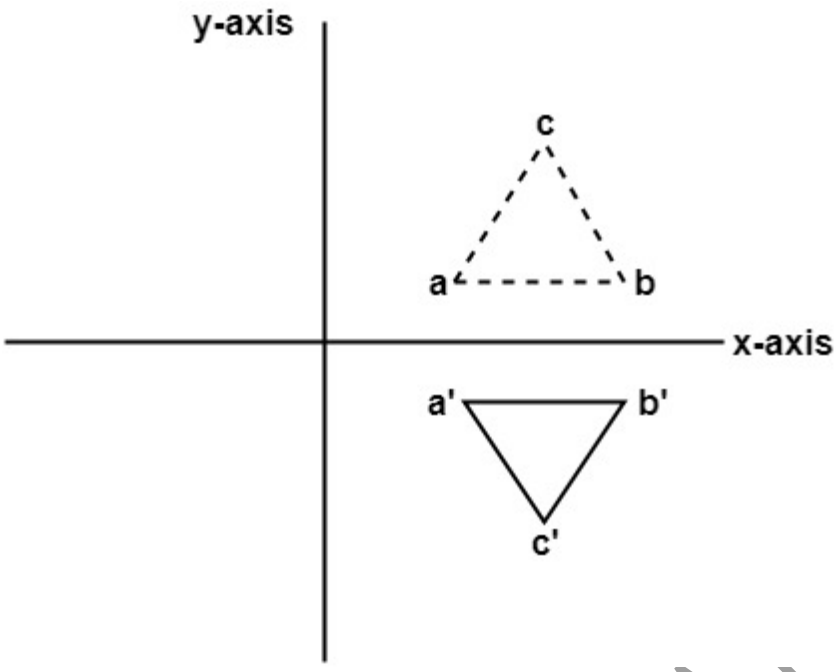
1. Reflection about the x-axis
2. Reflection about the y-axis
3. Reflection about an axis perpendicular to xy plane and passing through the origin
4. Reflection about line y=x

1. Reflection about x-axis: The object can be reflected about x-axis with the help of the following matrix

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

In this transformation value of x will remain same whereas the value of y will become negative. Following figures shows the reflection of the object axis. The object will lie another side of the x-axis.

DIWAKAR EDUCATION HUB



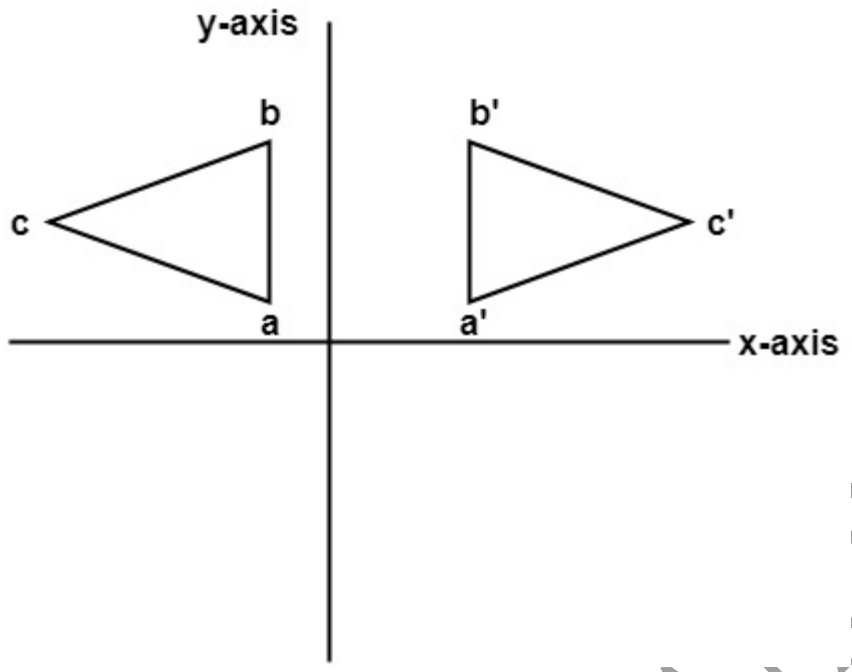
2. Reflection about y-axis: The object can be reflected about y-axis with the help of following transformation matrix

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Here the values of x will be reversed, whereas the value of y will remain the same. The object will lie another side of the y-axis.

The following figure shows the reflection about the y-axis

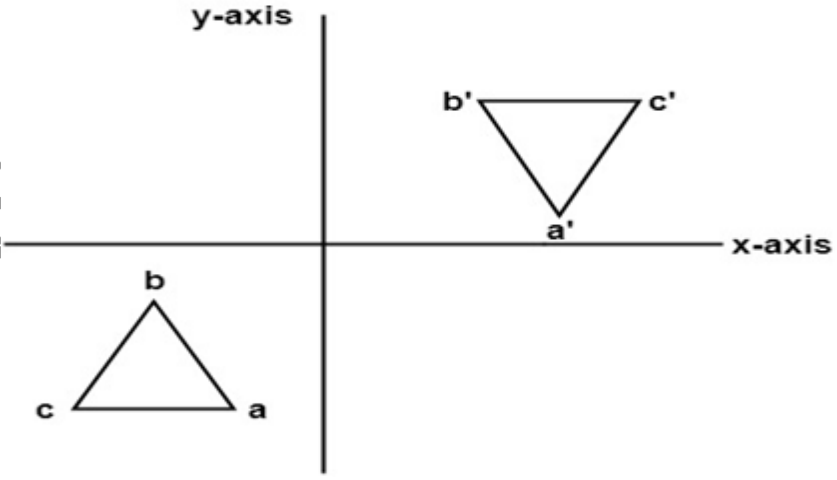
DIWAKAR EDUCATION HUB



3. Reflection about an axis perpendicular to xy plane and passing through origin:

In the matrix of this transformation is given below

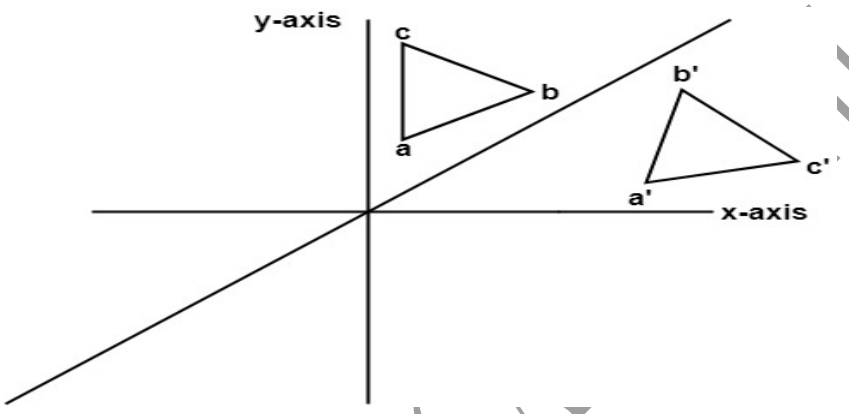
$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



In this value of x and y both will be reversed. This is also called as half revolution about the origin.

4. Reflection about line y=x: The object may be reflected about line y = x with the help of following transformation matrix

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



First of all, the object is rotated at 45°. The direction of rotation is clockwise. After it reflection is done concerning x-axis. The last step is the rotation of y=x back to its original position that is counterclockwise at 45°.

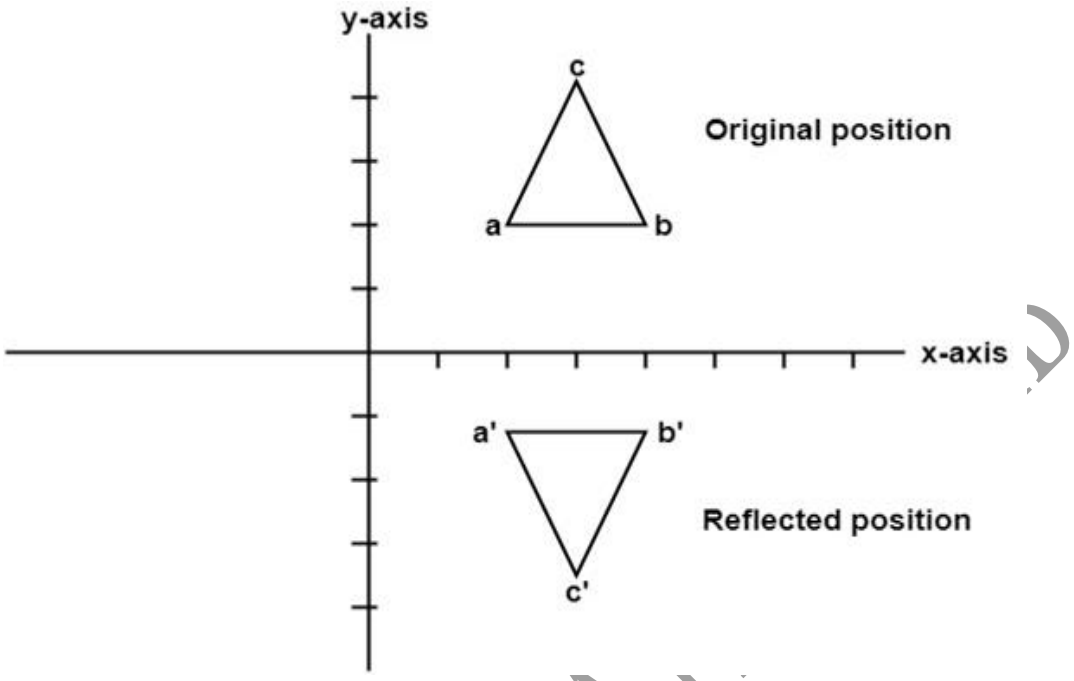
Example: A triangle ABC is given. The coordinates of A, B, C are given as

A (3 4)
B (6 4)
C (4 8)

Find reflected position of triangle i.e., to the x-axis.

Solution:

DIWAKAR EDUCATION HUB



The matrix for reflection about x axis $\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

The a point coordinates after reflection

$$(x, y) = (3, 4) \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$(x, y) = [3, -4]$$

The b point coordinates after reflection

$$(x, y) = (6, 4) \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$(x, y) = [6, -4]$$

The coordinate of point c after reflection

DIWAKAR EDUCATION

$$(x, y) = (4, 8)\begin{bmatrix}1 & 0 & 0\\ 0 & -1 & 0\\ 0 & 0 & 1\end{bmatrix}$$

$$(x, y) = [4, -8]$$

a (3, 4) becomes a^1 (3, -4)
b (6, 4) becomes b^1 (6, -4)
c (4, 8) becomes c^1 (4, -8)

Program to perform Mirror Reflection about a line:

```
#include <iostream.h>
#include <conio.h>
#include <graphics.h>
#include <math.h>
#include <stdlib.h>
#define pi 3.14
class arc
{
   float x[10],y[10],theta,ref[10][10],ang;
      float p[10][10],p1[10][10],x1[10],y1[10],xm,ym;
   int i,k,j,n;
   public:
   void get();
   void cal ();
   void map ();
```

DIWAKAR EDUCATION HUB

```
    void graph ();
    void plot ();
    void plot1();
};
void arc::get ()
{
    cout<<"\n ENTER ANGLE OF LINE INCLINATION AND Y INTERCEPT";
    cin>> ang >> b;
    cout <<"\n ENTER NO OF VERTICES";
    cin >> n;
    cout <<"\n ENTER";
    for (i=0; i<n; i++)
    {
       cout<<"\n x["<<i<<"] and y["<<i<<"]";
    }
    theta =(ang * pi)/ 180;
    ref [0] [0] = cos (2 * theta);
    ref [0] [1] = sin (2 * theta);
    ref [0] [2] = -b *sin (2 * theta);
    ref [1] [0] = sin (2 * theta);
    ref [1] [1] = -cos (2 * theta);
```

DIWAKAR EDUCATION HUB

```
   ref [1] [2] = b * (cos (2 * theta)+1);
   ref [2] [0]=0;
   ref [2] [1]=0;
   ref [2] [2] = 1;
}
void arc :: cal ()
{
   for (i=0; i < n; i++)
   {
      p[0] [i] = x [i];
      p [1] [i] = y [i];
      p [2] [i] = 1;
   }
   for (i=0; i<3;i++)
   {
      for (j=0; j<n; j++)
      {
         p1 [i] [j]=0;
         for (k=0;k<3; k++)
      }
      p1 [i] [j] + = ref [i] [k] * p [k] [j];
```

DIWAKAR EDUCATION HUB

```
        }
for (i=0; i<n; i++)
  {
   x1 [i]=p1[0] [i];
   y1 [i] = p1 [1] [i];
   }
}
void arc :: map ()
{
   int gd = DETECT,gm;
   initgraph (&gd, &gm, " ");
        int errorcode = graphresult ();
   /* an error occurred */
   if (errorcode ! = grOK)
   {
      printf ("Graphics error: %s \n", grapherrormsg (errorcode));
      printf ("Press any key to halt:");
      getch ();
      exit (1); /* terminate with an error code */
   }
}
```

DIWAKAR EDUCATION HUB

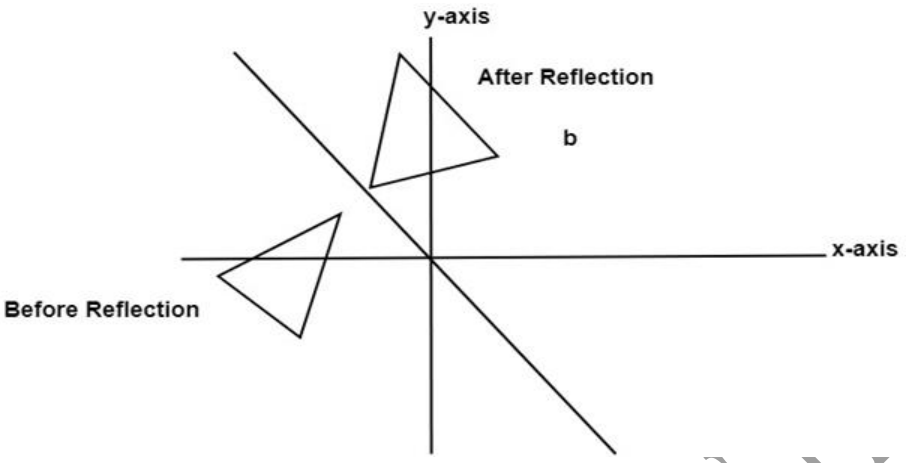
```
void arc :: graph ()
{
   xm=getmaxx ()/2;
   ym=getmaxy ()/2;
   line (xm, 0, xmm 2*ym);
}
void arc :: plot 1 ()
{
   for (i=0; i <n-1; i++)
   {
      circle (x1[i]+xm, (-y1[i]+ym), 2);
      line (x1[i]+xm, (-y1[i]+ym), x1[i+1]+xm, (-y1[i+1]+ym));
   }
      line (x1[n-1)+xm, (-y1[n-1]+ym), x1[0]+xm, (-y1[0]+ym));
      getch();
}
void arc :: plot ()
{
   for (i=0; i <n-1; i++)
   {
      circle (x1[i]+xm, (-y1[i]+ym, 2);
```

UWAKAR EDUCATION HUB

```
        line (x1[i]+xm, (-y1[i]+ym), x[i+1]+xm, (-y1[i+1]+ym));
    }
        line (x[n-1]+xm, (-y1[n-1]+ym), x[0]+xm, (-y[0]+ym));
        getch();
}
void main ()
{
    class arc a;
    clrscr();
    a.map();
    a.graph();
    a.get();
    a.cal();
    a.plot();
    a.plot1();
    getch();
}
```

Output:

DIWAKAR EDUCATION HUB

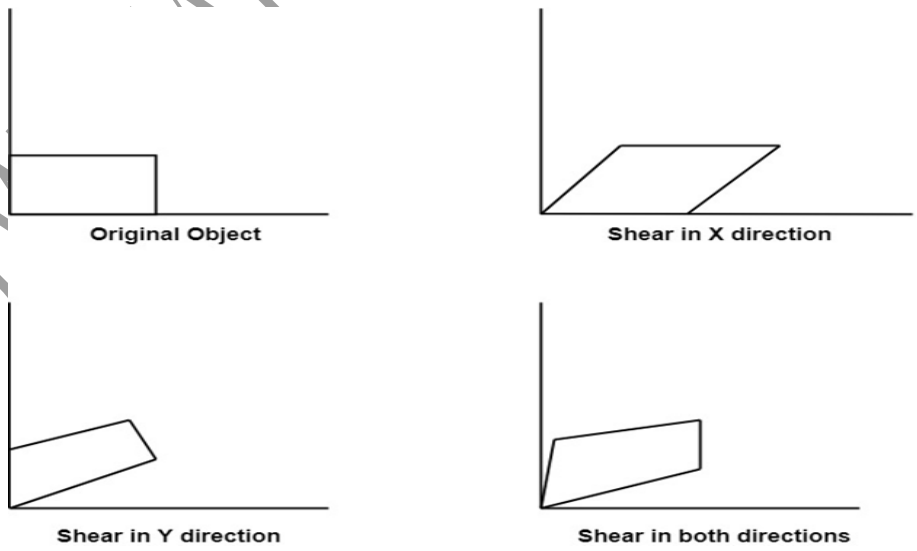


Shearing:

It is transformation which changes the shape of object. The sliding of layers of object occur. The shear can be in one direction or in two directions.

Shearing in the X-direction: In this horizontal shearing sliding of layers occur. The homogeneous matrix for shearing in the x-direction is shown below:

$$\begin{bmatrix} 1 & 0 & 0 \\ Sh_x & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



DIVYA EDUCATION HUB

Shearing in the Y-direction: Here shearing is done by sliding along vertical or y-axis.

$$\begin{bmatrix} 1 & Sh_y & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Shearing in X-Y directions: Here layers will be slided in both x as well as y direction. The sliding will be in horizontal as well as vertical direction. The shape of the object will be distorted. The matrix of shear in both directions is given by:

$$\begin{bmatrix} 1 & Sh_y & 0 \\ Sh_x & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Matrix Representation of 2D Transformation

DIWAKAR EDUCATION HUB

1. Scaling	$\begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix}$
2. Rotation (clockwise)	$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$
3. Rotation (anti-clock)	$\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$
4. Translation	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ t_x & t_y \end{bmatrix}$
5. Reflection (about x axis)	$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
6. Reflection (about y axis)	$\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$
7. Reflection (about origin)	$\begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$
8. Reflection about Y=X	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
9. Reflection about Y= −X	$\begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}$
10. Shearing in X direction	$\begin{bmatrix} 1 & 0 \\ Sh_x & 1 \end{bmatrix}$
11. Shearing in Y direction	$\begin{bmatrix} 1 & Sh_y \\ 0 & 1 \end{bmatrix}$
12. Shearing in both x and y direction	$\begin{bmatrix} 1 & Sh_y \\ Sh_x & 1 \end{bmatrix}$

Homogeneous Coordinates

The rotation of a point, straight line or an entire image on the screen, about a point other than origin, is achieved by first moving the image until the point of rotation occupies the origin, then performing rotation, then finally moving the image to its original position.

The moving of an image from one place to another in a straight line is called a translation. A translation may be done by adding or subtracting to each point, the amount, by which picture is required to be shifted.

Translation of point by the change of coordinate cannot be combined with other transformation by using simple matrix application. Such a combination is essential if we wish to rotate an image about a point other than origin by translation, rotation again translation.

To combine these three transformations into a single transformation, homogeneous coordinates are used. In homogeneous coordinate system, two-dimensional coordinate positions (x, y) are represented by triple-coordinates.

Homogeneous coordinates are generally used in design and construction applications. Here we perform translations, rotations, scaling to fit the picture into proper position.

Example of representing coordinates into a homogeneous coordinate system: For two-dimensional geometric transformation, we can choose homogeneous parameter h to any non-zero value. For our convenience take it as one. Each two-dimensional position is then represented with homogeneous coordinates (x, y, 1).

Following are matrix for two-dimensional transformation in homogeneous coordinate:

DIWAKAR EDUCATION HUB

1. Translation $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix}$ or $\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$

2. Scaling $\begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$

3. Rotation (clockwise) $\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$

4. Rotation (anti-clock) $\begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$

5. Reflection against X axis $\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

6. Reflection against Y axis $\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

7. Reflection against origin $\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

8. Reflection against line Y=X $\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

9. Reflection against Y= –X $\begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

10. Shearing in X direction $\begin{bmatrix} 1 & 0 & 0 \\ Sh_x & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

11. Shearing in Y direction $\begin{bmatrix} 1 & Sh_y & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

12. Shearing in both x and y direction $\begin{bmatrix} 1 & Sh_y & 0 \\ Sh_x & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Composite Transformation:

A number of transformations or sequence of transformations can be combined into single one called as composition. The resulting matrix is called as composite matrix. The process of combining is called as concatenation.

Suppose we want to perform rotation about an arbitrary point, then we can perform it by the sequence of three transformations

1. Translation
2. Rotation
3. Reverse Translation

The ordering sequence of these numbers of transformations must not be changed. If a matrix is represented in column form, then the composite transformation is performed by multiplying matrix in order from right to left side. The output obtained from the previous matrix is multiplied with the new coming matrix.

Example showing composite transformations:

The enlargement is with respect to center. For this following sequence of transformations will be performed and all will be combined to a single one

Step1: The object is kept at its position as in fig (a)

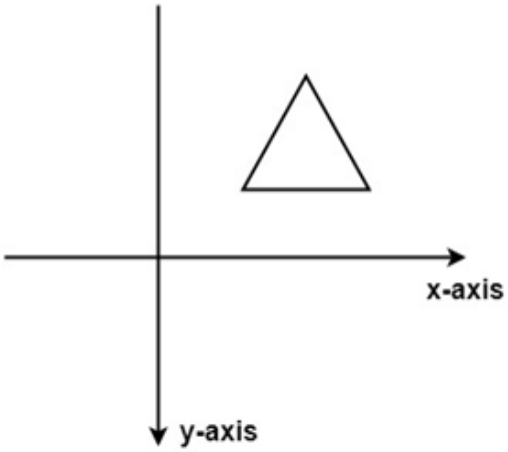
Step2: The object is translated so that its center coincides with the origin as in fig (b)

Step3: Scaling of an object by keeping the object at origin is done in fig (c)

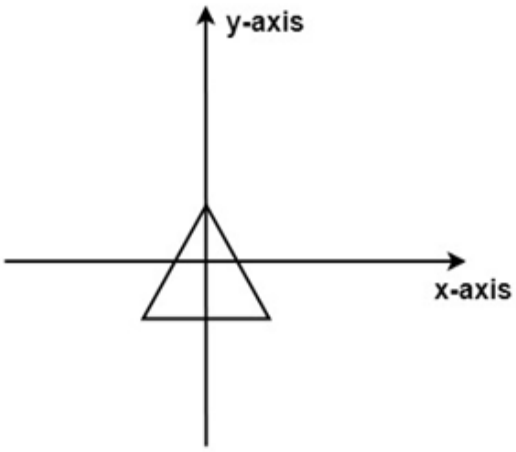
Step4: Again translation is done. This second translation is called a reverse translation. It will position the object at the origin location.

Above transformation can be represented as $T_V.ST_V^{-1}$

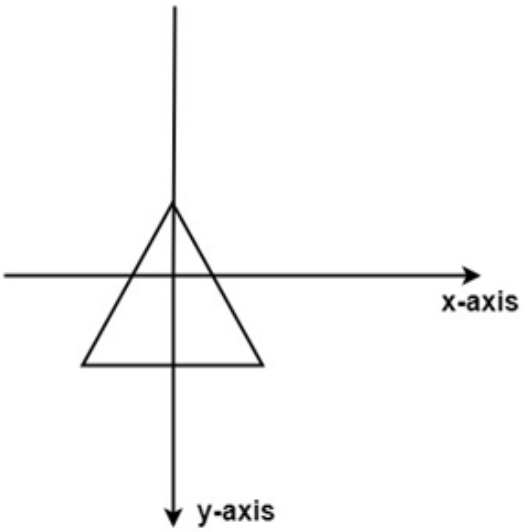
DIWAKAR EDUCATION HUB



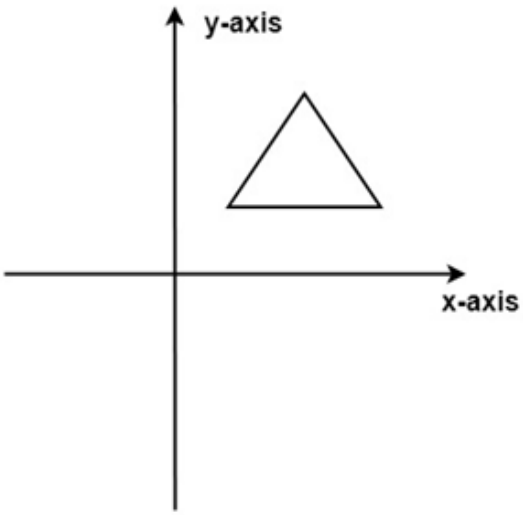
(original position of object)
Fig:(a)



(object is translated to origin)
Fig:(b)



(object is enlarged)
Fig:(c)



(object is retranslated to original position)
Fig:(d)

Note: Two types of rotations are used for representing matrices one is column method. Another is the row method.

Row Method

1. Translation

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix}$$

2. Scaling

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ S_x & S_y & 1 \end{bmatrix}$$

Column Method

1.Translation

$$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

2. Scaling

$$\begin{bmatrix} 1 & 0 & S_x \\ 0 & 1 & S_y \\ 0 & 0 & 1 \end{bmatrix}$$

Advantage of composition or concatenation of matrix:

1. It transformations become compact.
2. The number of operations will be reduced.
3. Rules used for defining transformation in form of equations are complex as compared to matrix.

Composition of two translations:

Let t_1 t_2 t_3 $t_4$are translation vectors. They are two translations P_1 and P_2. The matrix of P_1 and P_2 given below. The P_1 and $P_2$are represented using Homogeneous matrices and P will be the final transformation matrix obtained after multiplication.

$$P_1 = \begin{bmatrix} 1 & 0 & t_1 \\ 0 & 1 & t_2 \\ 0 & 0 & 1 \end{bmatrix} \qquad P_2 = \begin{bmatrix} 1 & 0 & t_3 \\ 0 & 1 & t_4 \\ 0 & 0 & 1 \end{bmatrix}$$

$$P = \begin{bmatrix} 1 & 0 & t_1 + t_2 \\ 0 & 1 & t_3 + t_4 \\ 0 & 0 & 1 \end{bmatrix}$$

Above resultant matrix show that two successive translations are additive.

DIWAKAR EDUCATION HUB

Composition of two Rotations: Two Rotations are also additive

Composition of two Scaling: The composition of two scaling is multiplicative. Let S_{11} and $S_{12}$are matrix to be multiplied.

$$S_{11} = \begin{bmatrix} S_1 & 0 & 0 \\ 0 & S_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$S_{12} = \begin{bmatrix} S_3 & 0 & 0 \\ 0 & S_4 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$S = S_{11} * S_{12} = S_{11}(S_1.S_2).S_{12}(S_3.S_4) = S(S_1.S_2.S_3.S_4)$$

Computer Graphics Window to Viewport Co-ordinate Transformation

Once object description has been transmitted to the viewing reference frame, we choose the window extends in viewing coordinates and selects the viewport limits in normalized coordinates.

Object descriptions are then transferred to normalized device coordinates:

We do this thing using a transformation that maintains the same relative placement of an object in normalized space as they had in viewing coordinates.

If a coordinate position is at the center of the viewing window:

It will display at the center of the viewport.

Fig shows the window to viewport mapping. A point at position (xw, yw) in window mapped into position (xv, yv) in the associated viewport.

DIWAKAR EDUCATION HUB

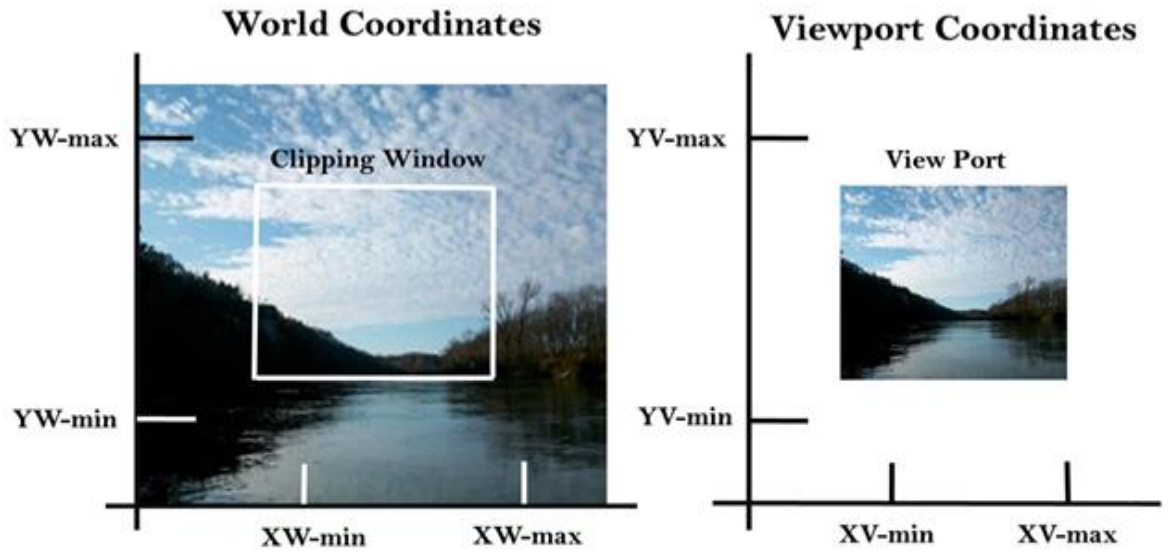


Fig: Window to viewport mapping

In order to maintain the same relative placement of the point in the viewport as in the window, we require:

$$\frac{xv - xv_{min}}{xv_{max} - xv_{min}} = \frac{xw - xw_{min}}{xw_{max} - xw_{min}} \quad \text{.....equation 1}$$

$$\frac{yv - yv_{min}}{yv_{max} - yv_{min}} = \frac{yw - yw_{min}}{yw_{max} - yw_{min}}$$

Solving these impressions for the viewport position (xv, yv), we have

$xv = xv_{min} + (xw - xw_{min})sx$

$yv = yv_{min} + (yw - yw_{min})sy$ .....equation 2

Where scaling factors are

$$sx = \frac{xv_{max} - xv_{min}}{xw_{max} - xw_{min}}$$

$$sy = \frac{yv_{max} - yv_{min}}{yw_{max} - yw_{min}}$$

Equation (1) and Equation (2) can also be derived with a set of transformation that converts the window or world coordinate area into the viewport or screen coordinate area. This conversation is performed with the following sequence of transformations:

1. Perform a scaling transformation using a fixed point position (xw_{min},yw_{min}) that scales the window area to the size of the viewport.
2. Translate the scaled window area to the position of the viewport. Relative proportions of objects are maintained if the scaling factors are the same (sx=sy).

From normalized coordinates, object descriptions are mapped to the various display devices.

Any number of output devices can we open in a particular app, and three windows to viewport transformation can be performed for each open output device.

This mapping called workstation transformation (It is accomplished by selecting a window area in normalized space and a viewport area in the coordinates of the display device).

As in fig, workstation transformation to partition a view so that different parts of normalized space can be displayed on various output devices).

DIWAKAR EDUCATION HUB

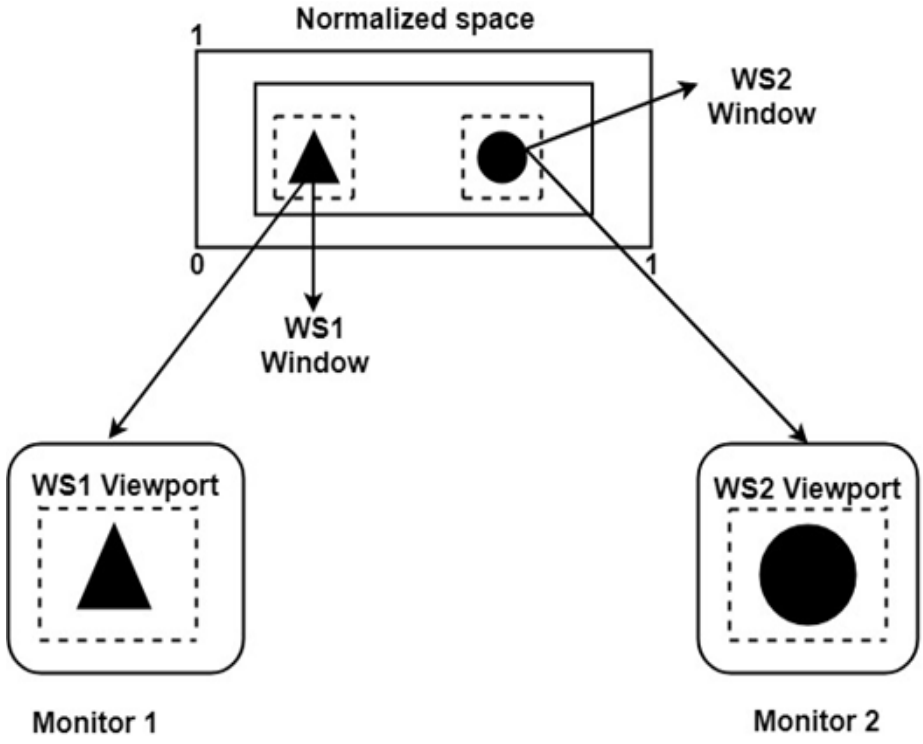
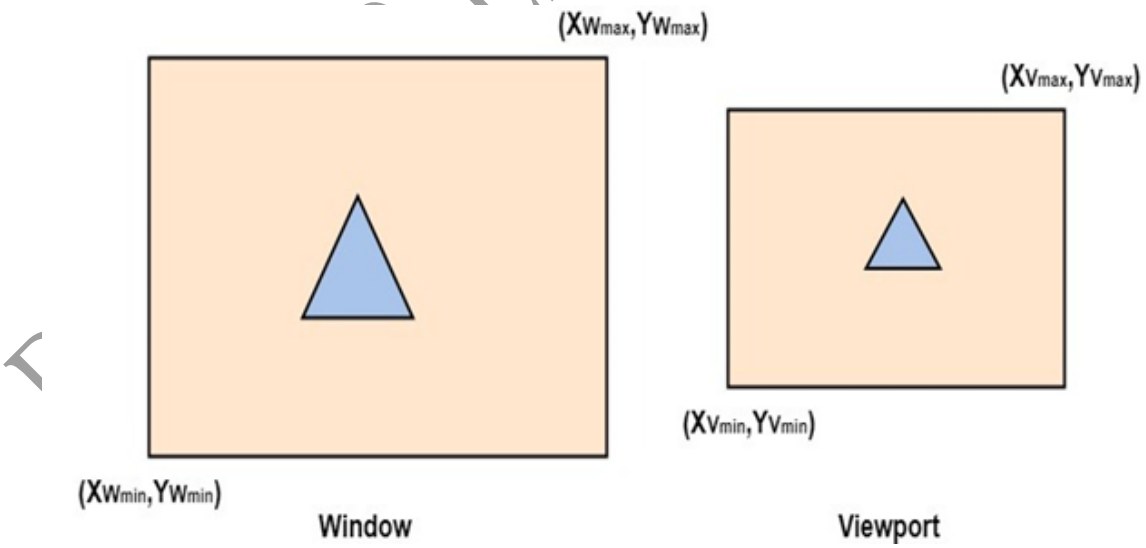


Fig:Mapping selected parts of a scene in normalized coordinates to different video monitors with workstation transformation.

Matrix Representation of the above three steps of Transformation:



Step1: Translate window to origin 1

T_x=-Xw_{min} T_y=-Yw_{min}

Step2: Scaling of the window to match its size to the viewport

$S_x=(Xy_{max}-Xv_{min})/(Xw_{max}-Xw_{min})$

$S_y=(Yv_{max}-Yv_{min})/(Yw_{max}-Yw_{min})$

Step3: Again translate viewport to its correct position on screen.

$T_x=Xv_{min}$

$T_y=Yv_{min}$

Above three steps can be represented in matrix form:

$VT=T * S * T_1$

T = Translate window to the origin

S=Scaling of the window to viewport size

T_1=Translating viewport on screen.

$$T = \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -Xw_{min} & -Yw_{min} & 1 \end{vmatrix}$$

$$S = \begin{vmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

$$T_1 = \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ Xv_{min} & Yv_{min} & 1 \end{vmatrix}$$

Viewing Transformation= $T * S * T_1$

Advantage of Viewing Transformation:

We can display picture at device or display system according to our need and choice.

Note:

- World coordinate system is selected suits according to the application program.

DIWAKAR EDUCATION HUB

- Screen coordinate system is chosen according to the need of design.
- Viewing transformation is selected as a bridge between the world and screen coordinate.

Line Clipping:

It is performed by using the line clipping algorithm. The line clipping algorithms are:

1. Cohen Sutherland Line Clipping Algorithm
2. Midpoint Subdivision Line Clipping Algorithm
3. Liang-Barsky Line Clipping Algorithm

Cohen Sutherland Line Clipping Algorithm:

In the algorithm, first of all, it is detected whether line lies inside the screen or it is outside the screen. All lines come under any one of the following categories:

1. Visible
2. Not Visible
3. Clipping Case

1. Visible: If a line lies within the window, i.e., both endpoints of the line lies within the window. A line is visible and will be displayed as it is.

2. Not Visible: If a line lies outside the window it will be invisible and rejected. Such lines will not display. If any one of the following inequalities is satisfied, then the line is considered invisible. Let A (x_1,y_2) and B (x_2,y_2) are endpoints of line.

x_{min},x_{max} are coordinates of the window.

y_{min},y_{max} are also coordinates of the window.

$x_1 > x_{max}$

$x_2 > x_{max}$

$y_1 > y_{max}$

DIWAKAR EDUCATION HUB

$y_2 > y_{max}$

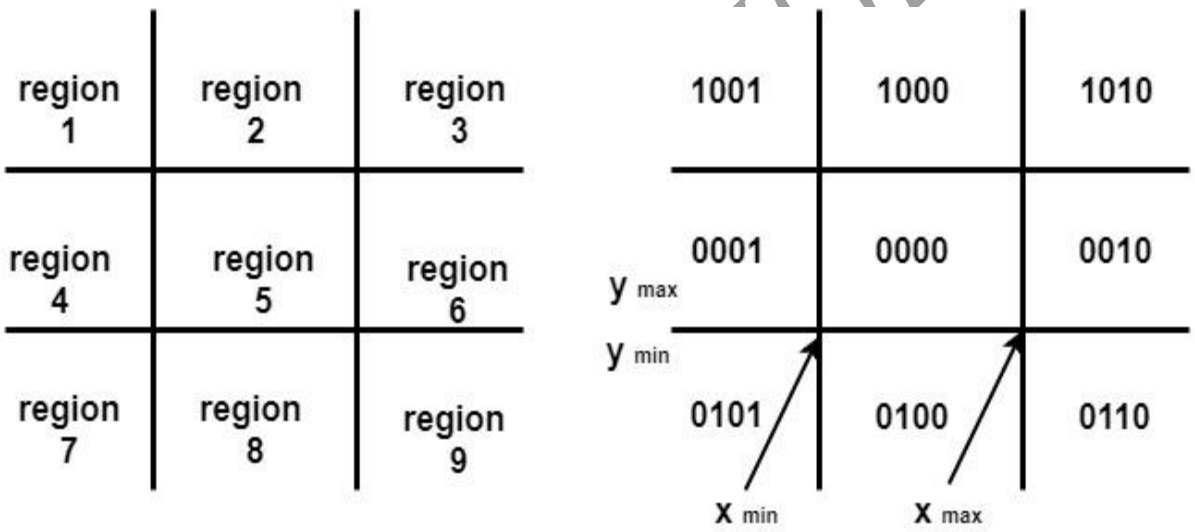
$x_1 < x_{min}$

$x_2 < x_{min}$

$y_1 < y_{min}$

$y_2 < y_{min}$

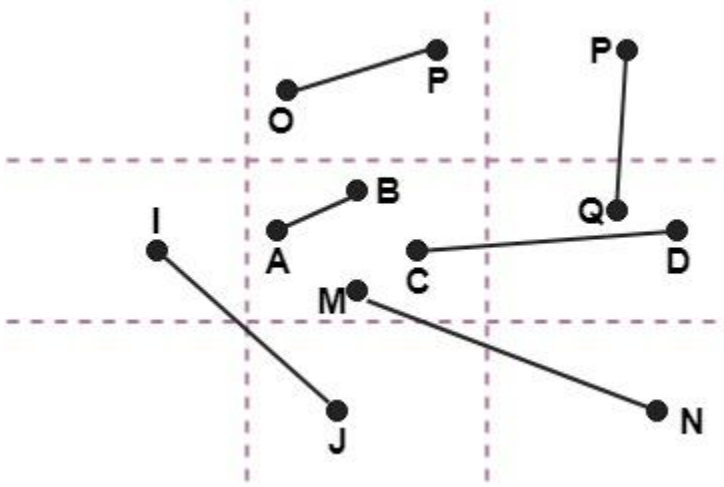
3. Clipping Case: If the line is neither visible case nor invisible case. It is considered to be clipped case. First of all, the category of a line is found based on nine regions given below. All nine regions are assigned codes. Each code is of 4 bits. If both endpoints of the line have end bits zero, then the line is considered to be visible.



9 region bits assigned to 9 regions

The center area is having the code, 0000, i.e., region 5 is considered a rectangle window.

Following figure show lines of various types



Line AB is the visible case
Line OP is an invisible case
Line PQ is an invisible line
Line IJ are clipping candidates
Line MN are clipping candidate
Line CD are clipping candidate

Advantage of Cohen Sutherland Line Clipping:

1. It calculates end-points very quickly and rejects and accepts lines quickly.
2. It can clip pictures much large than screen size.

Algorithm of Cohen Sutherland Line Clipping:

Step1: Calculate positions of both endpoints of the line

Step2: Perform OR operation on both of these end-points

Step3: If the OR operation gives 0000

Then

line is considered to be visible

else

Perform AND operation on both endpoints

If And ≠ 0000

DIWAKAR EDUCATION HUB

then the line is invisible

else

And=0000

Line is considered the clipped case.

Step4: If a line is clipped case, find an intersection with boundaries of the window

$m=(y_2-y_1)(x_2-x_1)$

(a) If bit 1 is "1" line intersects with left boundary of rectangle window

$y_3=y_1+m(x-X_1)$

where $X = X_{wmin}$

where $X_{wmin}$is the minimum value of X co-ordinate of window

(b) If bit 2 is "1" line intersect with right boundary

$y_3=y_1+m(X-X_1)$

where $X = X_{wmax}$

where X more is maximum value of X co-ordinate of the window

(c) If bit 3 is "1" line intersects with bottom boundary

$X_3=X_1+(y-y_1)/m$

where $y = y_{wmin}$

y_{wmin} is the minimum value of Y co-ordinate of the window

(d) If bit 4 is "1" line intersects with the top boundary

$X_{3=X}1+(y-y_1)/m$

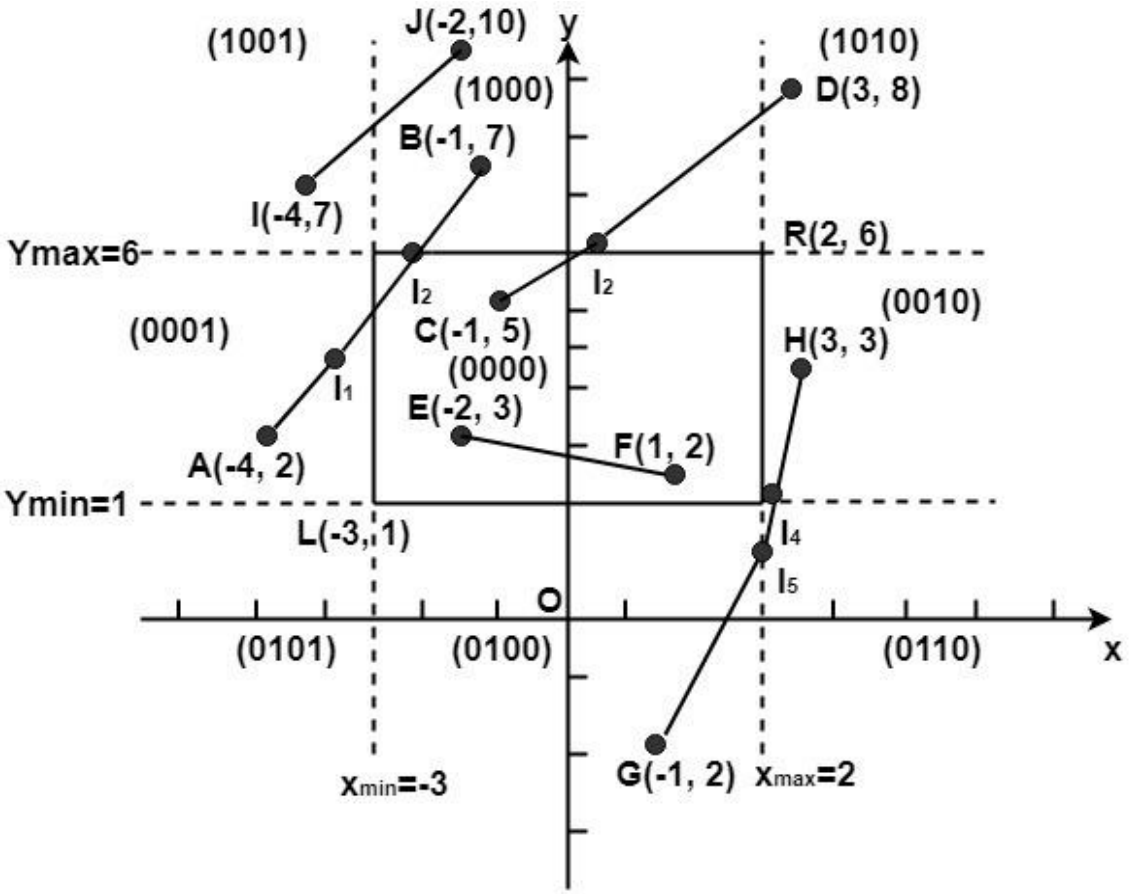
where $y = y_{wmax}$

y_{wmax} is the maximum value of Y co-ordinate of the window

Example of Cohen-Sutherland Line Clipping Algorithm:

Let R be the rectangular window whose lower left-hand corner is at L (-3, 1) and upper right-hand corner is at R (2, 6). Find the region codes for the endpoints in fig:

DIWAKAR EDUCATION HUB



The region code for point (x, y) is set according to the scheme

Bit 1 = sign $(y-y_{max})$=sign (y-6) Bit 3 = sign $(x-x_{max})$= sign (x-2)

Bit 2 = sign $(y_{min}-y)$=sign(1-y) Bit 4 = sign $(x_{min}-x)$=sign(-3-x)

Here

$$\text{sign}(a) = \begin{cases} 1 & \text{if a is positive} \\ 0 & \text{otherwise} \end{cases}$$

So

A (-4, 2)→ 0001 F (1, 2)→ 0000

B (-1, 7) → 1000 G (1, -2) →0100

C (-1, 5)→ 0000 H (3, 3) → 0100

D (3, 8) → 1010 I (-4, 7) → 1001
E (-2, 3) → 0000 J (-2, 10) → 1000

We place the line segments in their appropriate categories by testing the region codes found in the problem.

Category1 (visible): EF since the region code for both endpoints is 0000.

Category2 (not visible): IJ since (1001) AND (1000) =1000 (which is not 0000).

Category 3 (candidate for clipping): AB since (0001) AND (1000) = 0000, CD since (0000) AND (1010) =0000, and GH. since (0100) AND (0010) =0000.

The candidates for clipping are AB, CD, and GH.

In clipping AB, the code for A is 0001. To push the 1 to 0, we clip against the boundary line x_{min}=-3. The resulting intersection point is I_1 (-3,3 $\frac{2}{3}$). We clip (do not display) AI_1 and I_1 B. The code for $I_1$is 1001. The clipping category for I_1 B is 3 since (0000) AND (1000) is (0000). Now B is outside the window (i.e., its code is 1000), so we push the 1 to a 0 by clipping against the line y_{max}=6. The resulting intersection is I_2 (-1 $\frac{3}{5}$,6). Thus I_2 B is clipped. The code for I_2 is 0000. The remaining segment I_1 I_2 is displayed since both endpoints lie in the window (i.e., their codes are 0000).

For clipping CD, we start with D since it is outside the window. Its code is 1010. We push the first 1 to a 0 by clipping against the line y_{max}=6. The resulting intersection I_3 is ($\frac{3}{5}$,6),and its code is 0000. Thus I_3 D is clipped and the remaining segment CI_3 has both endpoints coded 0000 and so it is displayed.

For clipping GH, we can start with either G or H since both are outside the window. The code for G is 0100, and we push the 1 to a 0 by clipping against the line y_{min}=1.The resulting intersection point is I_4 (2 $\frac{1}{5}$,1) and its code is 0010. We clip GI_4 and work on I_4 H. Segment I_4 H is not displaying since (0010) AND (0010) =0010.

DIWAKAR EDUCATION HUB

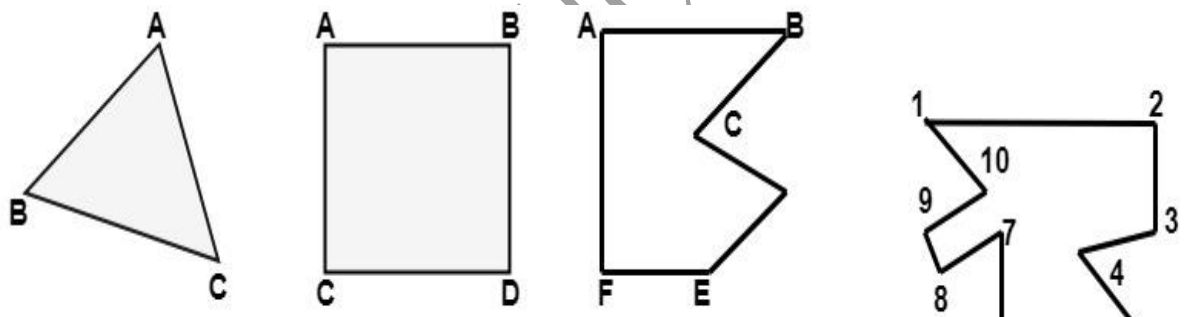
Polygon:

Polygon is a representation of the surface. It is primitive which is closed in nature. It is formed using a collection of lines. It is also called as many-sided figure. The lines combined to form polygon are called sides or edges. The lines are obtained by combining two vertices.

Example of Polygon:

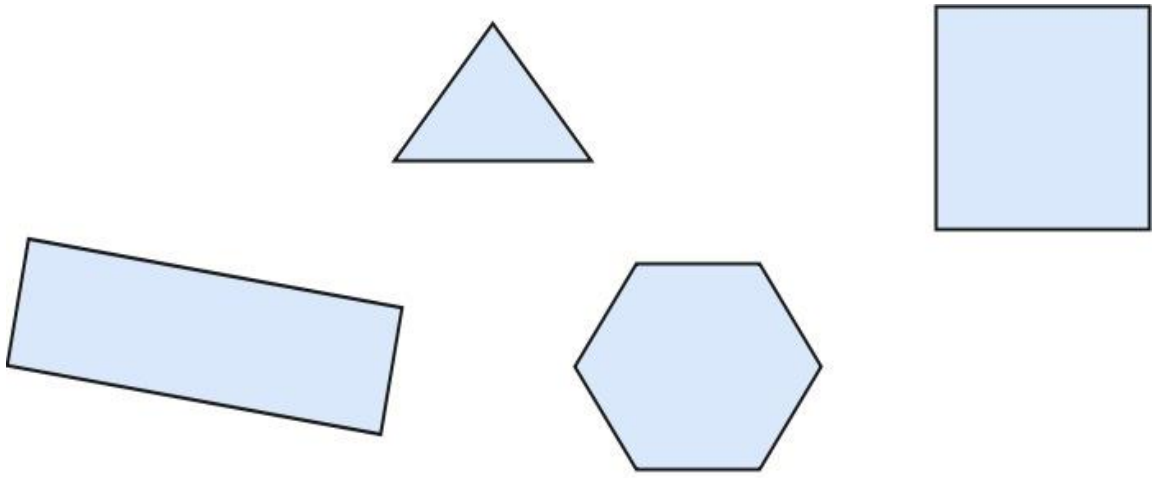
1. Triangle
2. Rectangle
3. Hexagon
4. Pentagon

Following figures shows some polygons.

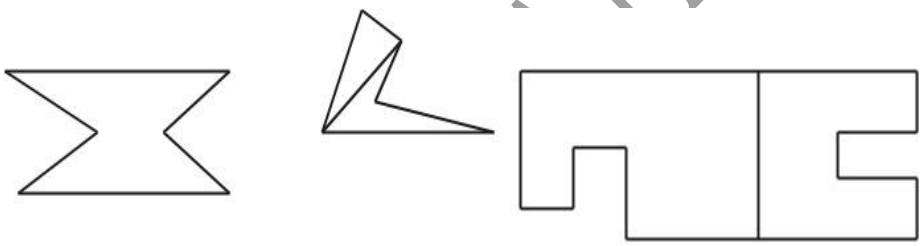


Polygons

DIWAKAR EDUCATION HUB



Convex Polygon



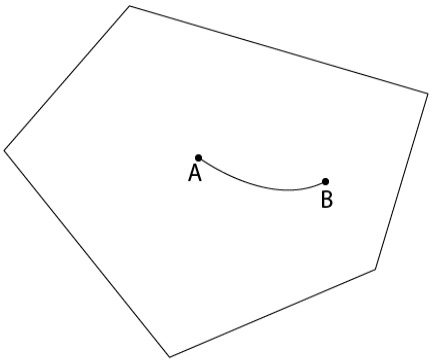
Concave Polygon

Types of Polygons

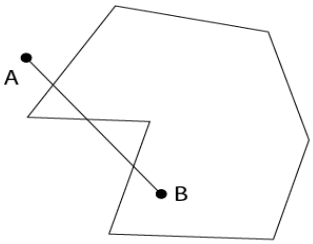
1. Concave
2. Convex

A polygon is called convex of line joining any two interior points of the polygon lies inside the polygon. A non-convex polygon is said to be concave. A concave polygon has one interior angle greater than 180°. So that it can be clipped into similar polygons.

DIWAKAR EDUTECH



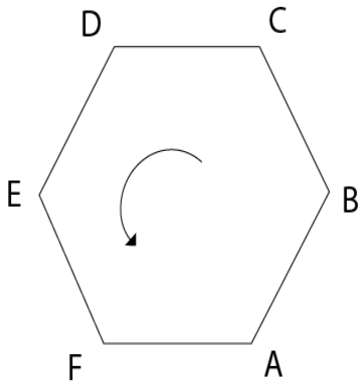
Convex polygon



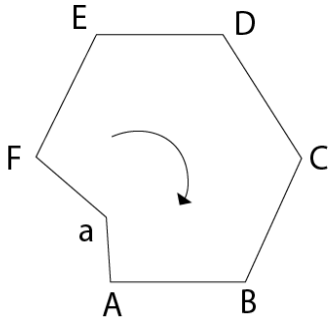
Concave polygon

A polygon can be positive or negative oriented. If we visit vertices and vertices visit produces counterclockwise circuit, then orientation is said to be positive.

DIWAKAR EDUCATION HUB



Polygon with positive orientation



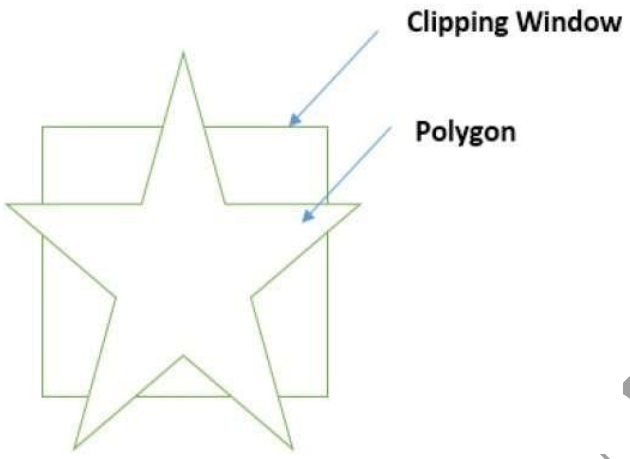
Polygon with negative orientation

Polygon Clipping SutherlandHodgmanAlgorithmSutherlandHodgmanAlgorithm

A polygon can also be clipped by specifying the clipping window. Sutherland Hodgeman polygon clipping algorithm is used for polygon clipping. In this algorithm, all the vertices of the polygon are clipped against each edge of the clipping window.

First the polygon is clipped against the left edge of the polygon window to get new vertices of the polygon. These new vertices are used to clip the polygon against right edge, top edge, bottom edge, of the clipping window as shown in the following figure.

DIWAKAR EDUCATION HUB



While processing an edge of a polygon with clipping window, an intersection point is found if edge is not completely inside clipping window and the a partial edge from the intersection point to the outside edge is clipped. The following figures show left, right, top and bottom edge clippings –

DIWAKAR EDUCATION HUB

Computer Graphics

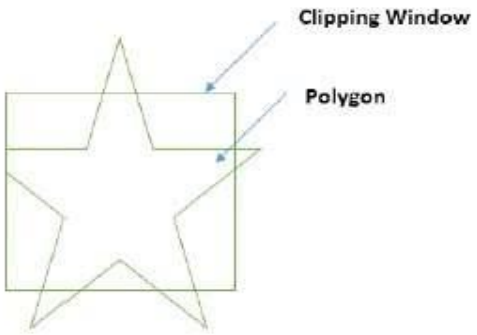


Figure: Clipping Left Edge

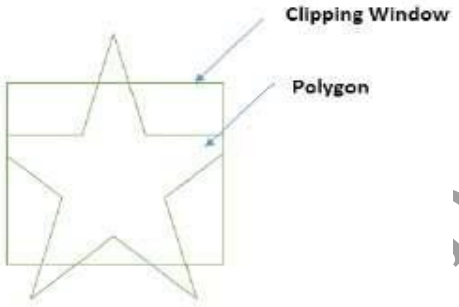


Figure: Clipping Right Edge

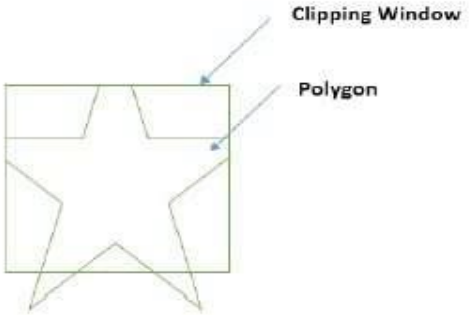


Figure: Clipping Top Edge

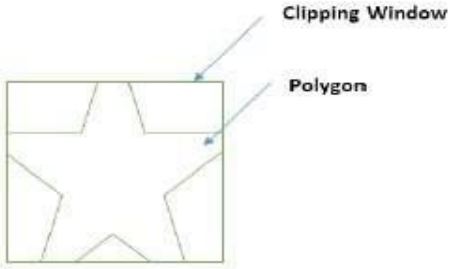


Figure: Clipping Bottom Edge

3-D Object Representation, Geometric Transformations and Viewing

9. 3D Object Representations

Methods:

- Polygon and Quadric surfaces: For simple Euclidean objects
- Spline surfaces and construction: For curved surfaces
- Procedural methods: Eg. Fractals, Particle systems
- Physically based modeling methods
- Octree Encoding
- Isosurface displays, Volume rendering, etc.

DIWAKAR

Classification:

Boundary Representations (B-reps) eg. Polygon facets and spline patches Space-partitioning representations eg. Octree Representation

Objects may also associate with other properties such as mass, volume, so as to determine their response to stress and temperature etc.

Polygon Surfaces

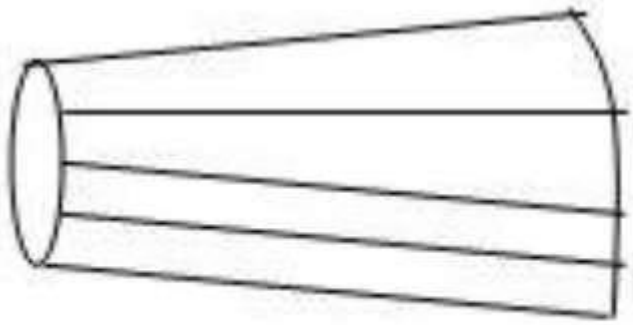
Objects are represented as a collection of surfaces. 3D object representation is divided into two categories.

- **Boundary Representations** B−repsB−reps − It describes a 3D object as a set of surfaces that separates the object interior from the environment.
- **Space−partitioning representations** − It is used to describe interior properties, by partitioning the spatial region containing an object into a set of small, non-overlapping, contiguous solids usuallycubesusuallycubes.

The most commonly used boundary representation for a 3D graphics object is a set of surface polygons that enclose the object interior. Many graphics system use this method. Set of polygons are stored for object description. This simplifies and speeds up the surface rendering and display of object since all surfaces can be described with linear equations.

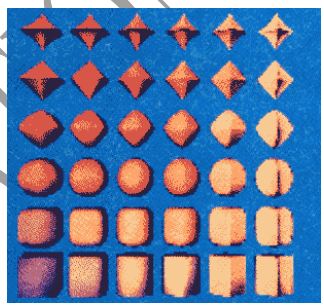
The polygon surfaces are common in design and solid-modeling applications, since their **wireframe display** can be done quickly to give general indication of surface structure. Then realistic scenes are produced by interpolating shading patterns across polygon surface to illuminate.

DIWAKAR EDUCATION HUB



A 3D object represented by polygons

Curved Surfaces



1. Regular curved surfaces can be generated as

- Quadric Surfaces, eg. Sphere, Ellipsoid, or
- Superquadrics, eg. Superellipsoids

These surfaces can be represented by some simple parametric equations, eg, for ellipsoid:

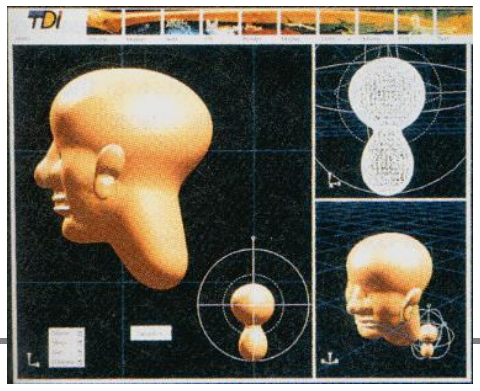
$x = r_x \cos^{s1}\phi \cos^{s2}\theta, \quad -\pi/2 <= \phi <= \pi/2$

$y = r_y \cos^{s1}\phi \sin^{s2}\theta, \quad -\pi <= \theta <= \pi$

$z = r_z \sin^{s1}\phi$

Where s1, r_x, r_y, and r_z are constants. By varying the values of ϕ and θ, points on the surface can be computed.

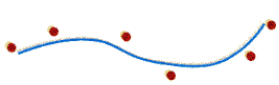
2. Irregular surfaces can also be generated using some special formulating approach, to form a kind of **blobby objects** -- The shapes showing a certain degree of fluidity.

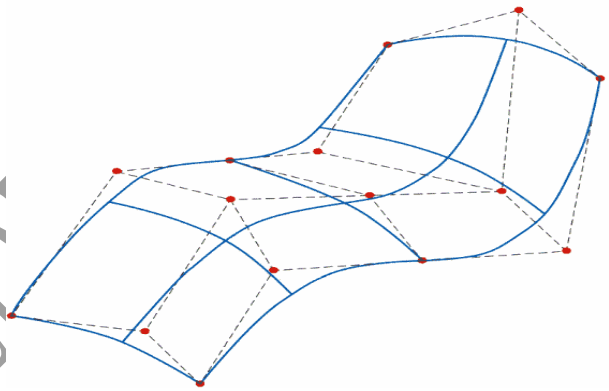


3. Spline Representations

Spline means a flexible strip used to produce a smooth curve through a designated set of points. Several small weights are distributed along the length of the strip to hold it in position on the drafting table as the curve is drawn.

We can mathematically describe such a curve with a piecewise cubic polynomial function => spline curves. Then a spline surface can be described with 2 sets of orthogonal spline curves.





DIWAKAR EDUCATION HUB

Quadric Surfaces

Quadric surfaces are defined by quadratic equations in two dimensional space. Spheres and cones are examples of quadrics. The quadric surfaces of RenderMan are surfaces of revolution in which a finite curve in two dimensions is swept in three dimensional space about one axis to create a surface. A circle centered at the origin forms a sphere. If the circle is not centered at the origin, the circle sweeps out a torus. A line segment with one end lying on the axis of rotation forms a cone. A line segment parallel to the axis of rotation forms a cylinder. The generalization of a line segment creates a hyperboloid by rotating an arbitrary line segment in three dimensional space about the Z axis. The axis of rotation is always the z axis. Each quadric routine has a sweep parameter, specifying the

angular extent to which the quadric is swept about z axis. Sweeping a quadric by less than 360 degrees leaves an open surface.

Quadrics

Many common shapes can be modeled with quadrics. Although it is possible to convert quadrics to patches, they are defined as primitives because special-purpose rendering programs render them directly and because their surface parameters are not necessarily preserved if they are converted to patches. Quadric primitives are particularly useful in solid and molecular modeling applications.

All the following quadrics are rotationally symmetric about the z axis. In all the quadrics *u* and *v* are assumed to run from 0 to 1. These primitives all define a bounded region on a quadric surface. It is not possible to define infinite quadrics. Note that each quadric is defined relative to the origin of the object coordinate system. To position them at another point or with their symmetry axis in another direction requires the use a modeling transformation. The geometric normal to the surface points ``outward'' from the z-axis, if the *current orientation* matches the orientation of the *current transformation* and "inward" if they don't match. The sense of a quadric can be reversed by giving negative parameters. For example, giving a negative *thetamax* parameter in any of the following definitions will turn the quadric inside-out.

Each quadric has a *parameterlist*. This is a list of token-array pairs where each token is one of the standard geometric primitive variables or a variable which has been defined with **RiDeclare**. Position variables should not be given with quadrics. All angular arguments to these functions are given in degrees. The trigonometric functions used in their definitions are assumed to also accept angles in degrees.

RiSphere(radius, zmin, zmax, thetamax, parameterlist)

RtFloat radius;

RtFloat zmin, zmax;

DIWAKAR EDUCATION HUB

RtFloat thetamax;

Requests a sphere defined by the following equations:

$$\phi_{min} = \begin{cases} asin\left(\frac{zmin}{radius}\right) & \text{if } zmin > -radius \\ -90.0 & \text{if } zmin \le -radius \end{cases}$$

$$\phi_{max} = \begin{cases} asin\left(\frac{zmax}{radius}\right) & \text{if } zmax < radius \\ 90.0 & \text{if } zmax \ge radius \end{cases}$$

$$\phi = \phi_{min} + v \cdot (\phi_{max} - \phi_{min})$$

$$\theta = u \cdot thetamax$$

$$x = radius \cdot cos(\theta) \cdot cos(\phi)$$

$$y = radius \cdot sin(\theta) \cdot cos(\phi)$$

$$z = radius \cdot sin(\phi)$$

Note that if zmin > -radius or zmax < radius, the bottom or top of the sphere is open, and that if thetamax is not equal to 360 degrees, the sides are also open.

RIB BINDING

Sphere radius zmin zmax thetamax parameterlist

Sphere [radius zmin zmax thetamax] parameterlist

EXAMPLE

RiSphere(0.5, 0.0, 0.5, 360.0, RI_NULL);

RiCone(height, radius, thetamax, parameterlist)

RtFloat height;

RtFloat radius;

RtFloat thetamax;

DIWAKAR EDUCATION HUB

Requests a cone defined by the following equations:

$$\theta = u \cdot \text{thetamax}$$
$$x = \text{radius} \cdot (1 - v) \cdot \cos(\theta)$$
$$y = \text{radius} \cdot (1 - v) \cdot \sin(\theta)$$
$$z = v \cdot \text{height}$$

Note that the bottom of the cone is open, and if *thetamax* is not equal to 360 degrees, the sides are open.

RIB BINDING

Cone height radius thetamax parameterlist

Cone [height radius thetamax] parameterlist

EXAMPLE

RtColor four_colors[4];

RiCone(0.5, 0.5, 270.0, "Cs", (**RtPointer**)four_colors, RI_NULL);

RiCylinder(radius, zmin, zmax, thetamax, parameterlist)

RtFloat radius;

RtFloat zmin, zmax;

RtFloat thetamax;

Requests a cylinder defined by the following equations:

$$\theta = u \cdot \text{thetamax}$$
$$x = \text{radius} \cdot \cos(\theta)$$
$$y = \text{radius} \cdot \sin(\theta)$$
$$z = \text{zmin} + v \cdot (\text{zmax} - \text{zmin})$$

DIWAKAR EDUCATION HUB

Note that the cylinder is open at the top and bottom, and if *thetamax* is not equal to 360 degrees, the sides also are open.

RIB BINDING

Cylinder radius zmin zmax thetamax parameterlist

Cylinder [radius zmin zmax thetamax] parameterlist

Bezier Curves

Bezier curve is discovered by the French engineer **Pierre Bézier**. These curves can be generated under the control of other points. Approximate tangents by using control points are used to generate curve. The Bezier curve can be represented mathematically as −

$$\sum_{k=0}^{n} P_i B_i^n(t)$$

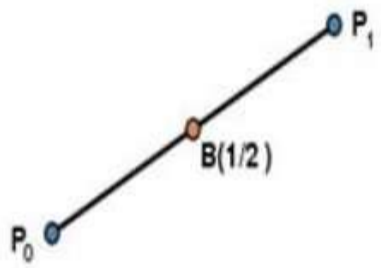
Where pipi is the set of points and Bni(t)Bin(t) represents the Bernstein polynomials which are given by −

$$B_i^n(t) = \binom{n}{i}(1-t)^{n-i}t^i$$

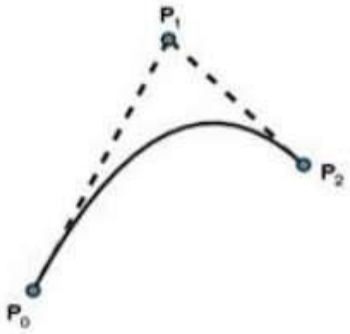
Where **n** is the polynomial degree, **i** is the index, and **t** is the variable.

The simplest Bézier curve is the straight line from the point P0P0 to P1P1. A quadratic Bezier curve is determined by three control points. A cubic Bezier curve is determined by four control points.

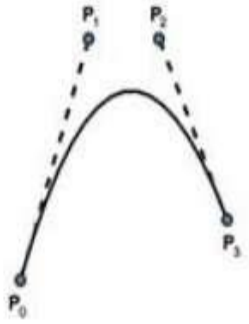
DIWAKAR EDUCATION HUB



Simple Bezier Curve



Quadratic Bazier Curve



Cubic Bazier Curve

Properties of Bezier Curves

Bezier curves have the following properties –

- They generally follow the shape of the control polygon, which consists of the segments joining the control points.
- They always pass through the first and last control points.
- They are contained in the convex hull of their defining control points.
- The degree of the polynomial defining the curve segment is one less that the number of defining polygon point. Therefore, for 4 control points, the degree of the polynomial is 3, i.e. cubic polynomial.
- A Bezier curve generally follows the shape of the defining polygon.
- The direction of the tangent vector at the end points is same as that of the vector determined by first and last segments.
- The convex hull property for a Bezier curve ensures that the polynomial smoothly follows the control points.
- No straight line intersects a Bezier curve more times than it intersects its control polygon.
- They are invariant under an affine transformation.

DIWAKAR EDUCATION

- Bezier curves exhibit global control means moving a control point alters the shape of the whole curve.
- A given Bezier curve can be subdivided at a point t=t0 into two Bezier segments which join together at the point corresponding to the parameter value t=t0.

B-Spline Curves

The Bezier-curve produced by the Bernstein basis function has limited flexibility.

- First, the number of specified polygon vertices fixes the order of the resulting polynomial which defines the curve.
- The second limiting characteristic is that the value of the blending function is nonzero for all parameter values over the entire curve.

The B-spline basis contains the Bernstein basis as the special case. The B-spline basis is non-global.

A B-spline curve is defined as a linear combination of control points Pi and B-spline basis function Ni,Ni, k tt given by

$$C(t) = \sum_{i=0}^{n} P_i N_{i,k}(t), \qquad n \geq k-1, \qquad t \epsilon [tk-1, tn+1]$$

Where,

DIWAKAR EDUCATION HUB

- { p_i : i=0, 1, 2....n} are the control points
- k is the order of the polynomial segments of the B-spline curve. Order k means that the curve is made up of piecewise polynomial segments of degree k - 1,
- the $N_{i,k}(t)$ are the "normalized B-spline blending functions". They are described by the order k and by a non-decreasing sequence of real numbers normally called the "knot sequence".

$$t_i : i = 0, \ldots n + K$$

The $N_{i,k}$ functions are described as follows –

$$N_{i,1}(t) = \begin{cases} 1, & if\ u \epsilon [t_i, t_{i+1}) \\ 0, & Otherwise \end{cases}$$

and if k > 1,

$$N_{i,k}(t) = \frac{t - t_i}{t_{i+k-1}} N_{i,k-1}(t) + \frac{t_{i+k} - t}{t_{i+k} - t_{i+1}} N_{i+1,k-1}(t)$$

and

$$t \epsilon [t_{k-1}, t_{n+1})$$

Properties of B-spline Curve

B-spline curves have the following properties –

- The sum of the B-spline basis functions for any parameter value is 1.
- Each basis function is positive or zero for all parameter values.
- Each basis function has precisely one maximum value, except for k=1.
- The maximum order of the curve is equal to the number of vertices of defining polygon.

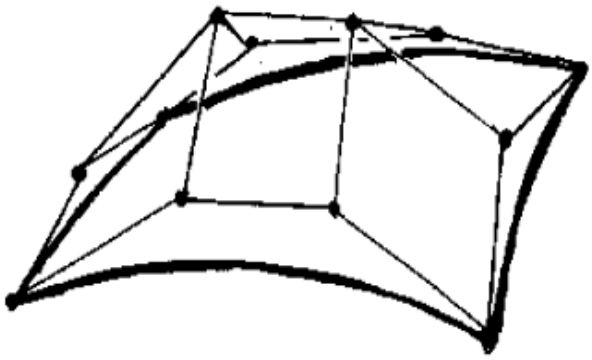
- The degree of B-spline polynomial is independent on the number of vertices of defining polygon.
- B-spline allows the local control over the curve surface because each vertex affects the shape of a curve only over a range of parameter values where its associated basis function is nonzero.
- The curve exhibits the variation diminishing property.
- The curve generally follows the shape of defining polygon.
- Any affine transformation can be applied to the curve by applying it to the vertices of defining polygon.
- The curve line within the convex hull of its defining polygon.

Bezier and B-spline Surfaces

Bezier Surfaces.

Like the Bezier curves, the Bezier surfaces use the Bernstein polynomials as blending functions. We now have the control points being points on a design net, which is again a rectangular mesh spread over area of interest. The x and y coordinates of the control points are fixed and the shape of the surface varies as the control points are moved up or down. The figure below shows a Bezier surface with an associated control net of 3 points by 4 points. The values of the points are stored in an array B(4,3) and the blending functions are obtained from the equations of the Bezier curves.

DIWAKAR EDUCATION HUB



Bezier Surface and Control Net

Thus, the three point Bezier curve is given by :

$$B(v) = (1-v)2B(I,1) + 2v(1-v)B(I,2) + v2B(I,3)$$

or

$$B(v) = b(20)(v)\ B(I,1) + b(21)(v)\ B(I,2) + b(22)(v)\ B(I,3)$$

and the 4-point Bezier curve needed in the u-direction has the form:

$$B(u) = (1-u)3B(1,J) + 3u(1-u)2B(2,J) + 3u2(1-u)B(3,J) + u3B(4,J)$$

or $B(u) = b\{30\}(u)B(1,J) + b\{31\}(u)\ B(2,J) + b\{32\}(u)\ B(3,J) + b\{33\}(u)\ B(4,J)$

These must be combined together in the matrix equation to calculate any point on the surface. The equation of a point on the Bezier surface is given by:

r(u,v) = [b{30}(u) b{31}(u) b{32}(u) b{33}(u)] B(1,1)B(1,2) B(1,3) b{20}(v)

B(2,1) B(2,2) B(2,3) b{21}(v)

B(3,1) B(3,2) B(3,3) b{22}(v)

B(4,1) B(4,2) B(4,3)

To obtain the equation of one of the edge-curves, e.g. when u=0, we must substitute in the matrix equation:

DIWAKAR EDUCATION HUB

r(0,v) = B(1,1) B(1,2) B(1,3) (1-v)(1-v)

2v(1-v)

v*v

which becomes r(0,v) = (1-v)(1-v)B(1,1) + 2v(1-v)B(1,2) + v*vB(1,3) and similarly for the other edge curves. This form of the Bezier curve assumes a rectangular mesh with m+1 points in the one direction and n+1 points in the other. It requires the edge curves defining the patches to be coplanar and does not provide local control within a patch. A surface may be made up of several Bezier patches and, as for the curves, if we require the tangents to be continuous across the joins, then the sets of control points across the curves must be not only coplanar but also collinear.

The Proceedings of the Royal Society, Series A, for February 1971 contained the papers from a meeting on `Computer Aids in Mechanical Engineering Design'. This included a paper by Bezier on his `Unisurf' package used by the Renault car company to design their car bodies. Although it is not obvious from the paper, this was the earliest use of Bezier surfaces, which were invented for this package. Another paper described the `Multiobject' package by Armit, which used the same equations. The paper by Sabin, discribing the `Numerical Master Geometry' in use at the British Aircraft Corporation, shows the extensive use of Bicubic Surface tiles in their work because they are particularly concerned to get the extra smoothness of functions which can only be guaranteed by bicubic or higher-order patches.

This volume of the proceedings gives an interesting insight into the way that these packages were viewed at the time. In some respects, things have changed little since then. They describe a situation where many users had their own two or three C.A.D. programs to use, and there was very little guidance on the best choice for a newcomer to the field. Nowadays, there is a bewildering variety of large and comprehensive packages for C.A.D. but the newcomer still has little

DIWAKAR EDUCATION HUB

guidance amongst an even larger choice of software. Unfortunately the cost of a mistake is very much larger, since these packages are now big business.

B-Spline Surfaces.

The use of B-Spline surfaces is necessary in some CAD applications to give local control of the shape. The equation of this surface is given by

Q(u,v) = sum{i=0 to m} sum{j=0 to n} B(i+1,j+1) N{i,k}(u) M{j,l}(v)

In the x-direction, x(i) are the elements of the k-knot vector and

N{i,1}(u) = 1 if x(i) <= u <= x(i+1)

= 0 otherwise

and

$$N\{i.k\}(u) = \frac{(u-x(i)) * N\{i,k-1\}(u)}{(x(i+k-1)-x(i))} + \frac{(x(i+k)-u) * N\{i+1,k-1\}(u)}{(x(i+k)-x(i+1))}$$

In the y-direction, the y(j) are elements of the l-knot vector and we have similar expressions for M{j,1}(v) and M{j,l}(v).

This simple version of the B-Spline expects a rectangular net and requires all polygons in the one-direction to have the same degree of multiplicity (i.e. to use one knot-vector for all lines across the net). We can obtain greater flexibility by relaxing these conditions, but at the expense of greater complexity. These curves are widely used in many CAD packages.

Illumination model

DIWAKAR EDUCATION HUB

Illumination model, also known as Shading model or Lightning model, is used to calculate the intensity of light that is reflected at a given point on surface. There are three factors on which lightning effect depends on:

1. **Light Source :**
   Light source is the light emitting source. There are three types of light sources:
   1. **Point Sources –** The source that emit rays in all directions (A bulb in a room).
   2. **Parallel Sources –** Can be considered as a point source which is far from the surface (The sun).
   3. **Distributed Sources –** Rays originate from a finite area (A tubelight).

Their position, electromagnetic spectrum and shape determine the lightning effect.

2. **Surface :**
   When light falls on a surface part of it is reflected and part of it is absorbed. Now the surface structure decides the amount of reflection and absorption of light. The position of the surface and positions of all the nearby surfaces also determine the lightning effect.
3. **Observer :**
   The observer's position and sensor spectrum sensitivities also affect the lightning effect.
   1. **Ambient Illumination :**
      Assume We are standing on a road, facing a building with glass exterior and sun rays are falling on that building reflecting back from it and the falling on the object under observation. This would be **Ambient Illumination**. In simple words, Ambient Illumination is the one where source of light is indirect.

DIWAKAR EDUCATION HUB

The reflected intensity I_{amb} of any point on the surface is:

$$I_{amb} = K_a I_a$$

Where, I_a : ambient light intensity
K_a : surface ambient reflectivity, value of K_a varies from 0 to 1

2. Diffuse Reflection :

Diffuse reflection occurs on the surfaces which are rough or grainy. In this reflection the brightness of a point depends upon the angle made by the light source and the surface.

The reflected intensity I_{diff} of a point on the surface is:

$$I_{diff} = K_d I_p \cos(\theta) = K_d I_p (N \cdot L)$$

Where, I_p : the point light intensity
K_d : the surface diffuse reflectivity, value of K_d varies from 0 to 1
N : the surface normal
L : the light direction

3. Specular Reflection :

When light falls on any shiny or glossy surface most of it is reflected back, such reflection is known as Specular Reflection.

Phong Model is an empirical model for Specular Reflection which provides us with the formula for calculation the reflected intensity I_{spec}:

$$I_{spec} = W(\theta)\, I_l \cos^n(\Phi)$$

where, $W(\theta) : K_s$
L : direction of light source
N : normal to the surface
R : direction of reflected ray
V : direction of observer
Θ : Angle between L and R
Φ : angle between R and V

Polygen Rending Methods

Basically, the process of **creating images from scenes is called rendering**. Recently, the phrase **image synthesis** has also been applied to the creation of computer-generated three-dimensional images. Within this context, the **Polygon Rendering Methods** refer to **giving proper intensity at each point** in a **graphical object** in order to make it **look like real world object**. Different Rendering Methods are Constant Intensity Shading, Gouraud Shading, Phong Shading and Fast Phong Shading.

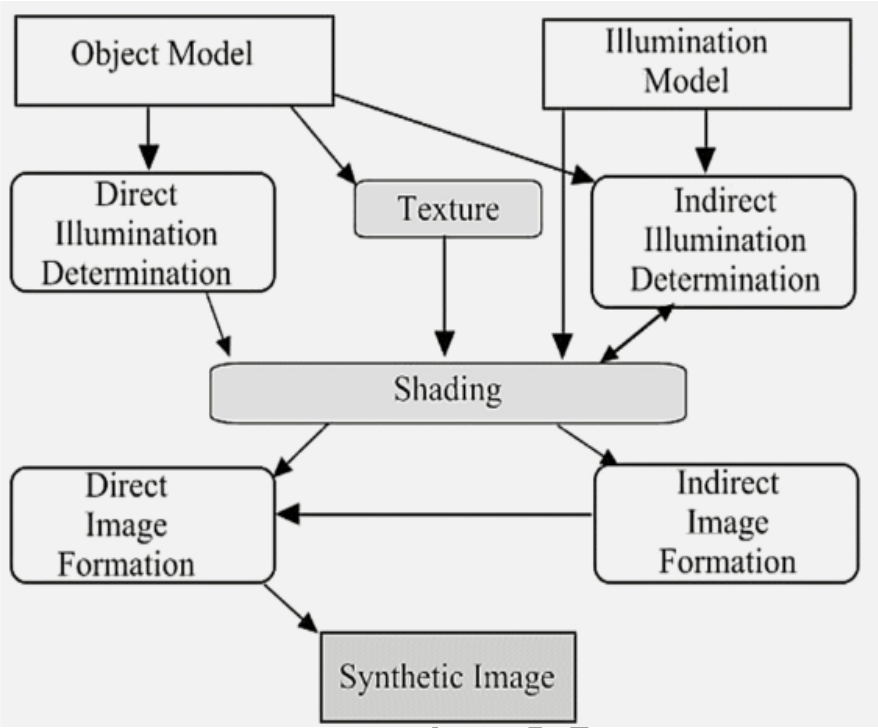


Image of Flow Chart of polygen rending methods

Gouraud shading

This Intensity-Interpolation scheme, developed by Gouraud and usually referred to as Gouraud Shading, renders a polygon surface by linear interpolating intensity value across the surface. Intensity values for each polygon are coordinate with the value of adjacent polygons along the common edges, thus eliminating the intensity discontinuities that can occur in flat shading.

Each polygon surface is rendered with Gouraud Shading by performing the following calculations:

1. Determining the average unit normal vector at each polygon vertex.
2. Apply an illumination model to each vertex to determine the vertex intensity.
3. Linear interpolate the vertex intensities over the surface of the polygon.

At each polygon vertex, we obtain a normal vector by averaging the surface normals of all polygons staring that vertex as shown in fig:

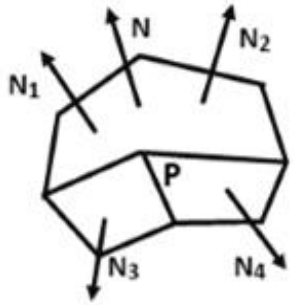


Fig: The normal vertex at vertex V is calculated as the average of surface normal for each polygon sharing the vertex.

Thus, for any vertex position V, we acquire the unit vertex normal with the calculation

$$\mathbf{N_V} = \frac{\sum_{k=1}^{n} \mathbf{N_k}}{\left|\sum_{k=1}^{n} \mathbf{N_k}\right|}$$

Once we have the vertex normals, we can determine the intensity at the vertices from a lighting model.

Following figures demonstrate the next step: Interpolating intensities along the polygon edges. For each scan line, the intensities at the intersection of the scan line with a polygon edge are linearly interpolated from the intensities at the edge endpoints. **For example:** In fig, the polygon edge with endpoint vertices at position 1 and 2 is intersected by the scanline at point 4. A fast method for obtaining the intensities at point 4 is to interpolate between intensities I_1 and I_2 using only the vertical displacement of the scan line.

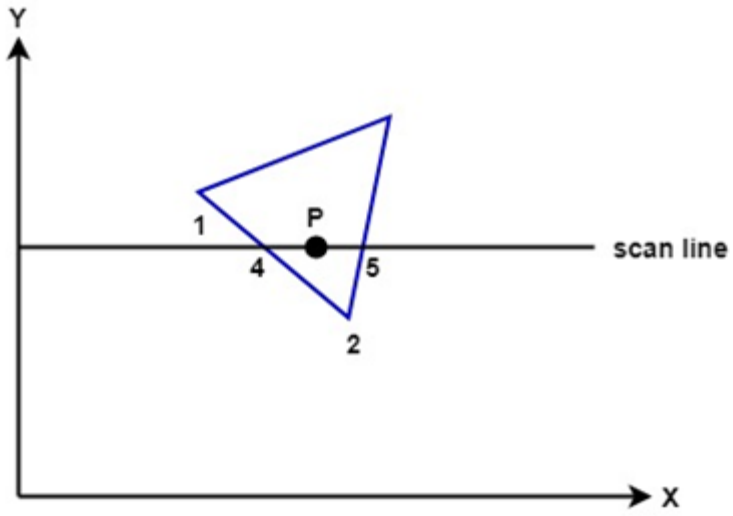


Fig: For Gouraud Shading, the intensity at point 4 is linearly interpolated from the intensities at vertices 1 and 2. The intensity at point 5 is linearly interpolated from intensities at vertices 2 and 3. An interior point P is then assigned an intensity value that is linearly interpolated from intensities at position 4 and 5.

$$I_4 = \frac{y_4 - y_2}{y_1 - y_2} I_1 + \frac{y_1 - y_4}{y_1 - y_2} I_2$$

Similarly, the intensity at the right intersection of this scan line (point 5) is interpolated from the intensity values at vertices 2 and 3. Once these bounding intensities are established for a scan line, an interior point (such as point P in the previous fig) is interpolated from the bounding intensities at point 4 and 5 as

$$I_p = \frac{x_5 - x_p}{x_5 - x_4} I_4 + \frac{x_p - x_4}{x_5 - x_4} I_5$$

Incremental calculations are used to obtain successive edge intensity values between scan lines and to obtain successive intensities along a scan line as shown in fig:

DIWAKAR EDUCATION

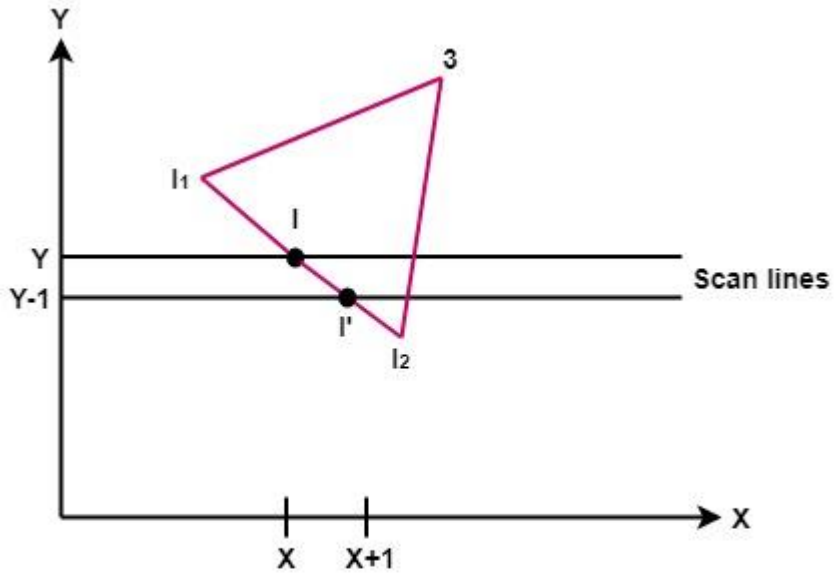


Fig: Incremental Interpolation of intensity value along a polygon edge for successive scan lines.

If the intensity at edge position (x, y) is interpolated as

$$I = \frac{Y-Y_2}{y_1-y_2} I_1 + \frac{Y_1-Y}{y_1-y_2} I_2$$

Then we can obtain the intensity along this edge for the next scan line, Y-1 as

$$I' = I + \frac{I_2-I_1}{y_1-y_2}$$

Similar calculations are used to obtain intensities at successive horizontal pixel positions along each scan line.

When surfaces are to be rendered in color, the intensities of each color component is calculated at the vertices. Gouraud Shading can be connected with a hidden-surface algorithm to fill in the visible polygons along each scan-line. An example of an object-shaded with the Gouraud method appears in the following figure:

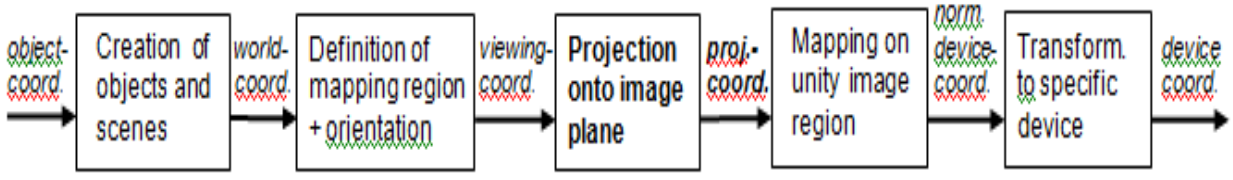


Fig: A Polygon mesh approximation of an object (a) is rendered with flat shading (b) and With Gouraud shading (c)

Gouraud Shading discards the intensity discontinuities associated with the constant-shading model, but it has some other deficiencies. Highlights on the surface are sometimes displayed with anomalous shapes, and the linear intensity interpolation can cause bright or dark intensity streaks, called Match bands, to appear on the surface. These effects can be decreased by dividing the surface into a higher number of polygon faces or by using other methods, such as Phong shading, that requires more calculations.

Viewing-Pipeline

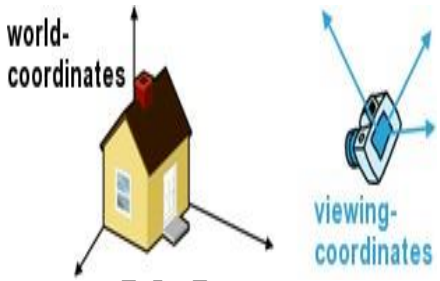
The viewing-pipeline in 3 dimensions is almost the same as the 2D-viewing-pipeline. Only after the definition of the viewing direction and orientation (i.e., of the camera) an additional projection step is done, which is the reduction of 3D-data onto a projection plane:



This projection step can be arbitrarily complex, depending on which 3D-viewing concepts should be used.

Viewing-Coordinates

Similar to photography there are certain degrees of freedom when specifying the camera:



1. Camera position in space
2. Viewing direction from this position
3. Orientation of the camera (view-up vector)
4. Size of the display window (corresponds to the focal length of a photo-camera)

With these parameters the *camera-coordinate system* is defined (viewing coordinates). Usually the xy-plane of

this viewing-coordinate system is orthogonal to the main viewing direction and the viewing direction is *in the direction of the negative z-axis*.

Based on the camera position the usual way to define the viewing-coordinate system is:

1. Choose a camera position (also called eye-point, or view-point).
2. Choose a viewing direction = Choose the z^- direction of the viewing-coordinates.
3. Choose a direction „upwards“. From this, the x-axis and y-axis can be calculated: the image-plane is orthogonal to the viewing direction. The parallel projection of the „view-up vector“ onto this image plane defines the y-axis of the viewing coordinates.
4. Calculate the x-axis as vector-product of the z- and y-axis.
5. The distance of the image-plane from the eye-point defines the viewing angle, which is the size of the scene to be displayed.

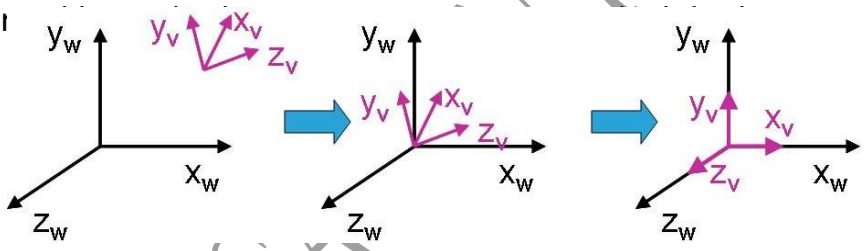
DIWAKAR EDUCATION H

In animations the camera-definition is often automatically calculated according to certain conditions, e.g. when the camera moves around an object or in flight-simulations, such that desired effects can be achieved in an uncomplicated way.

To convert world-coordinates to viewing-coordinates a series of simple transformations is needed: mainly a translation of the coordinate origins onto each other and afterwards 3 rotations, such that the coordinate-axes also coincide (two rotations for the first axis, one for the second axis, and the third axis is already correct then). Of course, all these

transformations can be mer
about like this:

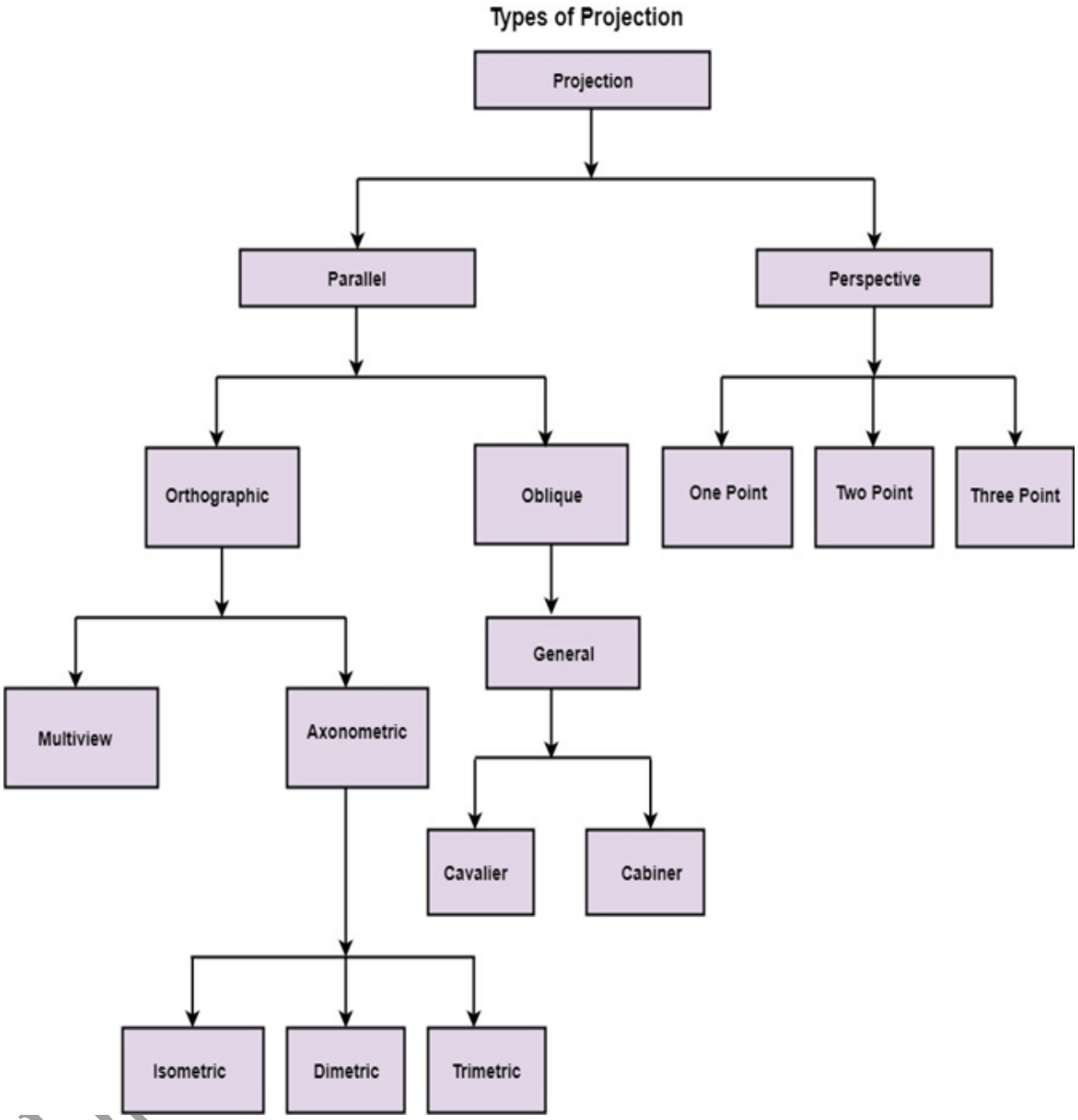
MWC,VC = Rz· Ry· Rx· T



Projection

It is the process of converting a 3D object into a 2D object. It is also defined as mapping or transformation of the object in projection plane or view plane. The view plane is displayed surface.

DIWAKAR EDUCATION HUB



Perspective Projection

In perspective projection farther away object from the viewer, small it appears. This property of projection gives an idea about depth. The artist use perspective projection from drawing three-dimensional scenes.

Two main characteristics of perspective are vanishing points and perspective foreshortening. Due to foreshortening object and lengths appear smaller from the center of projection. More we increase the distance from the center of projection, smaller will be the object appear.

Vanishing Point

It is the point where all lines will appear to meet. There can be one point, two point, and three point perspectives.

One Point: There is only one vanishing point as shown in fig (a)

Two Points: There are two vanishing points. One is the x-direction and other in the y -direction as shown in fig (b)

Three Points: There are three vanishing points. One is x second in y and third in two directions.

In Perspective projection lines of projection do not remain parallel. The lines converge at a single point called a center of projection. The projected image on the screen is obtained by points of intersection of converging lines with the plane of the screen. The image on the screen is seen as of viewer's eye were located at the centre of projection, lines of projection would correspond to path travel by light beam originating from object.

Important terms related to perspective

1. **View plane:** It is an area of world coordinate system which is projected into viewing plane.
2. **Center of Projection:** It is the location of the eye on which projected light rays converge.
3. **Projectors:** It is also called a projection vector. These are rays start from the object scene and are used to create an image of the object on viewing or view plane.

DIWAKAR EDUCATION HUB

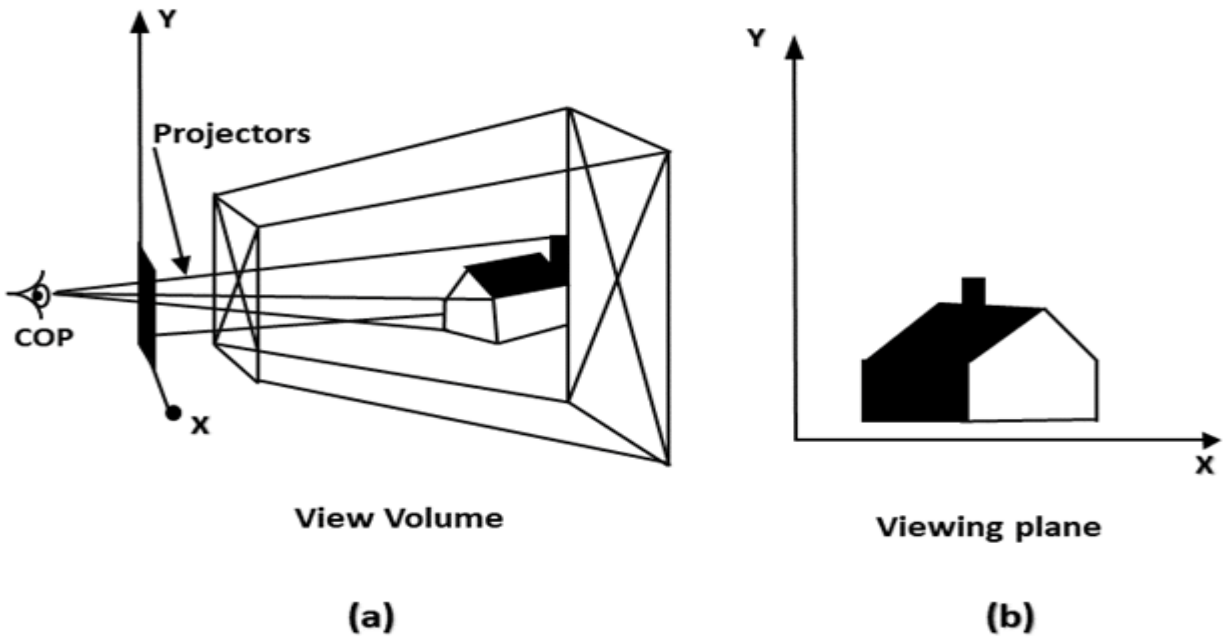


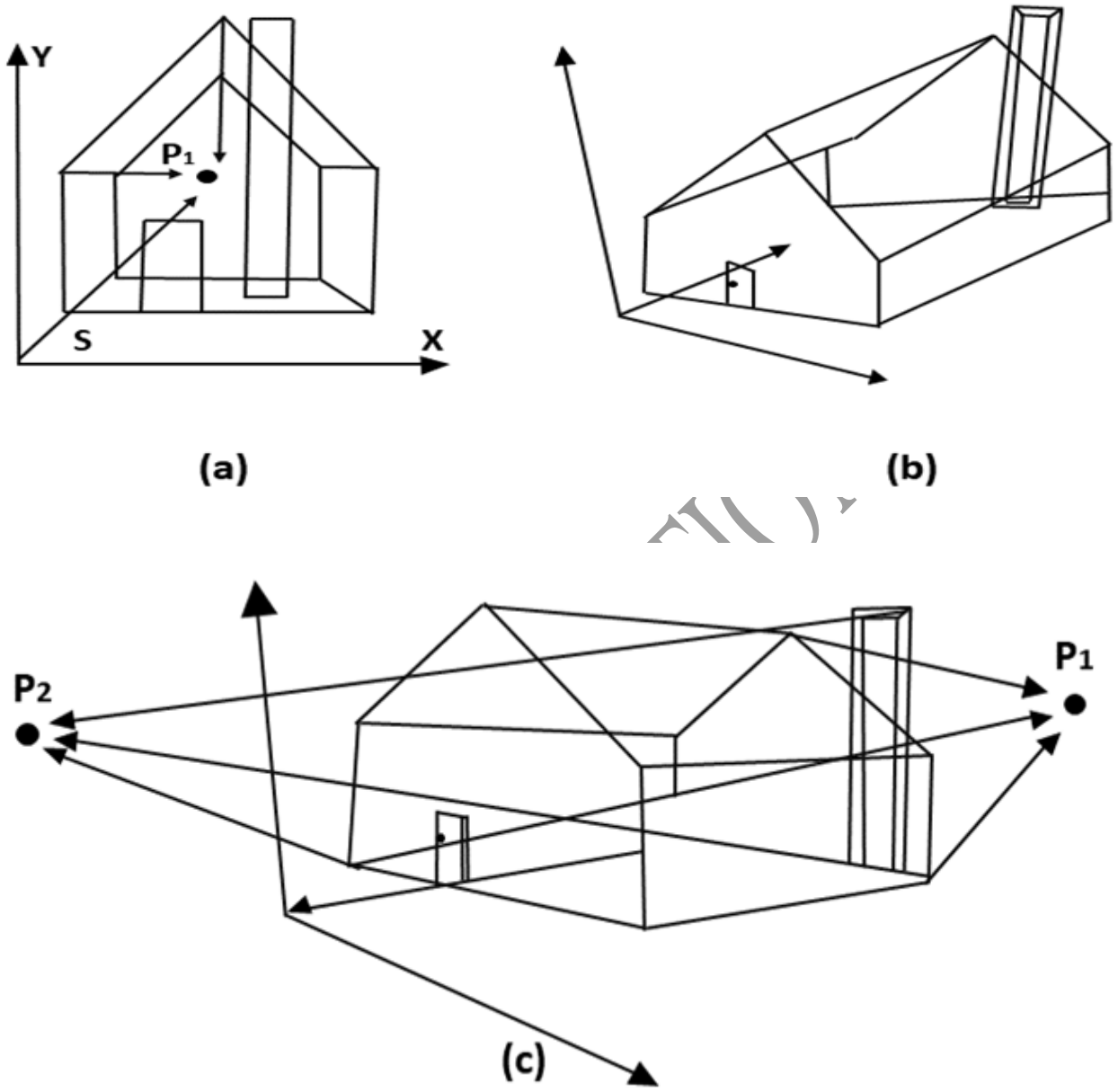
Fig: Prespective Projection

Anomalies in Perspective Projection

It introduces several anomalies due to these object shape and appearance gets affected.

1. **Perspective foreshortening:** The size of the object will be small of its distance from the center of projection increases.
2. **Vanishing Point:** All lines appear to meet at some point in the view plane.
3. **Distortion of Lines:** A range lies in front of the viewer to back of viewer is appearing to six rollers.

DIWAKAR EDUCA



Foreshortening of the z-axis in fig (a) produces one vanishing point, P_1. Foreshortening the x and z-axis results in two vanishing points in fig (b). Adding a y-axis foreshortening in fig (c) adds vanishing point along the negative y-axis.

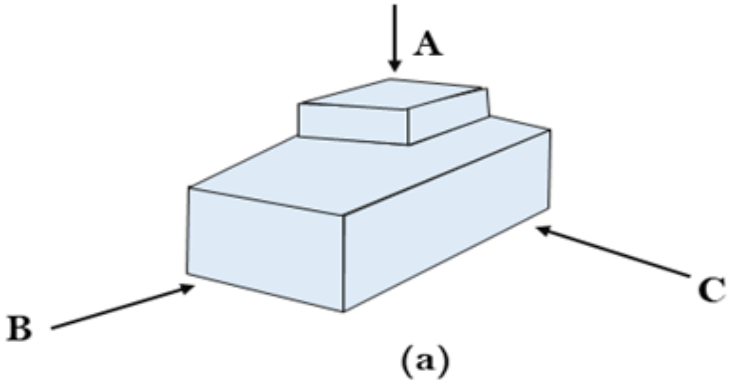
Parallel Projection

Parallel Projection use to display picture in its true shape and size. When projectors are perpendicular to view plane then is called **orthographic projection**.

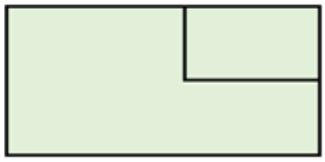
The parallel projection is formed by extending parallel lines from each vertex on the object until they intersect the plane of the screen. The point of intersection is the projection of vertex.

Parallel projections are used by architects and engineers for creating working drawing of the object, for complete representations require two or more views of an object using different planes.

DIWAKAR EDUCATION HUB

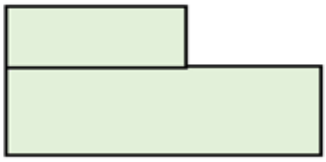


(a)



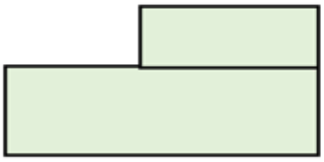
Parallel projection from top i.e. A direction

(b)



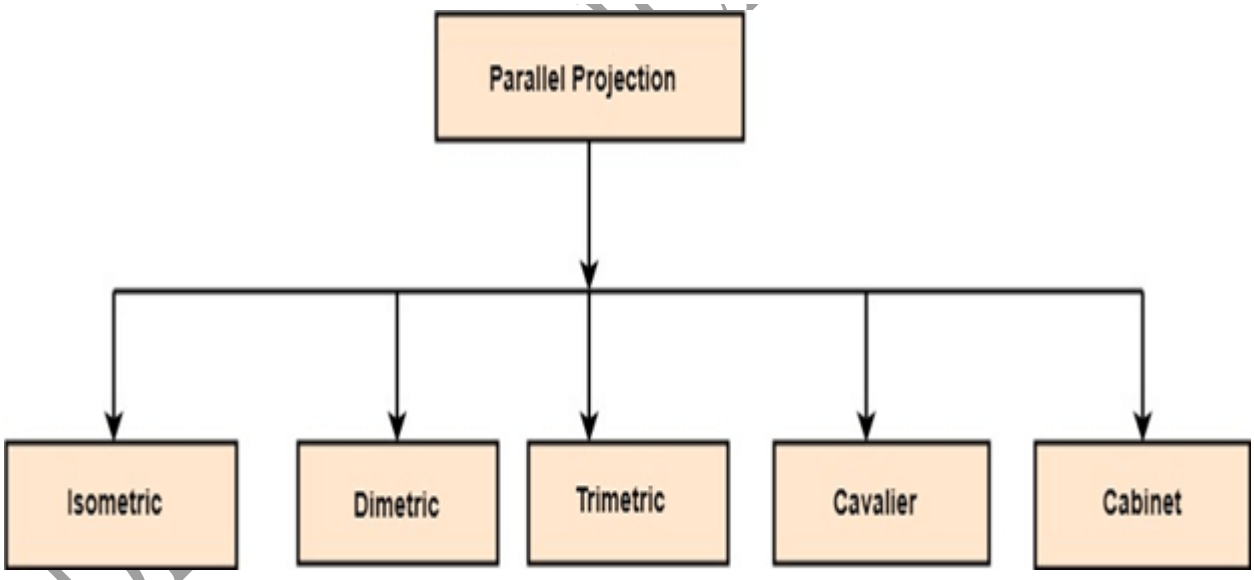
Parallel projection from top i.e. B direction

(c)



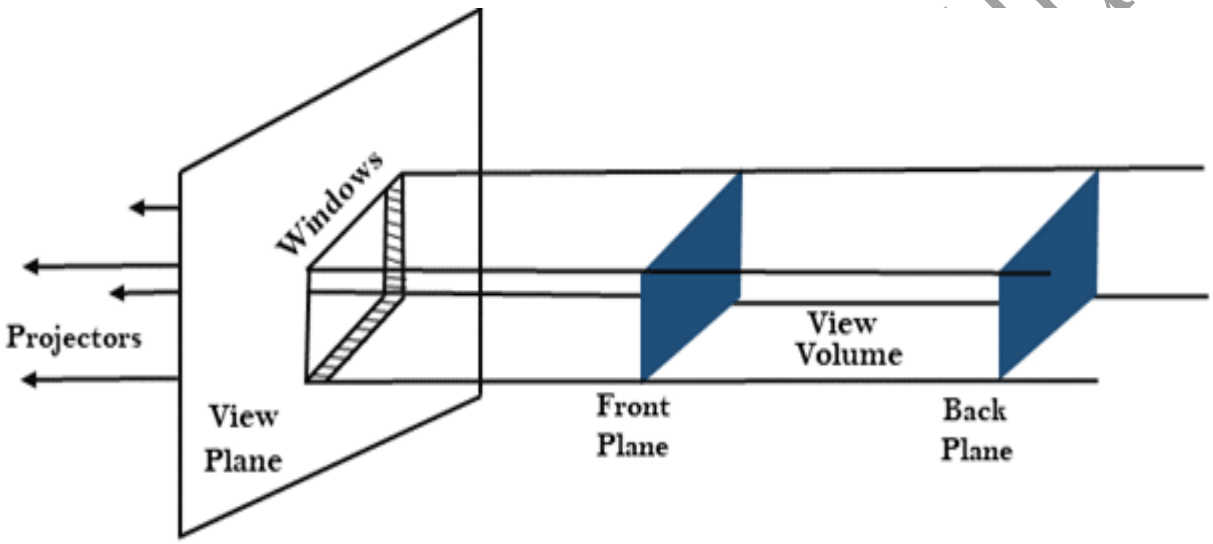
Parallel projection from top i.e. C direction

(d)



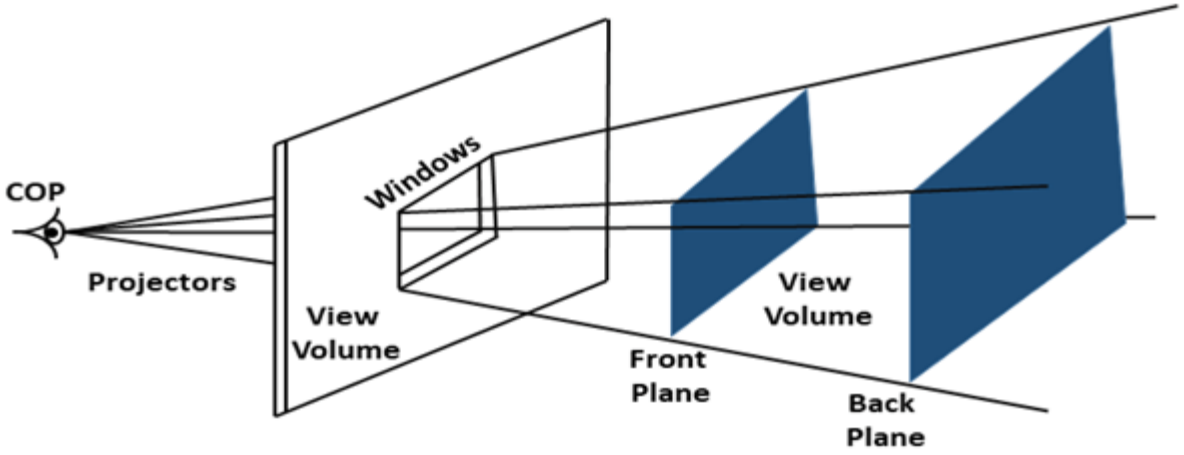
1. **Isometric Projection:** All projectors make equal angles generally angle is of 30°.
2. **Dimetric:** In these two projectors have equal angles. With respect to two principle axis.

3. **Trimetric:** The direction of projection makes unequal angle with their principle axis.
4. **Cavalier:** All lines perpendicular to the projection plane are projected with no change in length.
5. **Cabinet:** All lines perpendicular to the projection plane are projected to one half of their length. These give a realistic appearance of object.

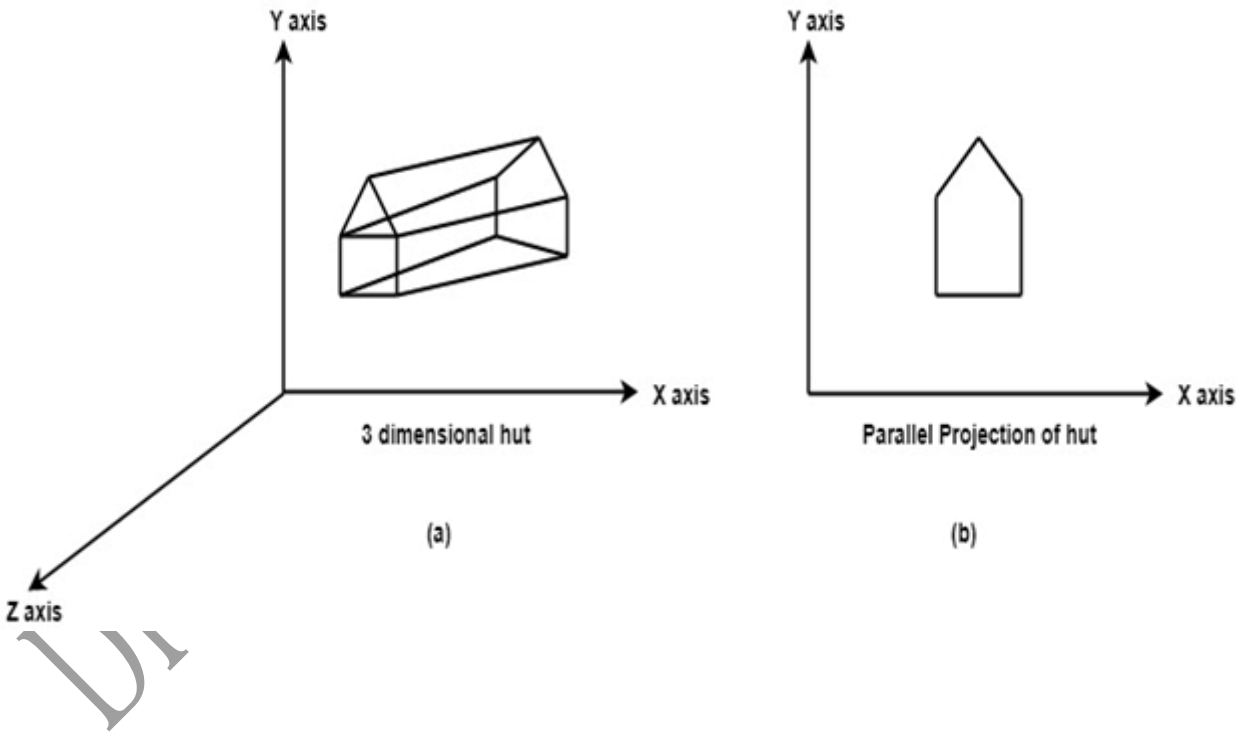


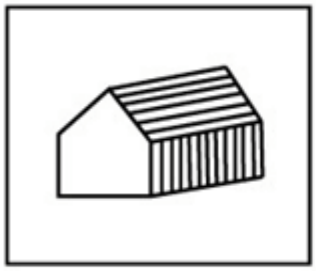
(a) Viewing Volume in orthographic projection

DIWAKAR

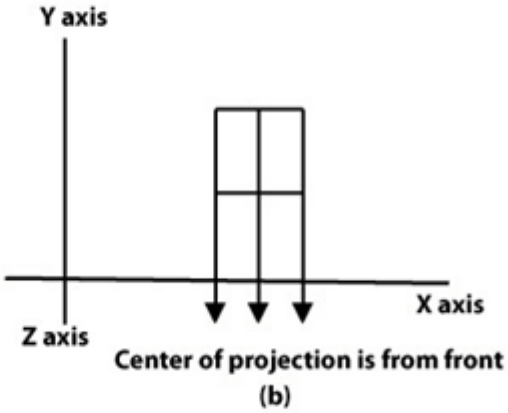


(b) Viewing volume in perspective projection

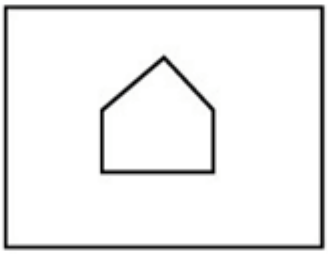




(a) Object original



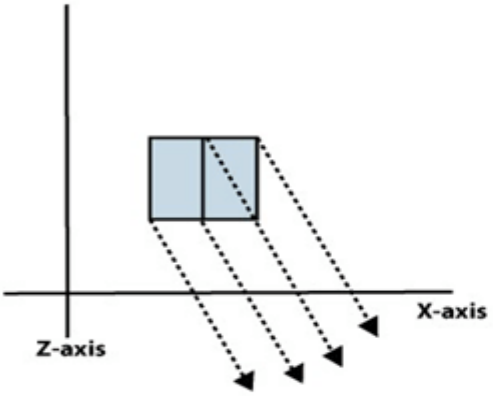
(b)



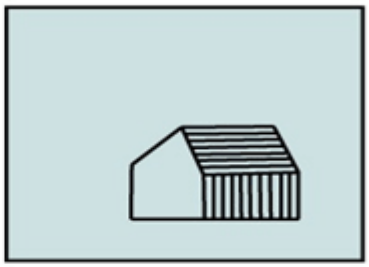
projected object

(c)

Fig (a) shows original object. Fig (b) shows object when projection is taken. Fig (c) gives projected object.



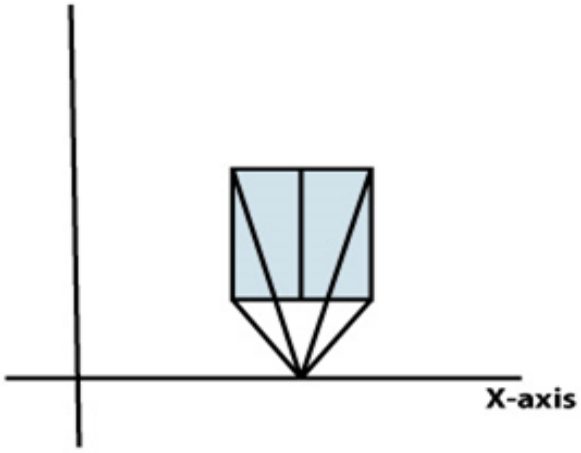
(d)



(e)

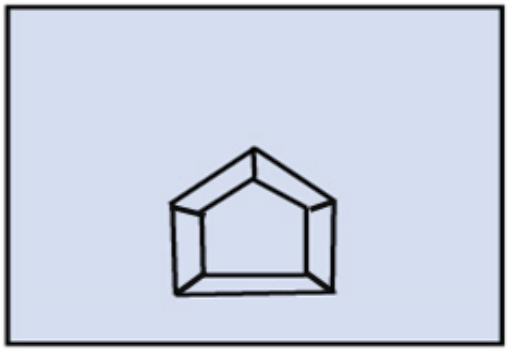
Fig (d) changes the direction of projection

Fig (e) shows object after changing direction of projection.

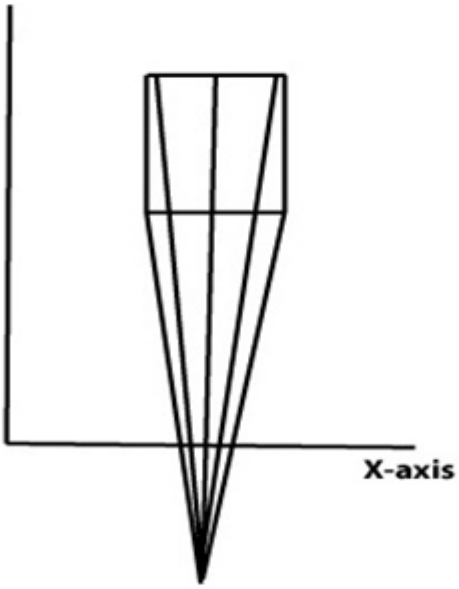


Center of projection meet at a chosen point

(a)



(b)



(a)



(b)

Frames of Reference

Computers will do exactly what We tell them to do. The question is about understanding what do we actually want to do and explaining that to a computer with enough rigor. For example we might want the computer to render a square, but this sentence is way too abstract for the computer to understand. More rigorous way to think about this is that we want lines between pairs of vertices (1,1), (−1,1), (−1,−1) and (1,−1), that are represented in the world space, to be drawn in a 400×300 area of the screen so that there is a linear mapping:

$$[-4,4]\times[-3,3]\rightarrow[0,400]\times[0,300].$$

This will already specify which pixels to color for our vertices. For example the vertex (1,1)(1,1) would be drawn on the pixel located at coordinates (250,200). Here We might notice a couple of problematic things:

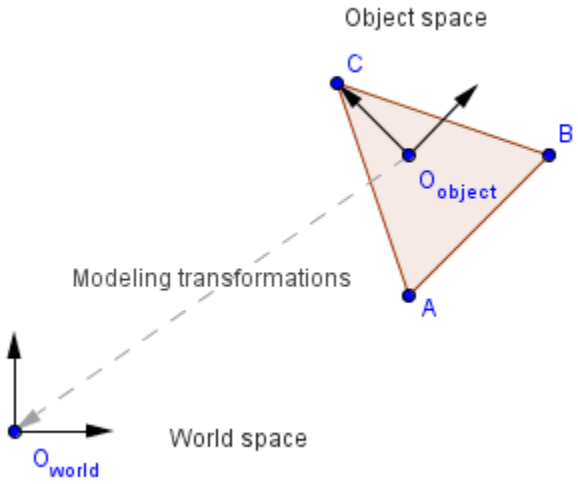
- Our geometry will be mirrored in the y direction, because in our world space y will point up, but in the screen space, y will point down.
- We are mapping from 2D to 2D. How to do it from 3D to 2D? What if we are not looking from the positive z direction?

On the other hand we might notice that with this kind of mapping our scene would always show the same amount of the scene if the resolutions changes. Although we might want to change that also.

There are actually many frames of references and transformations between our vertices and what we see on the screen. look at them one at a time.

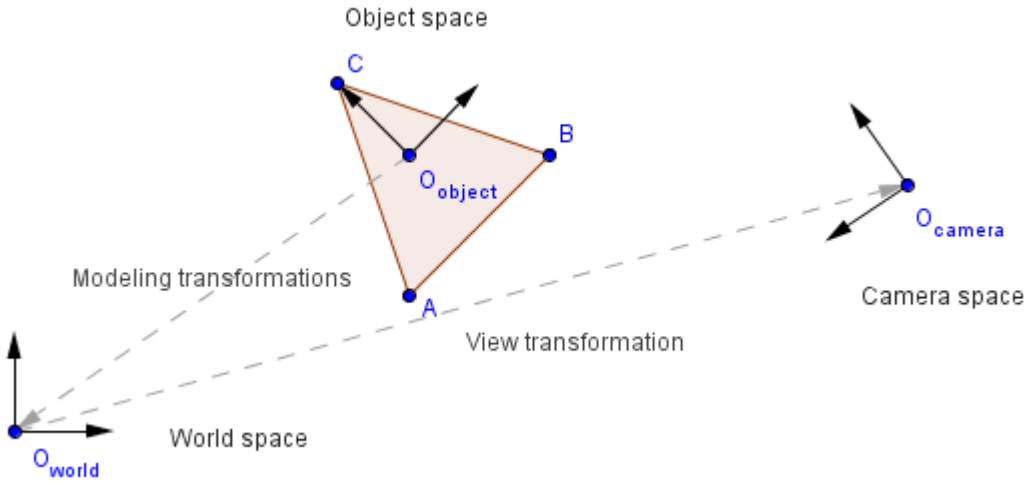
World Space

DIWAKAR EDUCATION HUB



First, our vertices are usually defined in the object space. We will try to center our object around some object origin (Object) that will serve as the scaling and rotation fixed point. Our object is located somewhere in the world space though, so we will use modelling transformations to represent the vertices with world space coordinates. Imagine if there are multiple triangles, they can not all be located in the world space origin, they are probably scattered all around the world space, and thus each have distinct world space coordinates. Although they might have the same object space (local) coordinates.

Camera Space



Secondly, there is a camera somewhere in the world space. This camera has its own coordinates in the world space and is transformed just like any other object / vertex. Our goal here is to position everything into camera space so that camera and world frames match. In other words, transform everything so that camera will be located at the world origin and the axes will be aligned.

Usually there is a function called *lookAt()*, where We can say where in the world is our camera located, which is the up direction and to what point is it pointed (is looking) at. This is a convenience function that underneath will construct the corresponding matrix that will do the camera transformation in this step. see, how this can be done.

We know:

- Camera's position in world space. Call it pp.
- Camera's up-vector. Call it uu.
- Camera's looking direction. Call it ll.

Well, the first step would be to align the origins. So we want to move all our objects so that the camera would be in the world origin. We had a translation pp that moved camera away from the origin, so the reverse would be just −p.

So the first transformations would be:

$$\begin{pmatrix} 1 & 0 & 0 & -p_x \\ 0 & 1 & 0 & -p_y \\ 0 & 0 & 1 & -p_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Next we need to align the axes. Now, the vector ll shows the direction we are facing, but because the z-axis should be in the negative direction, we will reverse it: w=−l. We assume that vectors uu and ww are orthogonal to each other and of unit length. We still need one orthogonal vector, the right vector, in order to have

DIWAKAR EDUCATION HUB

the camera basis. We could use the Gram-Schmidt process here, but a more simpler way to find an orthogonal unit vector, given already two, would be to use the cross product of vectors. So v=w×u. Notice that the cross product is not commutative and will follow the right-hand rule. The vector vv will emerge from the plane defined by ww and uu in the direction where the angle from w to u will be counter-clockwise. If We exchange ww and uu in the cross product, then the result will point to the opposite direction.

Now we have the vectors u, v and w that are the basis of the camera's frame of reference. The transformation that would transform the camera basis from the world basis (ignoring the translation for now) would be:

$$\begin{pmatrix} u_x & v_x & w_x & 0 \\ u_y & v_y & w_y & 0 \\ u_z & v_z & w_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Remember that we can look at the columns of the matrix to determine how the basis would transform. But that would be the transformation which would transform the camera space (coordinates) to the world space. We need to find a transformation to get from world space to camera space. Or in other words, transform the camera basis to be the current world space basis. We might remember from algebra that the inverse of an orthogonal matrix is the transpose of it. All rotation matrices are orthogonal matrices and a multiplication of two rotation matrices is a rotation matrix. Our camera basis will differ from the world basis only by a number of rotations. So in order to find out how to transform the world so that the camera basis would be aligned with the world basis, we only need to take the inverse of those rotation. That is the inverse of the matrix we just found. The inverse is currently just the transpose of it.

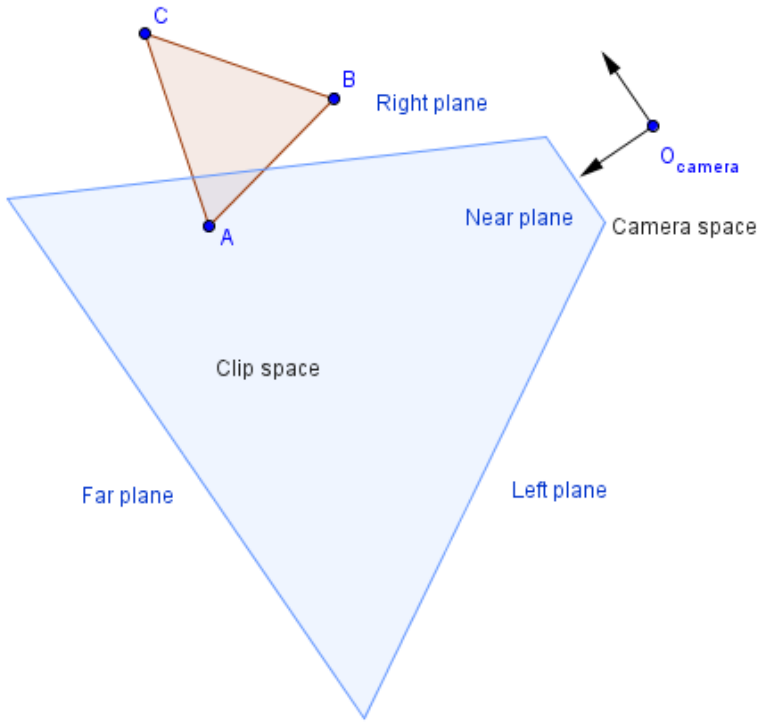
The total view transformation would be then:

UNKARED...ION HUB

$$V = \begin{pmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ w_x & w_y & w_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & -p_x \\ 0 & 1 & 0 & -p_y \\ 0 & 0 & 1 & -p_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

In some sources the view transformation is called the inverse of the camera transformation. There the camera transformation is the transformation that transforms the actual camera as an object. In other sources the camera transformation is the transformation that transforms all the vertices into the camera space. One should explain what is actually meant when using those terms. The term view transformation (the view matrix) is generally more common.

Clip Space



DIW EDUCATION HUB

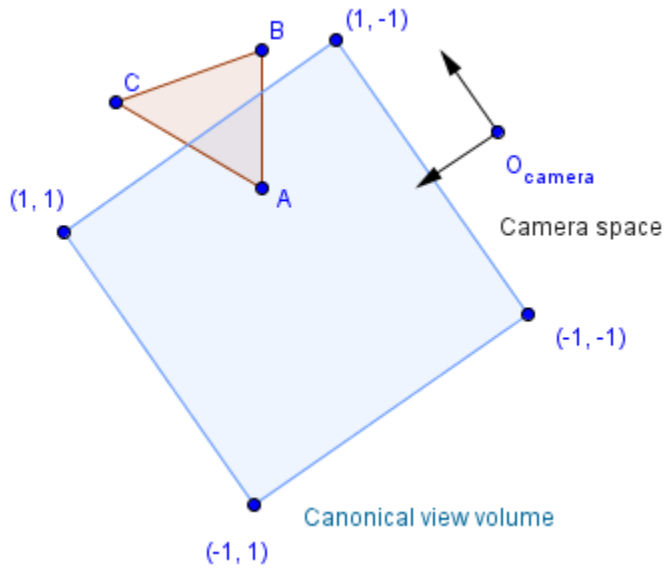
Third step is to construct the clip space. This is the space that should be seen from the camera. In our illustration here there is a 2D projection of it. In 3D We will also have a top and a bottom plane. Based on the transformation We might have to specify those plains directly or, as is the case with perspective projection, We can specify a field of view parameter that will calculate those itself.

Everything outside this clip space is clipped, i.e. ignored in rendering. It might depend on what kind of clipping techniques are in use. For example if a line crosses the clip space, then it might be segmented: another vertex is created at the border of the clip space. Objects that are wholly outside of the clip space, are ignored. This is one place where we want to be able to determine the rough locations of our objects quickly. Later we will see different methods (e.g. binary space partitioning, bounding volume hierarchy) that help to determine that. Mathematically We can think that the equations of the planes are constructed and intersections with other planes (ones defined by all of the triangles in the scene) are checked for. Of course, if we are only interested about vertices, then we can put every vertex in the plane equation and see which side of the plane a vertex lies.

We will look this and the next step in more detail when talking about different projections.

Canonical View Volume

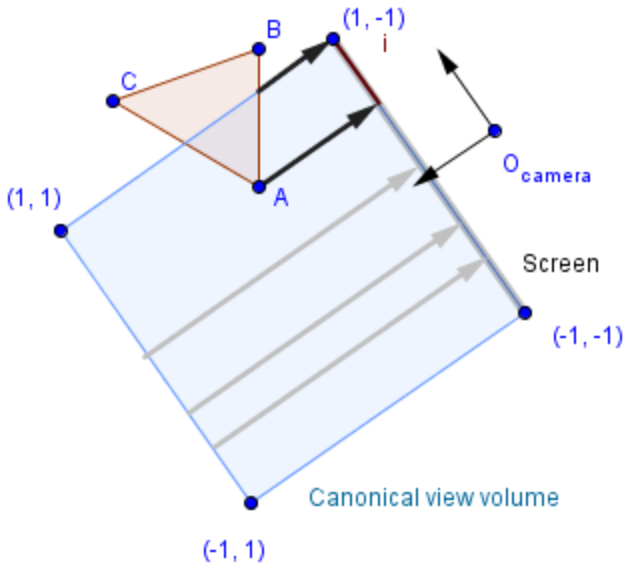
DIWAKAR EDUCATION HUB



Fourth step is to transform everything so that the clip space will form a canonical view volume. This canonical view volume is a cube in the coordinates [−1,1]×[−1,1]×[−1,1]. As we can see from the previous illustration we had kind of a weird shape for the clip space. This shape is a frustum of a pyramid in the case of perspective projection. In computer graphics we call it the view frustum. Transforming that into a cube will transform the whole space in a very affecting way. This is the part where we leave the world of affine transformations and perform a perspective projection transformation. It is actually quite simple, we will just put a non-zero value in the third column of the last row of our transformation matrix. As we remember, affine transformations needed the last row to be (0,0,0,1), but for perspective transformation we will have (0,0,p,0). Depending on how an API is implemented, we might have p=1 or p=−1 or even something else. After this transformation, there is a need for the perspective division, because the value of w will usually no longer be 1. Remember that in homogeneous coordinates (x,y,z,w)=(x/w,y/w,z/w).

As We can see, coordinates will be divided based on the value of w, which will depend on the value of z. This actually causes the objects further away to be projected smaller then objects that are closer to the camera - the same way we see the world.

Screen Space



Final step is to project the canonical view volume into screen space. Our screen has usually mapped the coordinates of pixels so that (0,0)(0,0) is in the top-left corner and positive directions are to the right and down. Depending on the actually screen resolution (or window / viewport size if we are not rendering full-screen), we will have to map the canonical view volume coordinates to different ranges. In either way, this will just be a linear mapping from the ranges $[-1,1]\times[-1,1] \rightarrow [0,width]\times[0,height]$ like we saw in the beginning.

This mapping is called the viewport transformation and the matrix that does it is quite simple. This step is usually done automatically. For the proportional linear mapping from one range into another we just need to scale the initial range to be of the same length as the final range, and then translate it so that they cover each other.

DWAKA EDUCA HUB

The length of the range $[-1, 1]$ is 2. For the length of the range $[0, width]$ we will take *width*. This is because we want the value -1 mapped to 0 and the value 1 mapped to *width*. So the scaling should be by $\frac{width}{2}$, after which we will have a range $[-\frac{width}{2}, \frac{width}{2}]$. Then we will just translate by $\frac{width}{2}$ and end up with a range $[0, width]$.

$$\begin{pmatrix} x_{screen} \\ y_{screen} \\ z_{canonical} \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{width}{2} & 0 & 0 & \frac{width}{2} \\ 0 & \frac{height}{2} & 0 & \frac{height}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_{canonical} \\ y_{canonical} \\ z_{canonical} \\ 1 \end{pmatrix}$$

Notice that the z value is brought along here. This is useful in order to later determine which objects are in front of other objects. Also the y values still seem to be reversed. This will be handled later by OpenGL, currently we just assume that the (0,0) in the screen space is at the bottom-left corner and the positive directions are up and right. Also the z value will later be normalized to a range [0,1].

DIWAKAR EDUCATION

DIWAKAR EDUCATION HUB

Programming Languages and Computer Graphics Unit – 3 MCQs

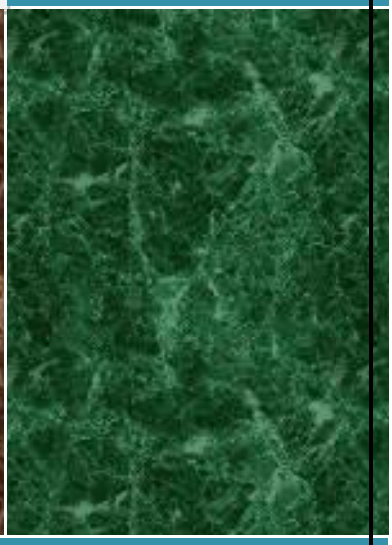
As per updated syllabus

DIWAKAR EDUCATION HUB



2020





THE LEARN WITH EXPERTIES

1) What is the 16-bit compiler allowable range for integer constants?

a) -3.4e38 to 3.4e38
b) -32767 to 32768
c) -32668 to 32667
d) -32768 to 32767

Answer: (d) -32768 to 32767

Explanation: In a 16-bit C compiler, we have 2 bytes to store the value.

- The range for signed integers is -32768 to 32767.
- The range for unsigned integers is 0 to 65535.
- The range for unsigned character is 0 to 255.

2) Study the following program:

```
1. main()
2. {printf("javatpoint");
3. main();}
```

What will be the output of this program?

a. Wrong statement

b. It will keep on printing javatpoint

c. It will Print javatpoint once

d. None of the these

Answer: (b) It will keep on printing javatpoint

Explanation: In this program, the main function will call itself again and again. Therefore, it will continue to print javatpoint.

3) What is required in each C program?

a. The program must have at least one function.

b. The program does not require any function.

c. Input data

d. Output data

Answer: (a) The program must have at least one function.

Explanation: Any C program has at least one function, and even the most trivial programs can specify additional functions. A function is a piece of code. In other words, it works like a sub-program.

4) What will this program print?

```
1. main()
2. {
3.   int i = 2;
4.   {
5.     int i = 4, j = 5;
6.      printf("%d %d", i, j);
7.   }
8.   printf("%d %d", i, j);
9. }
```

a. 4525

b. 2525

c. 4545

d. None of the these

Answer: (a) 4525

Explanation: In this program, it will first print the inner value of the function and then print the outer value of the function.

5) Which of the following comment is correct when a macro definition includes arguments?

a. The opening parenthesis should immediately follow the macro name.

b. There should be at least one blank between the macro name and the opening parenthesis.

c. There should be only one blank between the macro name and the opening parenthesis.

d. All the above comments are correct.

Answer: (a) The opening parenthesis should immediately follow the macro name.

Explanation: None

6) What is a lint?

a. C compiler

b. Interactive debugger

c. Analyzing tool

d. C interpreter

Answer: (c) Analyzing tool

Explanation: Lint is an analyzing tool that analyzes the source code by suspicious constructions, stylistic errors, bugs, and flag programming errors. Lint is a compiler-like tool in which it parses the source files of C programming. It checks the syntactic accuracy of these files.

7) What is the output of this statement "printf("%d", (a++))"?

a. The value of (a + 1)

b. The current value of a

c. Error message

d. Garbage

Answer: (b) The current value of "a".

Explanation: None

8) Study the following program:

```
1. main()
2. {
3.    char x [10], *ptr = x;
4.    scanf ("%s", x);
5.    change(&x[4]);
6. }
7.  change(char a[])
8.  {
9.    puts(a);
10. }
```

If abcdefg is the input, the output will be

a. abcd

b. abc

c. efg

d. Garbage

Answer: (c) efg

Explanation: None

---

9) Study the following program:

```
1. main()
2. {
3.   int a = 1, b = 2, c = 3:
4.   printf("%d", a + = (a + = 3, 5, a))
5. }
```

What will be the output of this program?

a. 6

b. 9

c. 12

d. 8

Answer: (d) 8

Explanation: It is an effect of the comma operator.

a + = (a + = 3, 5, a)

It first evaluates to "a + = 3" i.e. a = a + 3 then evaluate 5 and then evaluate "a".

Therefore, we will get the output is 4.

Then,

a + = 4

It gives 8 as the output.

10) What does this declaration mean?

```
1. int x : 4;
```

a. X is a four-digit integer.

b. X cannot be greater than a four-digit integer.

c. X is a four-bit integer.

d. None of the these

Hide Answer Workspace

Answer: (c) X is a four-bit integer.

Explanation: This means, "X" is a four bit integer.

11) Why is a macro used in place of a function?

a. It reduces execution time.

b. It reduces code size.

c. It increases execution time.

d. It increases code size.

Answer: (d) It reduces code size.

Explanation: Macro is used in place of a function because it reduces code size, and very efficient.

12) In the C language, the constant is defined _____.

a. Before main

b. After main

c. Anywhere, but starting on a new line.

d. None of the these.

Answer: (c) Anywhere, but starting on a new line.

Explanation: In the C language, the constant is defined anywhere, but starting on a new line.

13) How many times will the following loop execute?

1. **for**(j = 1; j <= 10; j = j-1)

a. Forever

b. Never

c. 0

d. 1

Answer: (a) Forever

Explanation: None

14) A pointer is a memory address. Suppose the pointer variable has p address 1000, and that p is declared to have type int*, and an int is 4 bytes long. What address is represented by expression p + 2?

a. 1002

b. 1004

c. 1006

d. 1008

Answer: (d) 1008

Explanation: None

15) What is the result after execution of the following code if a is 10, b is 5, and c is 10?

1. If ((a > b) && (a <= c))
2. a = a + 1;
3. **else**
4. c = c+1;

a. a = 10, c = 10

b. a = 11, c = 10

c. a = 10, c = 11

d. a = 11, c = 11

Answer: (b) a = 11, c = 10

Explanation: None

16) Which one of the following is a loop construct that will always be executed once?

a. for

b. while

c. switch

d. do while

Answer: (d) do while

Explanation: The body of a loop is often executed at least once during the do-while loop. Once the body is performed, the condition is tested. If the condition is valid, it will execute the body of a loop;

otherwise, control is transferred out of the loop.

17) Which of the following best describes the ordering of destructor calls for stack-resident objects in a routine?

a. The first object created is the first object destroyed; last created is last destroyed.

b. The first object destroyed is the last object destroyed; last created is first destroyed.

c. Objects are destroyed in the order they appear in memory, the object with the lowest memory address is destroyed first.

d. The order is undefined and may vary from compiler to compiler.

Answer: (b) The first object destroyed is the last object destroyed; last created is first destroyed.

Explanation: None

18) How many characters can a string hold when declared as follows?

1. **char** name[20]:

a. 18

b. 19

c. 20

d. None of the these

Answer: (b) 20

Explanation: None

19) Directives are translated by the

a. Pre-processor

b. Compiler

c. Linker

d. Editor

Answer: (a) Pre-processor

Explanation: In C language, the pre-processor is a macro processor that is dynamically used by the C programmer to modify the program before it is properly compiled (Before construction, pro-processor directives are implemented).

20) How many bytes does "int = D" use?

a. 0

b. 1

c. 2 or 4

d. 10

Answer: (c) 2 or 4

Explanation: The int type takes 2 or 4 bytes.

21) What feature makes C++ so powerful?

a. Easy implementation

b. Reusing the old code

c. Easy memory management

d. All of the above

Answer: (d) All of the above

Explanation: None

22) Which of the following will copy the null-terminated string that is in array src into array dest?

a. dest = src;

b. dest == src;

c. strcpy(dest, src);

d. strcpy(src, dest);

Answer: (c) strcpy(dest, src)

Explanation: strcpy is a string function that is used to copy the string between the two files. strcpy(destination, source)

23) In the statement "COUT << "javatpoint" << end1;", end1 is a ___.

a. Extractor

b. Inserter

c. Manipulator

d. Terminator

Answer: (c) Manipulator

Explanation: End1 is an I/O manipulator that takes effect in printing a new line '\ n' character and then flushing the output stream.

24) Each instance of a class has a different set of

a. Class interfaces

b. Methods

c. Return types

d. Attribute values

Answer: (d) Attribute values

Explanation: Each instance of the class has a different set of attribute values

25) How many instances of a class can be declared?

a. 1

b. 10

c. As per required

d. None of the these

Answer: (c) As per required

Explanation: You can always declare multiple instances of a class, as per required. Each object will hold its own individual inner variables (unless they are static, in which case they are shared).

26) What will the result of num variable after execution of the following statements?

1. **int** num = 58;
2. num % = 11;

a. 3

b. 5

c. 8

d. 11

Answer: (a) 3

Explanation: num = 58

num % = 11

num = num % 11

num = 58 % 11

num = 3

27) What is the maximum number of characters that can be held in the string variable char address line [40]?

a. 38

b. 39

c. 40

d. 41

Answer: (b) 39

Explanation: None

28) What will the result of num1 variable after execution of the following statements?

```
1. int j = 1, num1 = 4;
2. while (++j <= 10)
3. {
4.   num1++;
5. }
```

a. 11

b. 12

c. 13

d. 14

Answer: (c) 13

Explanation: None

29) What will the result of len variable after execution of the following statements?

```
1. int len;
2. char str1[] = {"39 march road"};
3. len = strlen(str1);
```

a. 11

b. 12

c. 13

d. 14

Answer: (c) 13

Explanation: strlen is a string function that counts the word and also count the space in the string. (39 march road) = 13

30) Study the following statement

```
1. #include <stdio.h>
2.    int main()
3.    {
4.       int *ptr, a = 10;
5.       ptr = &a;
6.       *ptr += 1;
7.       printf("%d,%d/n", *ptr, a);
8.    }
```

What will be the output?

a. 10, 10

b. 10, 11

c. 11, 10

d. 11, 11

Answer: (d) 11, 11

Explanation: None

31) Given the following statement, what will be displayed on the screen?

1. **int** * aPtr;
2. *aPtr = 100;
3. cout << *aPtr + 2;

a. 100

b. 102

c. 104

d. 108

Answer: (b) 102

Explanation: aPtr is an integer pointer which value is 100.

= *aPtr + 2

= 100 + 2

= 102

32) Give the following declarations and an assignment statement. Which one is equivalent to the expression str [4]?

1. **char** str[80];
2. **char** * p;
3. p = str;

a. p + 4

b. *p + 4

c. *(p + 4)

d. p [3]

Answer: (c) *(p + 4)

Explanation: None

33) Which one is the correct description for the variable balance declared below?

1. **int** ** balance;

a. Balance is a point to an integer

b. Balance is a pointer to a pointer to an integer

c. Balance is a pointer to a pointer to a pointer to an integer

d. Balance is an array of integer

Answer: (b) Balance is a pointer to a pointer to an integer

Explanation: This code description states that the remainder is a pointer to a pointer to an integer.

34) A class D is derived from a class B, b is an object of class B, d is an object of class D, and pb is a pointer to class B object. Which of the following assignment statement is not valid?

a. d = d;

b. b = d;

c. d = b;

d. *pb = d:

Answer: (c) d = b;

Explanation: A class D is derived from a class B, so "d" is not equal to b.

35) Which of the following statement is not true?

a. A pointer to an int and a pointer to a double are of the same size.

b. A pointer must point to a data item on the heap (free store).

c. A pointer can be reassigned to point to another data item.

d. A pointer can point to an array.

Answer: (b) A pointer must point to a data item on the heap (free store).

Explanation: None

36) Which of the following SLT template class is a container adaptor class?

a. Stack

b. List

c. Deque

d. Vector

Answer: (a) Stack

Explanation: Container Adaptors is the subset of Containers that provides many types interface for sequential containers, such as stack and queue.

37) What kinds of iterators can be used with vectors?

a. Forward iterator

b. Bi-directional iterator

c. Random access iterator

d. All of the above

Answer: (d) All of the above

Explanation: An iteration is like a pointer, indicating an element inside the container. All these types of iterations can be used with vectors.

38) Let p1 be an integer pointer with a current value of 2000. What is the content of p1 after the expression p1++ has been evaluated?

a. 2001

b. 2002

c. 2004

d. 2008

Answer: (c) 2004

Explanation: The size of one pointer integer is 4 bytes. The current value of p1 is 2000.

p1++ = p1 + 1

p1++ = 2004

39) Let p1 and p2 be integer pointers. Which one is a syntactically wrong statement?

a. p1 = p1 + p2;

b. p1 = p1 - 9;

c. p2 = p2 + 9;

d. cout << p1 - p2;

Answer: (a) p1 = p1 + p2;

Explanation: None

40) Suppose that cPtr is a character pointer, and its current content is 300. What will be the new value in cPtr after the following assignment?

1. cPtr = cPtr + 5;

a. 305

b. 310

c. 320

d. 340

Hide Answer Workspace

Answer: (a) 305

Explanation: cPtr = cPtr + 5

cPtr = 300 + 5

cPtr = 305

41) Which is valid expression in c language?

a. int my_num = 100,000;

b. int my_num = 100000;

c. int my num = 1000;

d. int my num == 10000;

Answer: (b) int my_num = 100000;

Explanation: Special symbol, Space, and comma cannot be used in a variable name in c language.

42) If addition had higher precedence than multiplication, then the value of the expression (1 + 2 * 3 + 4 * 5) would be which of the following?

a. 27

b. 47

c. 69

d. 105

Hide Answer Workspace

Answer: (d) 105

Explanation: (1 + 2 * 3 + 4 * 5)

= (1 + 2) * (3 + 4) * 5

= 3 * 7 * 5

= 105

43) What will be the output of this program?

```
1. int main()
2. {
3. int a=10, b=20;
4. printf("a=%d b=%d",a,b);
```

```
5. a=a+b;
6. b=a-b;
7. a=a-b;
8. printf("a=%d b=%d",a,b);
9. return 0;
10. }
```

a. a = 20, b = 20

b. a = 10, b = 20

c. a = 20, b = 10

d. a = 10, b = 10

Answer: (c) a = 20, b = 10

Explanation: This program is a swapping program.

a = a + b → a = 10 + 20 → a = 30

b = a - b → b = 30 - 20 → B = 10

a = a - b → a = 30 - 10 → a = 20

44) The following statements are about EOF. Which of them is true?

a. Its value is defined within stdio.h

b. Its value is implementation dependent

c. Its value can be negative

d. Its value should not equal the integer equivalent of any character

e. All of the these

Answer: (e) All of the these

Explanation: All statements are true

45) What does this statement mean?

```
1. x - = y + 1;
```

a. x = x - y + 1

b. x = -x - y - 1

c. x = x + y - 1

d. x = x - y - 1

Answer: (d) x = x - y - 1

Explanation: x - = y + 1

x = x - (y + 1)

So, x = x - y - 1

46) Study the following statement

```
1. for (i = 3; i < 15; i + = 3)
2. {printf ("%d", i);
3. ++i;
4. }
```

What will be the output?

a. 3 6 9 12

b. 3 6 9 12 15

c. 3 7 11

d. 3 7 11 15

Answer: (c) 3 7 15

Explanation: None

47) Study the following statement

```
1. main()
2. {
3.    char *s = "Hello,"
4.    "World!";
5.    printf("%s", s);
6. }
```

What will be the output?

a. Hello, World!

b. Hello,
World!

c. Hello

d. Compile error

Answer: (b) Hello, World!

Explanation: The output of this program is "Hello, World!". This program's output will not appear in the new line because the \ n escape sequence has not been used in this program.

48) Study the following array definition

```
1. int num[10] = {3, 3, 3};
```

Which of the following statement is correct?

a. num[9] is the last element of the array num

b. The value of num[8] is 3

c. The value of num[3] is 3

d. None of the above

Answer: (a) num[9] is the last element of the array num

Explanation: The num[9] is the last element of the array number because the total element in this array is 10, and the array starts with 0, so the last element of the array is the num[9].

49) What will the output after execution of the following statements?

```
1. main()
2. {
3.    printf ("\\n ab");
4.    printf ("\\b si");
5.    printf ("\\r ha");
6. }
```

a. absiha

b. asiha

c. haasi

d. hai

Answer: (d) hai

Explanation:

- \\n - newline - printf("\\nab"); - Prints 'ab'
- \\b - backspace - printf("\\bsi"); - firstly '\\b' removes 'b' from 'ab ' and then prints 'si'. So, after execution of printf("\\bsi"); it is 'asi'

- \\r - linefeed - printf("\\rha"); - Now here '\\r' moves the cursor to the start of the current line and then override 'asi' to 'hai'

50) What will the output after execution of the following statements?

```
1. void main()
2. {
3.   int i = 065, j = 65;
4.   printf ("%d %d", i, j);
5. }
```

a. 065 65

b. 53 65

c. 65 65

d. Syntax error

Answer: (b) 53 65

Explanation: This value (065) is an octal value, and it equals to the decimal value 53.

51. Prolog comes under _____
a) Logic Programming
b) Procedural Programming
c) OOP
d) Functional

Answer: a
Explanation: Prolog stands for Programming in Logic. The options mentioned are the four categories of programming. Prolog is a type of logic programming.

52. Java is procedural programming.
a) True
b) False

Answer: b
Explanation: The statement is false. Java is a type of object oriented programming language. It involves solving real-life problems as well.

53. A program that can execute high-level language programs.
a) Compiler
b) Interpreter
c) Sensor
d) Circuitry

Answer: b
Explanation: Interpreter is a program that can execute high-level language programs "directly," without first being translated into machine language.

54. Executables might be called _____
a) native code
b) executable code
c) complex code
d) machine code

Answer: a
Explanation: The executables are sometimes called native code. HLL are translated to Machine language called the native code.

55. Source program is compiled to an intermediate form called _____
a) Byte Code
b) Smart code

c) Executable code
d) Machine code

Answer: a
Explanation: The Source program is compiled to an intermediate form called byte code. For each supported platform, write a "virtual machine" emulator that reads byte code and emulates its execution.

56. _____ is the assembly language for an imaginary architecture.
a) Byte code
b) Machine code
c) Native code
d) Executable code

Answer: a
Explanation: Source program is compiled to an intermediate form – byte code. Byte code is the assembly language for an imaginary architecture.

57. JIT stands for?
a) Just in time
b) Jump in time
c) Jump in text
d) Jump in terms

Answer: a
Explanation: JIT stands for Just in time. JVMs actually compile each bytecode instruction to native code the first time it is used.

58. JVM stands for?
a) Java virtual machine
b) Java visual machine
c) JRE virtual machine
d) JRE visual machine

Answer: a
Explanation: JVM stands for Java Virtual Machine. Other related terms are JRE which is java runtime environment and JDK which is java development kit.

59. A language supported by MS. Net platform.
a) C
b) C++
c) java
d) C#

Answer: d
Explanation: C# is supported by MS. Net platform. JAVA was originally designed for web purposes.

60. Which of the following isn't a characteristic of High level languages?
a) machine code
b) platform independent
c) interactive execution
d) user-friendly

Answer: a
Explanation: HLL isn't in machine language. It is converted to machine language for further processing.

61. Which of the following approach help us understand better about Real time examples, say Vehicle or Employee of an Organisation?
a) Procedural approach
b) Object Oriented approach
c) Both a and b
d) None of the mentioned

Answer: b
Explanation: Object Oriented Programming

supports the concept of class and object which help us understand the real time examples better. Vehicle is defined to be a class. Car, truck, bike are defined to be objects of the class, Vehicle.

62. Which of the following Paradigm is followed by Object Oriented Language Design?
a) Process-Oriented Model
b) Data Controlling access to code.
c) Both a and b
d) None of the mentioned

Answer: b
Explanation: Object-oriented programming organises a program around its data(that is, objects) and a set of well-defined interfaces to that data.

63. Which of the following approach is followed by Object Oriented Language during the execution of a program?
a) Bottom up approach
b) Top down approach
c) Both a and b
d) None of the mentioned

Answer: a
Explanation: Bottom up approach begins with details and moves to higher conceptual level there by reducing complexity and making the concept easier to understand.

64. Which of the following is/are advantage of using object oriented programming?
a) Code Reusability
b) Can create more than one instance of a class without interference
c) Platform independent
d) All of the mentioned

Answer: d
Explanation: None.

65. Java Compiler translates the source code into?
a) Machine code
b) Assembly code
c) Byte code
d) JVM code

Answer: c
Explanation: Java program is converted into 'byte code' which makes it easier to run on wide variety of environments. Only the run-time package JVM has to be implemented for each platform.

66. What is the output of this program?

```
1. class PrintTest {
2.    public static void main(String
   args[])
3.    {
4.       int num = 10;
5.       if (NUM < 100) {
6.          System.out.println("The
   value of num is"+ num);
7.       }
8.    }
9. }
```

a) Compilation error
b) Run time error
c) The value of num is 10
d) None of the mentioned

Answer: a
Explanation: Java is case-sensitive. The variable 'num' used in print statement is not same as the variable 'NUM' used in the if conditional statement.

67. Which of the following is called as 'Comiplation unit'?
a) Object file
b) Byte code
c) Source file

d) All of the mentioned

Answer: c
Explanation: None.

68. What is the output of this program?

```
1. class TypeChecking {
2.    public static void main(String
   args[])
3.    {
4.       int num = 10.5;
5.       System.out.println("Output
   :The value of num is" +num);
6.    }
7. }
```

a) Output: The value of num is 10.5
b) Run-time error
c) Compilation error
d) None of the mentioned

Answer: c
Explanation: Java is storngly typed language. The variable num is declared as 'int' but the value assigned to it is 'float', as a result the code does not get compiled.

69. Which is a named memory location assigned a value by the program?
a) variable
b) literal
c) identifier
d) None of the mentioned

Answer: a
Explanation: None.

70. What is Address Binding?
a) going to an address in memory
b) locating an address with the help of another address
c) binding two addresses together to form a new address in a different memory space
d) a mapping from one address space to another

Answer: d
Explanation: None.

71. Binding of instructions and data to memory addresses can be done at _____
a) Compile time
b) Load time
c) Execution time
d) All of the mentioned

Answer: d
Explanation: None.

72. If the process can be moved during its execution from one memory segment to another, then binding must be _____
a) delayed until run time
b) preponed to compile time
c) preponed to load time
d) none of the mentioned

Answer: a
Explanation: None.

73. What is Dynamic loading?
a) loading multiple routines dynamically
b) loading a routine only when it is called
c) loading multiple routines randomly
d) none of the mentioned

Answer: b
Explanation: None.

74. What is the advantage of dynamic loading?
a) A used routine is used multiple times

b) An unused routine is never loaded
c) CPU utilization increases
d) All of the mentioned

Answer: b
Explanation: None.

75. The idea of overlays is to _____
a) data that are needed at any given time
b) enable a process to be larger than the amount of memory allocated to it
c) keep in memory only those instructions
d) all of the mentioned

Answer: d
Explanation: None.

76. The _____ must design and program the overlay structure.
a) programmer
b) system architect
c) system designer
d) none of the mentioned

Answer: a
Explanation: None.

77. The _____ swaps processes in and out of the memory.
a) Memory manager
b) CPU
c) CPU manager
d) User

Answer: a
Explanation: None.

78. If a higher priority process arrives and wants service, the memory manager can swap out the lower priority process to execute the higher priority process. When the higher priority process finishes, the lower priority process is swapped back in and continues execution. This variant of swapping is sometimes called?
a) priority swapping
b) pull out, push in
c) roll out, roll in
d) none of the mentioned

Answer: c
Explanation: None.

79. If binding is done at assembly or load time, then the process _____ be moved to different locations after being swapped out and in again.
a) can
b) must
c) can never
d) may

Answer: c
Explanation: None.

80. Which of the following is not the primary objectives in the analysis model?
a) describing the customer complaints
b) establishing a basis for the creation of a software design
c) defining a set of requirements that can be validated once the software is built
d) none of the mentioned

Answer: d
Explanation: All the options are covered in analysis model.

81. A description of each function presented in the DFD is contained in a _____

a) data flow
b) process specification
c) control specification
d) data store

Answer: b
Explanation: None.

82. Which diagram indicates the behaviour of the system as a consequence of external events?
a) data flow diagram
b) state transition diagram
c) control specification diagram
d) workflow diagram

Answer: b
Explanation: The state transition diagram represents the various modes of behavior (called states) of the system and the manner in which transitions are made from state to state.

83. A data model contains
a) data object
b) attributes
c) relationships
d) all of the mentioned

Answer: d
Explanation: The data model consists of three interrelated pieces of information: the data object,
the attributes that describe the data object, and the relationships that connect data objects to one another.

84. _____ defines the properties of a data object and take on one of the three different characteristics.
a) data object
b) attributes
c) relationships
d) data object and attributes

Answer: b
Explanation: They can be used to name an instance of the data object, describe the instance, or make reference to another instance in another table.

85. Which of the following is not a data type?
a) Symbolic Data
b) Alphanumeric Data
c) Numeric Data
d) Alphabetic Data

Answer: a
Explanation: Data types are of three basic types: Numeric, Alphabetic and Alphanumeric. Numeric Data consists of only numbers.
Alphabetic Data consists of only letters and a blank character and alphanumeric data consists of symbols.

86. *@Ac# is a type of _____ data.
a) Symbolic
b) Alphanumeric
c) Alphabetic
d) Numeric

Answer: b
Explanation: Alphanumeric data consists of symbols. Alphanumeric data may be a letter, either in uppercase or lowercase or some special symbols like #,^,*,(, etc.

87. Which of the following is not a valid representation in bits?
a) 8-bit
b) 24-bit
c) 32-bit
d) 64-bit

Answer: b
Explanation: There are no criteria like the 24-bit representation of numbers. Numbers can be written in 8-bit, 16-bit, 32-bit and 64-bit as per the IEEE format.

88. What are the entities whose values can be changed called?
a) Constants
b) Variables
c) Modules
d) Tokens

Answer: b
Explanation: Variables are the data entities whose values can be changed. Constants have a fixed value. Tokens are the words which are easily identified by the compiler.

89. Which of the following is not a basic data type in C language?
a) float
b) int
c) real
d) char

Answer: c
Explanation: There are 5 basic data types in C language: int, char, float, double, void. Int is for the representation of integers, char is for strings and characters, float and double are for floating point numbers whereas void is a valueless special data type.

90. BOOLEAN is a type of data type which basically gives a tautology or fallacy.
a) True
b) False

Answer: a
Explanation: A Boolean representation is for giving logical values. It returns either true or false. If a result gives a truth value, it is called tautology whereas if it returns a false term, it is referred to as fallacy.

91. What does FORTRAN stands for?
a) Formula Transfer
b) Formula Transformation
c) Formula Translation
d) Format Transformation

Answer: c
Explanation: FORTRAN is a type of computer language. It was developed for solving mathematical and scientific problems. It is very commonly used among the scientific community.

92. The program written by the programmer in high level language is called _____
a) Object Program
b) Source Program
c) Assembled Program
d) Compiled Program

Answer: b
Explanation: The program written by the programmer is called a source program. The program generated by the compiler after

compilation is called an object program.
The object program is in machine language.

93. A standardized language used for commercial applications.
a) C
b) Java
c) COBOL
d) FORTRAN

Answer: c
Explanation: COBOL is a language used in business and commercial applications. It stands for Common Business Oriented Language. It is imperative, procedural as well as object oriented language.

94. _____ define how the locations can be used.
a) Data types
b) Attributes
c) Links
d) Data Objects

Answer: b
Explanation: Attributes can determine how any location can be used. Attributes can be type, name, component, etc. Data objects are the variables and constants in a program.

95. Which was the first purely object oriented programming language developed?
a) Java
b) C++
c) SmallTalk
d) Kotlin

Answer: c
Explanation: SmallTalk was the first programming language developed which was purely object oriented. It was developed by Alan Kay. OOP concept came into the picture in 1970's.

96. Which of the following best defines a class?
a) Parent of an object
b) Instance of an object
c) Blueprint of an object
d) Scope of an object

Answer: b
Explanation: A class is Blueprint of an object which describes/ shows all the functions and data that are provided by an object of a specific class. It can't be called as parent or instance of an object. Class in general describes all the properties of an object.

97. Who invented OOP?
a) Alan Kay
b) Andrea Ferro
c) Dennis Ritchie
d) Adele Goldberg

Answer: a
Explanation: Alan Kay invented OOP, Andrea Ferro was a part of SmallTalk Development. Dennis invented C++ and Adele Goldberg was in team to develop SmallTalk but Alan actually had got rewarded for OOP.

98. What is the additional feature in classes that was not in structures?
a) Data members
b) Member functions

c) Static data allowed
d) Public access specifier

Answer: b
Explanation: Member functions are allowed inside a class but were not present in structure concept. Data members, static data and public access specifiers were present in structures too.

99. Which is not feature of OOP in general definitions?
a) Code reusability
b) Modularity
c) Duplicate/Redundant data
d) Efficient Code

Answer: c
Explanation: Duplicate/Redundant data is dependent on programmer and hence can't be guaranteed by OOP. Code reusability is done using inheritance. Modularity is supported by using different code files and classes. Codes are more efficient because of features of OOP.

100. Pure OOP can be implemented without using class in a program. (True or False)
a) True
b) False

Answer: b
Explanation: It's false because for a program to be pure OO, everything must be written inside classes. If this rule is violated, the program can't be labelled as purely OO.

101. Which Feature of OOP illustrated the code reusability?
a) Polymorphism
b) Abstraction
c) Encapsulation
d) Inheritance

Answer: d
Explanation: Using inheritance we can reuse the code already written and also can avoid creation of many new functions or variables, as that can be done one time and be reused, using classes.

102. Which language does not support all 4 types of inheritance?
a) C++
b) Java
c) Kotlin
d) Small Talk

Answer: b
Explanation: Java doesn't support all 4 types of inheritance. It doesn't support multiple inheritance. But the multiple inheritance can be implemented using interfaces in Java.

103. How many classes can be defined in a single program?
a) Only 1
b) Only 100
c) Only 999
d) As many as you want

Answer: d
Explanation: Any number of classes can be defined inside a program, provided that their names are different. In java, if public class is present then it must have the same name as that of file.

104. When OOP concept did first came into picture?
a) 1970's
b) 1980's
c) 1993
d) 1995

Answer: a
Explanation: OOP first came into picture in 1970's by Alan and his team. Later it was used by some programming languages and got implemented successfully, SmallTalk was first language to use pure OOP and followed all rules strictly.

105. Why Java is Partially OOP language?
a) It supports usual declaration of primitive data types
b) It doesn't support all types of inheritance
c) It allows code to be written outside classes
d) It does not support pointers

Answer: a
Explanation: As Java supports usual declaration of data variables, it is partial implementation of OOP. Because according to rules of OOP, object constructors must be used, even for declaration of variables.

106. Which concept of OOP is false for C++?
a) Code can be written without using classes
b) Code must contain at least one class
c) A class must have member functions
d) At least one object should be declared in code

Answer: b
Explanation: In C++, it's not necessary to use classes, and hence codes can be written without using OOP concept. Classes may or may not contain member functions, so it's not a necessary condition in C++. And, an object can only be declared in a code if its class is defined/included via header file.

107. Which header file is required in C++ to use OOP?
a) iostream.h
b) stdio.h
c) stdlib.h
d) OOP can be used without using any header file

Answer: d
Explanation: We need not include any specific header file to use OOP concept in C++, only specific functions used in code need their respective header files to be included or classes should be defined if needed.

108. Which of the two features match each other?
a) Inheritance and Encapsulation
b) Encapsulation and Polymorphism
c) Encapsulation and Abstraction
d) Abstraction and Polymorphism

Answer: c
Explanation: Encapsulation and Abstraction are similar features. Encapsulation is actually binding all the properties in a single class or we can say hiding all the features of object inside a class. And Abstraction is hiding unwanted data (for user) and showing only the data required by the user of program.

109. Which feature allows open recursion, among the following?
a) Use of this pointer
b) Use of pointers
c) Use of pass by value
d) Use of parameterized constructor

Answer: a
Explanation: Use of this pointer allows an object to call data and methods of itself whenever needed. This helps us call the members of an object recursively, and differentiate the variables of different scopes.

110. Which of the following is not type of class?
a) Abstract Class
b) Final Class
c) Start Class
d) String Class

Answer: c
Explanation: Only 9 types of classes are provided in general, namely, abstract, final, mutable, wrapper, anonymous, input-output, string, system, network. We may further divide the classes into parent class and subclass if inheritance is used.

111. Class is pass by _____
a) Value
b) Reference
c) Value or Reference, depending on program
d) Copy

Answer: b
Explanation: Classes are pass by reference, and the structures are pass by copy. It doesn't depend on the program.

112. What is default access specifier for data members or member functions declared within a class without any specifier, in C++?
a) Private
b) Protected
c) Public
d) Depends on compiler

Answer: a
Explanation: The data members and member functions are Private by default in C++ classes, if none of the access specifier is used. It is actually made to increase the privacy of data.

113. Which is most appropriate comment on following class definition?

```
class Student
{
   int a;
   public : float a;
};
```

a) Error : same variable name can't be used twice
b) Error : Public must come first
c) Error : data types are different for same variable
d) It is correct

Answer: a
Explanation: Same variable can't be defined twice in same scope. Even if the data types are different, variable name must be different. There is no rule like Public member should come first or last.

114. Which is known as a generic class?
a) Abstract class
b) Final class
c) Template class

d) Efficient Code

Answer: c
Explanation: Template classes are known to be generic classes because those can be used for any data type value and the same class can be used for all the variables of different data types.

115. Size of a class is _____
a) Sum of the size of all the variables declared inside the class
b) Sum of the size of all the variables along with inherited variables in the class
c) Size of the largest size of variable
d) Classes doesn't have any size

Answer: d
Explanation: Classes doesn't have any size, actually the size of object of the class can be defined. That is done only when an object is created and its constructor is called.

116. Which class can have member functions without their implementation?
a) Default class
b) String class
c) Template class
d) Abstract class

Answer: d
Explanation: Abstract classes can have member functions with no implementation, where the inheriting subclasses must implement those functions.

117. Which of the following describes a friend class?
a) Friend class can access all the private members of the class, of which it is a friend
b) Friend class can only access protected members of the class, of which it is a friend
c) Friend class don't have any implementation
d) Friend class can't access any data member of another class but can use it's methods

Answer: a
Explanation: A friend class can access all the private members of another class, of which it is a friend. It is a special class provided to use when you need to reuse the data of a class but don't want that class to have those special functions.

118. What is the scope of a class nested inside another class?
a) Protected scope
b) Private scope
c) Global scope
d) Depends on access specifier and inheritance used

Answer: d
Explanation: It depends on the access specifier and the type of inheritance used with the class, because if the class is inherited then the nested class can be used by subclass too, provided it's not of private type.

119. Class with main() function can be inherited.
a) True
b) False

Answer: a
Explanation: The class containing main function can be inherited and hence the program can be executed using the derived class names also in java.

120. Which definition best describes an object?
a) Instance of a class
b) Instance of itself
c) Child of a class
d) Overview of a class

Answer: a
Explanation: An object is instance of its class. It can be declared in the same way that a variable is declared, only thing is you have to use class name as the data type.

121. How many objects can be declared of a specific class in a single program?
a) 32768
b) 127
c) 1
d) As many as you want

Answer: d
Explanation: You can create as many objects of a specific class as you want, provided enough memory is available.

122. Which among the following is false?
a) Object must be created before using members of a class
b) Memory for an object is allocated only after its constructor is called
c) Objects can't be passed by reference
d) Objects size depends on its class data members

Answer: c
Explanation: Objects can be passed by reference. Objects can be passed by value also. If the object of a class is not created, we can't use members of that class.

123. Which of the following is incorrect?
a) class student{ }s;
b) class student{ }; student s;
c) class student{ }s[];
d) class student{ }; student s[5];

Answer: c
Explanation: The array must be specified with a size. You can't declare object array, or any other linear array without specifying its size. It's a mandatory field.

124. The object can't be _____
a) Passed by reference
b) Passed by value
c) Passed by copy
d) Passed as function

Answer: d
Explanation: Object can't be passed as function as it is an instance of some class, it's not a function. Object can be passed by reference, value or copy. There is no term defined as pass as function for objects.

125. What is size of the object of following class (64 bit system)?

```
class student {  int rollno;  char  name[20];
static int studentno;  };
```

a) 20
b) 22
c) 24
d) 28

Answer: c
Explanation: The size of any object of student class will be of size 4+20=24, because static members are not really considered as property of a single object. So static variables size will not be added.

126. Functions can't return objects.
a) True
b) False

Answer: b
Explanation: Functions can always return an object if the return type is same as that of object being returned. Care has to be taken while writing the prototype of the function.

127. How members of an object are accessed?
a) Using dot operator/period symbol
b) Using scope resolution operator

c) Using member names directly
d) Using pointer only

Answer: a
Explanation: Using dot operator after the name of object we can access its members. It is not necessary to use the pointers. We can't use the names directly because it may be used outside the class.

128. If a local class is defined in a function, which of the following is true for an object of that class?
a) Object is accessible outside the function
b) Object can be declared inside any other function
c) Object can be used to call other class members
d) Object can be used/accessed/declared locally in that function

Answer: d
Explanation: For an object which belongs to a local class, it is mandatory to declare and use the object within the function because the class is accessible locally within the class only.

129. Which among the following is wrong?
a) class student{ }; student s;
b) abstract class student{ }; student s;
c) abstract class student{ }s[50000000];
d) abstract class student{ }; class toppers: public student{ }; topper t;

Answer: b
Explanation: We can never create instance of an abstract class. Abstract classes doesn't have constructors and hence when an instance is created there is no facility to initialize its members. Option d is correct because topper class is inheriting the base abstract class student, and hence topper class object can be created easily.

130. Which feature of OOP indicates code reusability?
a) Encapsulation
b) Inheritance
c) Abstraction
d) Polymorphism

Answer: b
Explanation: Inheritance indicates the code reusability. Encapsulation and abstraction are meant to hide/group data into one element. Polymorphism is to indicate different tasks performed by a single entity.

131. If a function can perform more than 1 type of tasks, where the function name remains same, which feature of OOP is used here?
a) Encapsulation
b) Inheritance
c) Polymorphism
d) Abstraction

Answer: c
Explanation: For the feature given above, the OOP feature used is Polymorphism. Example of polymorphism in real life is a kid, who can be a student, a son, a brother depending on where he is.

132. If different properties and functions of a real world entity is grouped or embedded into a single element, what is it called in OOP language?
a) Inheritance
b) Polymorphism
c) Abstraction
d) Encapsulation

Answer: d
Explanation: It is Encapsulation, which groups different properties and functions of a real world entity into single element. Abstraction, on other hand, is hiding of functional or exact working of codes and showing only the things which are required by the user.

133. Which of the following is not a feature of pure OOP?
a) Classes must be used
b) Inheritance
c) Data may/may not be declared using object
d) Functions Overloading

Answer: c
Explanation: Data must be declared using objects. Object usage is mandatory because it in turn calls its constructors, which in turn must have a class defined. If object is not used, it is a violation of pure OOP concept.

134. Which among the following doesn't come under OOP concept?
a) Platform independent
b) Data binding
c) Message passing
d) Data hiding

Answer: a
Explanation: Platform independence is not feature of OOP. C++ supports OOP but it's not a platform independent language. Platform independence depends on programming language.

135. Which feature of OOP is indicated by the following code?

```
class student{  int marks;  };

class topper:public student{  int age;
topper(int age){ this.age=age; } };
```

a) Inheritance
b) Polymorphism
c) Inheritance and polymorphism
d) Encapsulation and Inheritance

Answer: d
Explanation: Encapsulation is indicated by use of classes. Inheritance is shown by inheriting the student class into topper class. Polymorphism is not shown here because we have defined the constructor in the topper class but that doesn't mean that default constructor is overloaded.

136. Which feature may be violated if we don't use classes in a program?
a) Inheritance can't be implemented
b) Object must be used is violated
c) Encapsulation only is violated
d) Basically all the features of OOP gets violated

Answer: d
Explanation: All the features are violated because Inheritance and Encapsulation won't be implemented. Polymorphism and Abstraction are still possible in some cases, but the main features like data binding, object use and etc won't be used hence the use of class is must for OOP concept.

137. How many basic features of OOP are required for a programming language to be purely OOP?
a) 7

b) 6
c) 5
d) 4

Answer: a
Explanation: There are 7 basic features that define whether a programing language is pure OOP or not. The 4 basic features are inheritance, polymorphism, encapsulation and abstraction. Further, one is, object use is must, secondly, message passing and lastly, Dynamic binding.

138. The feature by which one object can interact with another object is _____
a) Data transfer
b) Data Binding
c) Message Passing
d) Message reading

Answer: c
Explanation: The interaction between two object is called the message passing feature. Data transfer is not a feature of OOP. Also, message reading is not a feature of OOP.

139. _____ underlines the feature of Polymorphism in a class.
a) Nested class
b) Enclosing class
c) Inline function
d) Virtual Function

Answer: d
Explanation: Virtual Functions can be defined in any class using the keyword virtual. All the classes which inherit the class containing the virtual function, define the virtual function as required. Redefining the function on all the derived classes according to class and use represents polymorphism.

140. Which feature in OOP is used to allocate additional function to a predefined operator in any language?
a) Operator Overloading
b) Function Overloading
c) Operator Overriding
d) Function Overriding

Answer: a
Explanation: The feature is operator overloading. There is not a feature named operator overriding specifically. Function overloading and overriding doesn't give addition function to any operator.

141. Which among doesn't illustrates polymorphism?
a) Function overloading
b) Function overriding
c) Operator overloading
d) Virtual function

Answer: b
Explanation: Function overriding doesn't illustrate polymorphism because the functions are actually different and theirs scopes are different. Function and operator overloading illustrate proper polymorphism. Virtual functions show polymorphism because all the classes which inherit virtual function, define the same function in different ways.

142. Exception handling is a feature of OOP.
a) True
b) False

Answer: a
Explanation: Exception handling is a feature of OOP as it includes classes concept in most of the cases. Also it may come handy while using inheritance.

143. Which among the following, for a pure OOP language, is true?
a) The language should follow 3 or more features of OOP
b) The language should follow at least 1 feature of OOP
c) The language must follow only 3 features of OOP
d) The language must follow all the rules of OOP

Answer: d
Explanation: The language must follow all the rules of OOP to be called a purely OOP language. Even if a single OOP feature is not followed, then it's known to be a partially OOP language.

144. Does OOP provide better security than POP?
a) Always true for any programming language
b) May not be true with respect to all programming languages
c) It depends on type of program
d) It's vice-versa is true

Answer: a
Explanation: It is always true as we have the facility of private and protected access specifiers. Also, only the public and global data are available globally or else the program should have proper permission to access the private data.

145. Which among the following best describes polymorphism?
a) It is the ability for a message/data to be processed in more than one form
b) It is the ability for a message/data to be processed in only 1 form
c) It is the ability for many messages/data to be processed in one way
d) It is the ability for undefined message/data to be processed in at least one way

Answer: a
Explanation: It is actually the ability for a message / data to be processed in more than one form. The word polymorphism indicates many-forms. So if a single entity takes more than one form, it is known as polymorphism.

146. What do you call the languages that support classes but not polymorphism?
a) Class based language
b) Procedure Oriented language
c) Object-based language
d) If classes are supported, polymorphism will always be supported

Answer: c
Explanation: The languages which support classes but doesn't support polymorphism, are known as object-based languages. Polymorphism is such an important feature, that is a language doesn't support this

feature, it can't be called as a OOP language.

147. Which among the following is the language which supports classes but not polymorphism?
a) SmallTalk
b) Java
c) C++
d) Ada

Answer: d
Explanation: Ada is the language which supports the concept of classes but doesn't support the polymorphism feature. It is an object-based programming language. Note that it's not an OOP language.

148. If same message is passed to objects of several different classes and all of those can respond in a different way, what is this feature called?
a) Inheritance
b) Overloading
c) Polymorphism
d) Overriding

Answer: c
Explanation: The feature defined in question defines polymorphism features. Here the different objects are capable of responding to the same message in different ways, hence polymorphism.

149. Which type of function among the following shows polymorphism?
a) Inline function
b) Virtual function
c) Undefined functions
d) Class member functions

Answer: b
Explanation: Only virtual functions among these can show polymorphism. Class member functions can show polymorphism too but we should be sure that the same function is being overloaded or is a function of abstract class or something like this, since we are not sure about all these, we can't say whether it can show polymorphism or not.

150. In case of using abstract class or function overloading, which function is supposed to be called first?
a) Local function
b) Function with highest priority in compiler
c) Global function
d) Function with lowest priority because it might have been halted since long time, because of low priority

Answer: b
Explanation: Function with highest priority is called. Here, it's not about the thread scheduling in CPU, but it focuses on whether the function in local scope is present or not, or if scope resolution is used in some way, or if the function matches the argument signature. So all these things define which function has the highest priority to be called in runtime. Local function could be one of the answer but we can't say if someone have used pointer to another function or same function name.

151. Which among the following can't be used for polymorphism?
a) Static member functions
b) Member functions overloading

c) Predefined operator overloading
d) Constructor overloading

Answer: a
Explanation: Static member functions are not property of any object. Hence it can't be considered for overloading/overriding. For polymorphism, function must be property of object, not only of class.

152. Which among the following can show polymorphism?
a) Overloading ||
b) Overloading +=
c) Overloading <<
d) Overloading &&

Answer: c
Explanation: Only insertion operator can be overloaded among all the given options. And the polymorphism can be illustrated here only if any of these is applicable of being overloaded. Overloading is type of polymorphism.

153. Polymorphism is possible in C language.
a) True
b) False

Answer: a
Explanation: It is possible to implement polymorphism in C language, even though it doesn't support class. We can use structures and then declare pointers which in turn points to some function. In this way we simulate the functions like member functions but not exactly member function. Now we can overload these functions, hence implementing polymorphism in C language.

154. Which problem may arise if we use abstract class functions for polymorphism?
a) All classes are converted as abstract class
b) Derived class must be of abstract type
c) All the derived classes must implement the undefined functions
d) Derived classes can't redefine the function

Answer: c
Explanation: The undefined functions must be defined is a problem, because one may need to implement few undefined functions from abstract class, but he will have to define each of the functions declared in abstract class. Being useless task, it is a problem sometimes.

155. Which among the following is not true for polymorphism?
a) It is feature of OOP
b) Ease in readability of program
c) Helps in redefining the same functionality
d) Increases overhead of function definition always

Answer: d
Explanation: It never increases function definition overhead, one way or another if you don't use polymorphism, you will use the definition in some other way, so it actually helps to write efficient codes.

156. If 2 classes derive one base class and redefine a function of base class, also overload some operators inside class body. Among these two things of function and

operator overloading, where is polymorphism used?
a) Function overloading only
b) Operator overloading only
c) Both of these are using polymorphism
d) Either function overloading or operator overloading because polymorphism can be applied only once in a program

Answer: d
Explanation: Both of them are using polymorphism. It is not necessary that polymorphism can be used only once in a program, it can be used anywhere, any number of times in a single program.

157. Which among the following best describes encapsulation?
a) It is a way of combining various data members into a single unit
b) It is a way of combining various member functions into a single unit
c) It is a way of combining various data members and member functions into a single unit which can operate on any data
d) It is a way of combining various data members and member functions that operate on those data members into a single unit

Answer: d
Explanation: It is a way of combining both data members and member functions, which operate on those data members, into a single unit. We call it a class in OOP generally. This feature have helped us modify the structures used in C language to be upgraded into class in C++ and other languages.

158. If data members are private, what can we do to access them from the class object?
a) Create public member functions to access those data members
b) Create private member functions to access those data members
c) Create protected member functions to access those data members
d) Private data members can never be accessed from outside the class

Answer: a
Explanation: We can define public member functions to access those private data members and get their value for use or alteration. They can't be accessed directly but is possible to be access using member functions. This is done to ensure that the private data doesn't get modified accidentally.

159. While using encapsulation, which among the following is possible?
a) Code modification can be additional overhead
b) Data member's data type can be changed without changing any other code
c) Data member's type can't be changed, or whole code have to be changed
d) Member functions can be used to change the data type of data members

Answer: b
Explanation: Data member's data type can be changed without changing any further code. All the members using that data can continue in the same way without any modification. Member functions can never change the data type of same class data members.

160. Find which of the following uses encapsulation?
a) void main(){ int a; void fun(int a=10; cout<<a); fun(); }
b) class student{ int a; public: int b;};

c) class student{int a; public: void disp(){ cout<<a;} };
d) struct topper{ char name[10]; public : int marks; }

Answer: c
Explanation: It is the class which uses both the data members and member functions being declared inside a single unit. Only data members can be there in structures also. And the encapsulation can only be illustrated if some data/operations are associated within class.

161. Encapsulation helps in writing _____ classes in java.
a) Mutable
b) Abstract
c) Wrapper
d) Immutable

Answer: d
Explanation: Immutable classes are used for caching purpose generally. And it can be created by making the class as final and making all its members private.

162. Which among the following should be encapsulated?
a) The data which is prone to change is near future
b) The data prone to change in long terms
c) The data which is intended to be changed
d) The data which belongs to some other class

Answer: a
Explanation: The data prone to change in near future is usually encapsulated so that it doesn't get changed accidentally. We encapsulate the data to hide the critical working of program from outside world.

163. How can Encapsulation be achieved?
a) Using Access Specifiers
b) Using only private members
c) Using inheritance
d) Using Abstraction

Answer: a
Explanation: Using access specifiers we can achieve encapsulation. Using this we can in turn implement data abstraction. It's not necessary that we only use private access.

164. Which among the following violates the principle of encapsulation almost always?
a) Local variables
b) Global variables
c) Public variables
d) Array variables

Answer: b
Explanation: Global variables almost always violates the principles of encapsulation. Encapsulation says the data should be accessed only by required set of elements. But global variable is accessible everywhere, also it is most prone to changes. It doesn't hide the internal working of program.

165. Which among the following would destroy the encapsulation mechanism if it was allowed in programming?
a) Using access declaration for private members of base class
b) Using access declaration for public members of base class
c) Using access declaration for local variable of main() function
d) Using access declaration for global variables

Answer: a
Explanation: If using access declaration for private members of base class was allowed in programming, it would have destroyed

whole concept of encapsulation. As if it was possible, any class which gets inherited privately, would have been able to inherit the private members of base class, and hence could access each and every member of base class.

166. Which among the following can be a concept against encapsulation rules?
a) Using function pointers
b) Using char* string pointer to be passed to non-member function
c) Using object array
d) Using any kind of pointer/array address in passing to another function

Answer: d
Explanation: If we use any kind of array or pointer as data member which should not be changed, but in some case its address is passed to some other function or similar variable. There are chances to modify its whole data easily. Hence Against encapsulation.

167. Consider the following code and select the correct option.

```
class student
{
    int marks;
    public : int* fun()
    {
        return &marks;
    }
};
main()
{
   student s;
   int *ptr=c.fun();
   return 0;
}
```

a) This code is good to go
b) This code may result in undesirable conditions
c) This code will generate error
d) This code violates encapsulation

Answer: d
Explanation: This code violates the encapsulation. By this code we can get the address of the private member of the class, hence we can change the value of private member, which is against the rules.

168. Encapsulation is the way to add functions in a user defined structure.
a) True
b) False

Answer: b
Explanation: False, because we can't call these structures if member functions are involved, it must be called class. Also, it is not just about adding functions, it's about binding data and functions together.

169. Using encapsulation data security is _____
a) Not ensured
b) Ensured to some extent
c) Purely ensured
d) Very low

Answer: b
Explanation: The encapsulation can only ensure data security to some extent. If pointer and addresses are misused, it may violate encapsulation. Use of global variables also makes the program vulnerable, hence we can't say that encapsulation gives pure security.

170. Which among the following best defines abstraction?
a) Hiding the implementation

b) Showing the important data
c) Hiding the important data
d) Hiding the implementation and showing only the features

Answer: d
Explanation: It includes hiding the implementation part and showing only the required data and features to the user. It is done to hide the implementation complexity and details from the user. And to provide a good interface in programming.

172. Hiding the implementation complexity can _____
a) Make the programming easy
b) Make the programming complex
c) Provide more number of features
d) Provide better features

Answer: a
Explanation: It can make programming easy. The programming need not know how the inbuilt functions are working but can use those complex functions directly in the program. It doesn't provide more number of features or better features.

173. Class is _____ abstraction.
a) Object
b) Logical
c) Real
d) Hypothetical

Answer: b
Explanation: Class is logical abstraction because it provides a logical structure for all of its objects. It gives an overview of the features of an object.

174. Object is _____ abstraction.
a) Object
b) Logical
c) Real
d) Hypothetical

Answer: c
Explanation: Object is real abstraction because it actually contains those features of class. It is the implementation of overview given by class. Hence the class is logical abstraction and its object is real.

175. Abstraction gives higher degree of _____
a) Class usage
b) Program complexity
c) Idealized interface
d) Unstable interface

Answer: c
Explanation: It is to idealize the interface. In this way the programmer can use the programming features more efficiently and can code better. It can't increase the program complexity, as the feature itself is made to hide it.

176. Abstraction can apply to _____
a) Control and data
b) Only data
c) Only control
d) Classes

Answer: a
Explanation: Abstraction applies to both. Control abstraction involves use of subroutines and control flow abstraction.

Data abstraction involves handling pieces of data in meaningful ways.

177. Which among the following can be viewed as combination of abstraction of data and code.
a) Class
b) Object
c) Inheritance
d) Interfaces

Answer: b
Explanation: Object can be viewed as abstraction of data and code. It uses data members and their functioning as data abstraction. Code abstraction as use of object of inbuilt class.

178. Abstraction principle includes_____
a) Use abstraction at its minimum
b) Use abstraction to avoid longer codes
c) Use abstraction whenever possible to avoid duplication
d) Use abstraction whenever possible to achieve OOP

Answer: c
Explanation: Abstraction principle includes use of abstraction to avoid duplication (usually of code). It this way the program doesn't contain any redundant functions and make the program efficient.

179. Higher the level of abstraction, higher are the details.
a) True
b) False

Answer: b
Explanation: Higher the level of abstraction, lower are the details. The best way to understand this is to consider a whole system that is highest level of abstraction as it hides everything inside. And next lower level would contain few of the computer components and so on.

180. Encapsulation and abstraction differ as _____
a) Binding and Hiding respectively
b) Hiding and Binding respectively
c) Can be used any way
d) Hiding and hiding respectively

Answer: a
Explanation: Abstraction is hiding the complex code. For example, we directly use cout object in C++ but we don't know how is it actually implemented. Encapsulation is data binding, as in, we try to combine a similar type of data and functions together.

182. If two classes combine some private data members and provides public member functions to access and manipulate those data members. Where is abstraction used?
a) Using private access specifier for data members
b) Using class concept with both data members and member functions
c) Using public member functions to access and manipulate the data members
d) Data is not sufficient to decide what is being used

Answer: c
Explanation: It is the concept of hiding program complexity and actual working in

background. Hence use of public member functions illustrates abstraction here.

183. A phone is made up of many components like motherboard, camera, sensors and etc. If the processor represents all the functioning of phone, display shows the display only, and the phone is represented as a whole. Which among the following have highest level of abstraction?
a) Motherboard
b) Display
c) Camera
d) Phone

Answer: d
Explanation: Phone as a whole have the highest level of abstraction. This is because the phone being a single unit represents the whole system. Whereas motherboard, display and camera are its components.

184. Which among the following is not a level of abstraction?
a) Logical level
b) Physical level
c) View level
d) External level

Answer: d
Explanation: Abstraction is generally divided into 3 different levels, namely, logical, physical and view level. External level is not defined in terms of abstraction.

185. Using higher degree of abstraction _____
a) May get unsafe
b) May reduce readability
c) Can be safer
d) Can increase vulnerability

Answer: c
Explanation: It will make the code safer. One may think it reduces the readability, but the fact is, it actually helps us understand the code better. We don't have to read the complex code which is of no use in understanding the program.

186. Which among the following best describes the Inheritance?
a) Copying the code already written
b) Using the code already written once
c) Using already defined functions in programming language
d) Using the data and functions into derived segment

Answer: d
Explanation: It can only be indicated by using the data and functions that we use in derived class, being provided by parent class. Copying code is nowhere similar to this concept, also using the code already written is same as copying. Using already defined functions is not inheritance as we are not adding any of our own features.

187. How many basic types of inheritance are provided as OOP feature?
a) 4
b) 3
c) 2
d) 1

Answer: a
Explanation: There are basically 4 types of inheritance provided in OOP, namely, single

level, multilevel, multiple and hierarchical inheritance. We can add one more type as Hybrid inheritance but that is actually the combination any types of inheritance from the 4 basic ones.

188. Which among the following best defines single level inheritance?
a) A class inheriting a derived class
b) A class inheriting a base class
c) A class inheriting a nested class
d) A class which gets inherited by 2 classes

Answer: b
Explanation: A class inheriting a base class defines single level inheritance. Inheriting an already derived class makes it multilevel inheritance. And if base class is inherited by 2 other classes, it is multiple inheritance.

189. Which among the following is correct for multiple inheritance?
a) class student{public: int marks;}s; class stream{int total;}; class topper:public student, public stream{ };
b) class student{int marks;}; class stream{ }; class topper: public student{ };
c) class student{int marks;}; class stream:public student{ };
d) class student{ }; class stream{ }; class topper{ };

Answer: a
Explanation: Class topper is getting derived from 2 other classes and hence it is multiple inheritance. Topper inherits class stream and class student publicly and hence can use its features. If only few classes are defined, there we are not even using inheritance (as in option class student{ }; class stream{ }; class topper{ };).

190. Which programming language doesn't support multiple inheritance?
a) C++ and Java
b) C and C++
c) Java and SmallTalk
d) Java

Answer: d
Explanation: Java doesn't support multiple inheritance. But that feature can be implemented by using the interfaces concept. Multiple inheritance is not supported because of diamond problem and similar issues.

191. Which among the following is correct for a hierarchical inheritance?
a) Two base classes can be used to be derived into one single class
b) Two or more classes can be derived into one class
c) One base class can be derived into other two derived classes or more
d) One base class can be derived into only 2 classes

Answer: c
Explanation: One base class can be derived into the other two derived classes or more. If only one class gets derived by only 2 other classes, it is also hierarchical inheritance, but it is not a mandatory condition, because any number of derived classes can be there.

192. Which is the correct syntax of inheritance?

a) class derived_classname : base_classname{ /*define class body*/ };
b) class base_classname : derived_classname{ /*define class body*/ };
c) class derived_classname : access base_classname{ /*define class body*/ };
d) class base_classname :access derived_classname{ /*define class body*/ };

Answer: c
Explanation: Firstly, keyword class should come, followed by the derived class name. Colon is must followed by access in which base class has to be derived, followed by the base class name. And finally the body of class. Semicolon after the body is also must.

193. Which type of inheritance leads to diamond problem?
a) Single level
b) Multi-level
c) Multiple
d) Hierarchical

Answer: c
Explanation: When 2 or more classes inherit the same class using multiple inheritance and then one more class inherits those two base classes, we get a diamond like structure. Here, ambiguity arises when same function gets derived into 2 base classes and finally to 3rd level class because same name functions are being inherited.

194. Which access type data gets derived as private member in derived class?
a) Private
b) Public
c) Protected
d) Protected and Private

Answer: a
Explanation: It is a rule, that when a derived class inherits the base class in private access mode, all the members of base class gets derived as private members of the derived class.

195. If a base class is inherited in protected access mode then which among the following is true?
a) Public and Protected members of base class becomes protected members of derived class
b) Only protected members become protected members of derived class
c) Private, Protected and Public all members of base, become private of derived class
d) Only private members of base, become private of derived class

Answer: a
Explanation: As the programming language rules apply, all the public and protected members of base class becomes protected members of derived class in protected access mode. It can't be changed because it would hinder the security of data and may add vulnerability in the program.

196. How many types of inheritance are possible in C++?
a) 2
b) 3
c) 4
d) 5

Answer: d
Explanation: There are five types of inheritance that are possible in C++. Single level, Multilevel, multiple, hierarchical and hybrid. Here we count hybrid also because

it sometimes can bring up a new form of inheritance, Like inheritance using multiple and hierarchical, which sometimes results in diamond problem.

197. Which among the following is true?
a) Java supports all types of inheritance
b) Java supports multiple inheritance
c) Java doesn't support multiple inheritance
d) Java doesn't support inheritance

Answer: c
Explanation: Java doesn't support multiple inheritance. This is done to avoid the diamond problem that sometimes arises with inherited functions. Though, multiple inheritance can be implemented in java using interfaces.

198. Which type of inheritance is illustrated by the following code?

```
class student{ public: int marks; };
class topper: public student { public: char grade; };
class average{ public: int makrs_needed; };
class section: public average{ public: char name[10];  };
class overall: public average{  public: int students;  };
```

a) Single level
b) Multilevel and single level
c) Hierarchical
d) Hierarchical and single level

Answer: c
Explanation: It is hierarchical inheritance and single level inheritance. Since class topper is inheriting class student, it is single level inheritance. And then average is inherited by section and overall, so it is hierarchical inheritance. But both of them are separate. Hence it is not hybrid inheritance.

199. Which among the following best describes multiple inheritance?
a) Two classes being parent of any other classes
b) Three classes being parent of other classes
c) More than one class being parent of other child classes
d) More than one class being parent of single child

Answer: d
Explanation: If a class inherits more than one class, it is known as multiple inheritance. This should not be referred with only two or three classes being inherited. But there must be one class which inherits more than one class to be called as multiple inheritance.

200. How many types of inheritance can be used at a time in a single program?
a) Any two types
b) Any three types
c) Any 4 types
d) Any type, any number of times

Answer: d
Explanation: Any type of inheritance can be used in any program. There is no rule to use only few types of inheritance. Only thing that matters is how the classes are inherited and used.

201. Which among the following defines single level inheritance?
a) One base class derives another class
b) One derived class inherits from one base class
c) One base class inherits from one derived class
d) One derived class derives from another

derived class

Answer: b
Explanation: If only one base class is used to derive only one subclass, it is known as single level inheritance. The reason of this name is that we inherit the base class to one more level and stop the inheritance any further.

202. If class A and class B are derived from class C and class D, then _____
a) Those are 2 pairs of single inheritance
b) That is multilevel inheritance
c) Those is enclosing class
d) Those are all independent classes

Answer: a
Explanation: Since class A is derived from class C and then class B is derived from class D, there are two pairs of classes which shows single inheritance. Those two pairs are independent of each other though.

203. If single inheritance is used, program will contain _____
a) At least 2 classes
b) At most 2 classes
c) Exactly 2 classes
d) At most 4 classes

Answer: a
Explanation: The program will contain at least 2 classes in the sense of base and derived classes. At least one base class and one derived class must be there. Types of inheritance remains the same though.

204. Single level inheritance supports _____ inheritance.
a) Runtime
b) Compile time
c) Multiple inheritance
d) Language independency

Answer: a
Explanation: The runtime inheritance is done when object of a class is created to call a method. At runtime the function is searched if it is in class of object. If not, it will search in its parent classes and hierarchy for that method.

205. Which method in the code below is single level inherited?

```
class A
{
        protected int a, b;
        public: void show()
        {
                cout&lt;&lt;a&lt;&lt;b;
        }
};
class B: public A
{
        public: void disp()
        {

        cout&lt;&lt;a++&lt;&lt;b++;
        }
};
class C: private A, public B
{
        void avg()
        {
                cout&lt;&lt;(a+b)/2;
        }
};
```

a) Class A
b) Class B
c) Class C
d) None

Answer: b
Explanation: Class B is using single level inheritance. Class C is using multiple

inheritance. And class A is parent of other two classes.

206. If single level inheritance is used and an abstract class is created with some undefined functions, can its derived class also skip some definitions?
a) Yes, always possible
b) Yes, possible if only one undefined function
c) No, at least 2 undefined functions must be there
d) No, the derived class must implement those methods

Answer: d
Explanation: The derived class must implement those methods. This is because the parent class is abstract and hence will have some undefined functions which has to be defined in derived classes. Since we are using single level inheritance, if derived class doesn't implement those functions then one more class has to be there which will become multi-level inheritance.

207. Which among the following is false for single level inheritance?
a) There can be more than 2 classes in program to implement single inheritance
b) There can be exactly 2 classes to implement single inheritance in a program
c) There can be more than 2 independent classes involved in single inheritance
d) The derived class must implement all the abstract method if single inheritance is used

Answer: c
Explanation: If more than 2 independent classes are involved to implement the single level inheritance, it won't be possible as there must be only one child and one parent class and none other related class.

208. Which concept will result in derived class with more features (consider maximum 3 classes)?
a) Single inheritance
b) Multiple inheritance
c) Multilevel inheritance
d) Hierarchical inheritance

Answer: b
Explanation: If single inheritance is used then only feature of a single class are inherited, and if multilevel inheritance is used, the 2nd class might have use private inheritance. Hence only multiple inheritance can result in derived class with more features. This is not mandatory but in a case if we consider same number of features in each class, it will result the same.

209. Single level inheritance is safer than _____
a) Multiple inheritance
b) Interfaces
c) Implementations
d) Extensions

Answer: a
Explanation: Interfaces also represent a way of inheritance but is a wide word to decide which inheritance we are talking about in it, hence can't be considered. Implementation and extensions also doesn't match that level of specific idea. And multiple inheritance not so safe as it might result in some ambiguity.

210. Which language doesn't support single level inheritance?
a) Java
b) C++
c) Kotlin
d) All languages support it

Answer: d
Explanation: All the languages support single level inheritance. Since any class can inherit other classes as required, if single level inheritance was not allowed it would result in failing a lot of features of OOP.

211. Which among the following best defines multilevel inheritance?
a) A class derived from another derived class
b) Classes being derived from other derived classes
c) Continuing single level inheritance
d) Class which have more than one parent

Answer: b
Explanation: Only if the class is being derived from other derived class, it can be called as multilevel inheritance. If a class is derived from another class, it is single level inheritance. There must be more than one level of inheritance.

212. If there are 5 classes, E is derived from D, D from C, C from B and B from A. Which class constructor will be called first if the object of E or D is created?
a) A
b) B
c) C
d) A and B

Answer: a
Explanation: A is parent of all other classes indirectly. Since A is parent of B and B is parent of C and so on till E. Class A constructor will be called first always.

213. If there are 3 classes. Class C is derived from class B and B is derived from A, Which class destructor will be called at last if object of C is destroyed.
a) A
b) B
c) C
d) All together

Answer: a
Explanation: The destructors are called in the reverse order of the constructors being called. Hence in multilevel inheritance, the constructors are created from parent to child, which leads to destruction from child to parent. Hence class A destructor will be called at last.

214. Which Class is having highest degree of abstraction in multilevel inheritance of 5 levels?
a) Class at 1st level
b) Class 2nd last level
c) Class at 5th level
d) All with same abstraction

Answer: a
Explanation: The class with highest degree of abstraction will be the class at the 1st level. You can look at a simple example like, a CAR is more abstract than SPORTS CAR class. The level of abstraction decrease with each level as more details comes out.

215. If all the classes use private inheritance in multilevel inheritance then
_____
a) It will not be called multilevel inheritance
b) Each class can access only non-private members of its parent
c) Each subsequent class can access all members of previous level parent classes

d) None of the members will be available to any other class

Answer: b
Explanation: The classes will be able to access only the non-private members of its parent class. The classes are using private inheritance, hence all the members of the parent class become private in the derived class. In turn those won't be allowed for further inheritance or direct access outside the class.

216. Multilevel inheritance allows _____ in the program.
a) Only 7 levels of inheritance
b) At least 7 levels of inheritance
c) At most 16 levels of inheritance
d) As many levels of inheritance as required

Answer: d
Explanation: The multilevel inheritance allows any number of levels of inheritance. This is the maximum flexibility feature to make the members available to all the new classes and to add their own functionalities. The code reusability is used too.

217. Multiple inheritance is _____
a) When a class is derived from another class
b) When a class is derived from two or more classes
c) When a class is derived from other two derived classes
d) When a class is derived from exactly one class

Answer: b
Explanation: The multiple inheritance is used when a class is being derived using two base classes or more. This way a single class can have features of more than one classes inherited into a single unit. This lets us combine two class members into a single class.

218. Which problem arises due to multiple inheritance, if hierarchical inheritance is used previously for its base classes?
a) Diamond
b) Circle
c) Triangle
d) Loop

Answer: a
Explanation: The diamond problem arises when multiple inheritance is used. This problem arises because the same name member functions get derived into a single class. Which in turn creates ambiguity in calling those methods.

219. Which among the following is best to define hierarchical inheritance?
a) More than one classes being derived from one class
b) More than 2 classes being derived from single base class
c) At most 2 classes being derived from single base class
d) At most 1 class derived from another class

Answer: a
Explanation: When two or more classes get derived from a single base class, it is known as hierarchical inheritance. This gives us

freedom to use same code with different scopes and flexibility into different classes.

220. Do members of base class gets divided among all of its child classes?
a) Yes, equally
b) Yes, depending on type of inheritance
c) No, it's doesn't get divided
d) No, it may or may not get divided

Answer: c
Explanation: The class members doesn't get divided among the child classes. All the members get derived to each of the subclasses as whole. The only restriction is from the access specifiers used.

221. Which among the following best defines the hybrid inheritance?
a) Combination of two or more inheritance types
b) Combination of same type of inheritance
c) Inheritance of more than 7 classes
d) Inheritance involving all the types of inheritance

Answer: a
Explanation: When more than one type of inheritance are used together, it results in new type of inheritance which is in general known as hybrid inheritance. This may of may not have better capabilities.

222. How many types of inheritance should be used for hybrid?
a) Only 1
b) At least 2
c) At most two
d) Always more than 2

Answer: b
Explanation: There must be combination of at least 2 types of inheritance. The inheritance should be of different type.

223. Which of the following is the correct syntax of including a user defined header files in C++?
a) #include <userdefined.h>
b) #include <userdefined>
c) #include "userdefined"
d) #include [userdefined]

Answer: c
Explanation: C++ uses double quotes to include a user-defined header file. The correct syntax of including user-defined is #include "userdefinedname".

224. Which of the following is a correct identifier in C++?
a) 7var_name
b) 7VARNAME
c) VAR_1234
d) $var_name

Answer: c
Explanation: The rules for writing an identifier is as follows:
i) may contain lowercase/uppercase letters, digits or underscore(_) only
ii) should start with a non-digit character
iii) should not contain any special characters like @, $, etc.

225. Which of the following is called address operator?
a) *
b) &
c) _
d) %

Answer: b
Explanation: & operator is called address

operator and is used to access the address of a variable.

226. Which of the following is used for comments in C++?
a) // comment
b) /* comment */
c) both // comment or /* comment */
d) // comment */

Answer: c
Explanation: Both the ways are used for commenting in C++ programming. // is used for single line comments and /* … */ is used for multiple line comments.

227. What are the actual parameters in C++?
a) Parameters with which functions are called
b) Parameters which are used in the definition of a function
c) Variables other than passed parameters in a function
d) Variables that are never used in the function

Answer: a
Explanation: Actual parameters are those using which a function call is made i.e. which are actually passed in a function when that function is called.

228. What are the formal parameters in C++?
a) Parameters with which functions are called
b) Parameters which are used in the definition of the function
c) Variables other than passed parameters in a function
d) Variables that are never used in the function

229. Which function is used to read a single character from the console in C++?
a) cin.get(ch)
b) getline(ch)
c) read(ch)
d) scanf(ch)

Answer: a
Explanation: C++ provides cin.get() function to read a single character from console whereas others are used to read either a single or multiple characters.

230. Which function is used to write a single character to console in C++?
a) cout.put(ch)
b) cout.putline(ch)
c) write(ch)
d) printf(ch)

Answer: a
Explanation: C++ provides cout.put() function to write a single character to console whereas others are used to write either a single or multiple characters.

231. What are the escape sequences?
a) Set of characters that convey special meaning in a program
b) Set of characters that whose use are avoided in C++ programs
c) Set of characters that are used in the name of the main function of the program
d) Set of characters that are avoided in cout statements

Answer: a
Explanation: Escape sequence is a set of characters that convey a special meaning to the program. They are used to convey a meaning which cannot be conveyed directly.

232. Which of the following escape sequence represents carriage return?

a) \r
b) \n
c) \n\r
d) \c

Answer: a
Explanation: \r is used to represent carriage return which means move the cursor to the beginning of the next line.

233. Which of the following escape sequence represents tab?
a) \t
b) \t\r
c) \b
d) \a

Answer: a
Explanation: \t is used to represent tab which means a set of blank spaces in a line.

234. Who created C++?
a) Bjarne Stroustrup
b) Dennis Ritchie
c) Ken Thompson
d) Brian Kernighan

Answer: a
Explanation: Bjarne Stroustrup is the original creator of C++ during 1979 at AT&T Bell Labs.

235. Which of the following is called insertion/put to operator?
a) <<
b) >>
c) >
d) <

Answer: a
Explanation: << operator is called insertion or put to operator i.e. insert/put things to console/files.

236. Which of the following is called extraction/get from operator?
a) <<
b) >>
c) >
d) <

Answer: b
Explanation: >> operator is called extraction or get from operator i.e. extract/get things from console/files.

237. A language which has the capability to generate new data types are called _____
a) Extensible
b) Overloaded
c) Encapsulated
d) Reprehensible

Answer: a
Explanation: Languages that can produce/generate new data types are called extensible languages as they have the ability to handle new data types.

238. What is the size of wchar_t in C++?
a) 2
b) 4
c) 2 or 4
d) Based on the number of bits in the system

Answer: d
Explanation: Compiler wants to make CPU as more efficient in accessing the next value.

239. Pick the odd one out.
a) array type
b) character type
c) boolean type
d) integer type

Answer: a
Explanation: Array type is not the basic type and it is constructed using the basic type.

240. Which data type is used to represent the absence of parameters?
a) int
b) short
c) void
d) float

Answer: c
Explanation: Because void specifies an empty set of values/parameters.

241. What does '\a' escape code represent?
a) alert
b) backslash
c) tab
d) form feed

Answer: a
Explanation: Because \a is used to produce a beep sound.

242. Which type is best suited to represent the logical values?
a) integer
b) boolean
c) character
d) float

Answer: b
Explanation: Logical values can be either true or false, so the boolean type is suited for it.

243. Identify the user-defined types from the following?
a) enumeration
b) classes
c) both enumeration and classes
d) int

Answer: c
Explanation: They must be defined by the users before use, unlike the other types which are readily available.

244. Which operator is used to insert the data into file?
a) >>
b) <<
c) <
d) >

Answer: b
Explanation: You can write information to a file from your program using the stream insertion operator <<.

245. Which function is used to position back from the end of file object?
a) seekg
b) seekp
c) both seekg & seekp
d) seekf

Answer: a
Explanation: The member function seekg is used to position back from the end of file object.

246. How many objects are used for input and output to a string?
a) 1
b) 2
c) 3
d) 4

Answer: c
Explanation: The stringstream, ostringstream, and istringstream objects are used for input and output to a string.

247. What will be the output of the following C++ code?

```
1.  #include <iostream>
2.  #include <fstream>
3.  using namespace std;
4.  int main ()
5.  {
6.      int length;
7.      char * buffer;
8.      ifstream is;
9.      is.open ("sample.txt", ios :: binary );
10.     is.seekg (0, ios :: end);
11.     length = is.tellg();
12.     is.seekg (0, ios :: beg);
13.     buffer = new char [length];
14.     is.read (buffer, length);
15.     is.close();
16.     cout.write (buffer, length);
17.     delete[] buffer;
18.     return 0;
19. }
```

a) This is sample
b) sample
c) Error
d) Runtime error

Answer: d
Explanation: In this program, if the file exist, it will read the file. Otherwise it will throw an exception. A runtime error will occur because the value of the length variable will be “-1” if file doesn’t exist and in line 13 we are trying to allocate an array of size “-1”.

248. What will be the output of the following C++ code?

```
1.  #include <iostream>
2.  using namespace std;
3.  int main ()
4.  {
5.      char first, second;
6.      cout << "Enter a word: ";
7.      first = cin.get();
8.      cin.sync();
9.      second = cin.get();
10.     cout << first << endl;
11.     cout << second << endl;
12.     return 0;
13. }
```

a) first
b) second
c) returns first 2 letter or number from the entered word
d) third

Answer: c
Explanation: In this program, We are using the sync function to return the first two letters of the entered word.
Output:

```
$ g++ stream.cpp
$ a.out
Enter a word: steve
s
t
```

249. Which header file is required to use file I/O operations?
a) <ifstream>
b) <ostream>
c) <fstream>
d) <iostream>

Answer: c
Explanation: <fstream> header file is needed to use file I/O operations in C++. This header file contains all the file I/O operations definition.

250. Which of the following is used to create an output stream?
a) ofstream

b) ifstream
c) iostream
d) fsstream

Answer: a
Explanation: ofstream is used to create an output stream in C++ file handling operations. Ofstream objects are used to read files.

251. Which of the following is used to create a stream that performs both input and output operations?
a) ofstream
b) ifstream
c) iostream
d) fstream

Answer: d
Explanation: fstream is used to create a stream that performs both input and output operations in C++ file handling.

252. Which of the following is not used as a file opening mode?
a) ios::trunc
b) ios::binary
c) ios::in
d) ios::ate

Answer: a
Explanation: ios::trunc is used to truncate a file if it exists. It is not a file opening mode.

253. Which of the following statements are correct?

1) It is not possible to combine two or more file opening mode in open() method.

2) It is possible to combine two or more file opening mode in open() method.

3) ios::in and ios::out are input and output file opening mode respectively.

a) 1, 3
b) 2, 3
c) 3 only
d) 1, 2

Answer: a
Explanation: C++ allows to use one or more file opening mode in a single open() method. ios::in and ios::out are input and output file opening mode respectively.

254. By default, all the files in C++ are opened in _____ mode.
a) Text
b) Binary
c) ISCII
d) VTC

Answer: a
Explanation: By default, all the files in C++ are opened in text mode. They read the file as normal text.

255. What is the use of ios::trunc mode?
a) To open a file in input mode
b) To open a file in output mode
c) To truncate an existing file to half
d) To truncate an existing file to zero

Answer: d
Explanation: In C++ file handling, ios::trunc mode is used to truncate an existing file to zero length.

256. How many groups of output of operation are there in c++?
a) 1
b) 2
c) 3
d) 4

Answer: b
Explanation: There are two groups of output operation in c++. They are formatted output and unformatted output.

257. Pick out the correct objects about the instantiation of output stream.
a) cout
b) cerr
c) clog
d) all of the mentioned

Answer: d
Explanation: cout, cerr and clog are the standard objects for the instantiation of output stream class.

258. What is meant by ofstream in c++?
a) Writes to a file
b) Reads from a file
c) Writes to a file & Reads from a file
d) delete a file

Answer: a
Explanation: ofstream is a stream class to write on files.

259. Which operator is used for input stream?
a) >
b) >>
c) <
d) <<

Answer: b
Explanation: The operator of extraction is >> and it is used on the standard input stream.

260. Where does a cin stops it extraction of data?
a) By seeing a blank space
b) By seeing (
c) By seeing a blank space & (
d) By seeing <

Answer: a
Explanation: cin will stop its extraction when it encounters a blank space.

261. Which of the following is used for generic programming?
a) Virtual functions
b) Modules
c) Templates
d) Abstract Classes

Answer: c
Explanation: Templates are used for generic programming. They help in making generic functions and classes hence achieving the generic codes.

262. Which of the following is correct about templates?
a) It is a type of compile time polymorphism
b) It allows the programmer to write one code for all data types
c) Helps in generic programming
d) All of the mentioned

Answer: d
Explanation: Templates are used for generic programming hence allowing to write a single function for all data types. It is a type of compile time polymorphism.

263. What will be the output of the following C++ code?

```
#include <iostream>
using namespace std;
template <class T>
T max (T &a, T &b)
{
            cout << "Template Called ";
  return (a > b)? a : b;
}

template <>
int max <int> (int &a, int &b)
{
  cout << "Called ";
  return (a > b)? a : b;
}

int main ()
{
  int a = 10, b = 20;
  cout << max <int> (a, b);
}
```

a) Template Called 20
b) Called 20
c) Error
d) Segmentation fault

Answer: b
Explanation: For T = int we have created a separate definition for the above template function. Hence the call using int takes the newly defined function.

264. What will be the output of the following C++ code?

```
#include <iostream>
using namespace std;
template<int n>
struct funStruct
{
  static const int val = 2*funStruct<n-1>::val;
};

template<>
struct funStruct<0>
{
  static const int val = 1 ;
};

int main()
{
  cout << funStruct<10>::val << endl;
  return 0;
}
```

a) 1
b) 1024
c) Error
d) Segmentation fault

Answer: b
Explanation: The above call for struct will call the first struct for n > 0 and second one when n = 0. Therefore when value of n = 10 the until n becomes 0 first struct is called so we will call 2*2*2…10 times*1 which will give the result 2^{10} = 1024.

265. What is an exception in C++ program?
a) A problem that arises during the execution of a program
b) A problem that arises during compilation
c) Also known as the syntax error
d) Also known as semantic error

Answer: a
Explanation: An exception is defined as the problem in C++ program that arises during the execution of the program for example divide by zero error.

266. By default, what a program does when it detects an exception?
a) Continue running
b) Results in the termination of the program
c) Calls other functions of the program
d) Removes the exception and tells the programmer about an exception

Answer: b
Explanation: By default, whenever a program detects an exception the program crashes as it does not know how to handle it hence results in the termination of the program.

267. Why do we need to handle exceptions?
a) To avoid unexpected behaviour of a program during run-time
b) To let compiler remove all exceptions by itself
c) To successfully compile the program
d) To get correct output

Answer: a
Explanation: We need to handle exceptions in a program to avoid any unexpected behaviour during run-time because that behaviour may affect other parts of the program. Also, an exception is detected during run-time, therefore, a program may compile successfully even with some exceptions cases in your program.

268. How Exception handling is implemented in the C++ program?
a) Using Exception keyword
b) Using try-catch block
c) Using Exception block
d) Using Error handling schedules

Answer: b
Explanation: C++ provides a try-catch block to handle exceptions in your program.

269. Which part of the try-catch block is always fully executed?
a) try part
b) catch part
c) finally part
d) throw part

Answer: c
Explanation: finally part of the try-catch block is always executed whether exceptions are caught or not.

270. Which of the following is an exception in C++?
a) Divide by zero
b) Semicolon not written
c) Variable not declared
d) An expression is wrongly written

Answer: a
Explanation: Exceptions are those which are encountered during run-time of the program. semicolon, variable not declared and the wrong expression are compile-time errors, therefore, they are not exceptions. Divide by zero is the problem that is encountered during run-time, therefore, it is an exception.

271. Where should we place catch block of the derived class in a try-catch block?
a) Before the catch block of Base class
b) After the catch block of Base class
c) Anywhere in the sequence of catch blocks
d) After all the catch blocks

Answer: a
Explanation: C++ asks the programmer to place the catch block of derived class before a catch block of the base class, otherwise derived catch block will never be executed.

272. What id the syntax for catching any type of exceptions?
a) catch(Exception e)
b) catch(…)
c) catch(Exception ALL)
d) catch(ALL)

Answer: b
Explanation: catch(…) is used in C++ to catch all types of exceptions in a single catch block.

273. The C++ code which causes abnormal termination/behaviour of a program should be written under _____ block.
a) try
b) catch
c) finally
d) throw

Answer: a
Explanation: Code that leads to the abnormal termination of the program should be written under the try block.

274. Exception handlers are declared with _____ keyword.
a) try
b) catch
c) throw
d) finally

Answer: b
Explanation: C++ uses catch block to handle any exceptions that occur during run-time of the program.

275. Which of the following statements are correct about Catch handler?

i. It must be placed immediately after the try block

ii. It can have more than one parameters

iii. There must be one and only one catch handler for every try block

iv. There can be multiple catch handler for a try block

v. General catch handler can be kept anywhere after try block.

a) i, iv, v
b) i, ii, iii
c) i, iv
d) i, ii

Answer: c
Explanation: A catch block should always be placed after the try block and there can be multiple catch block following a try block.

276. What is the role of a constructor in classes?
a) To modify the data whenever required
b) To destroy an object
c) To initialize the data members of an object when it is created
d) To call private functions from the outer world

Answer: c
Explanation: A constructor is used in classes to initialize data members of class in order to avoid errors/segmentation faults.

277. Why constructors are efficient instead of a function init() defined by the user to initialize the data members of an object?
a) Because user may forget to call init() using that object leading segmentation fault
b) Because user may call init() more than once which leads to overwriting values
c) Because user may forget to define init() function
d) All of the mentioned

Answer: d
Explanation: We cannot use init() because as mentioned in options that user may forget to initialize the members which will lead to a segmentation fault. Also if the user calls the init() function more than once it may overwrite the values and may result into disastrous results. Also if any user forgets to define init() function then no object will be initialized whereas if any constructor is not defined in any class the class provides a default constructor for initialization.

278. What is a copy constructor?
a) A constructor that allows a user to move data from one object to another
b) A constructor to initialize an object with the values of another object
c) A constructor to check the whether to objects are equal or not
d) A constructor to kill other copies of a given object.

Answer: b
Explanation: Copy constructor allows the user to initialize an object with the values of another object instead of supplying the same set of values again to initialize the object.

279. What will be the output of the following C++ code?

```
#include <iostream>
#include <string>
using namespace std;
class A{
        int a;
public:
        A(int i){
                a = i;
        }
        void assign(int i){
                a = i;
        }
        int return_value(){
                return a;
        }
};
int main(int argc, char const *argv[])
{
        A obj;
        obj.assign(5);
        cout<<obj.return_value();
}
```

a) 5
b) 55
c) Error
d) Segmentation Fault

Answer: c
Explanation: As we have defined a constructor which takes an int parameter, so when we are trying to declare an object obj of class A without supplying any parameter then as a constructor is overwritten it will give an error saying that no matching function found. So whenever one writes a constructor then the default constructor is overwritten hence if you want to declare an object without parameter then you also have to define that constructor.

280. In the following C++ code how many times the string “A’s constructor called” will be printed?

```
#include <iostream>
#include <string>
using namespace std;
class A{
        int a;
public:
        A(){
                cout<<"A's constructor
called";
        }
};
class B{
        static A a;
public:
        B(){
                cout<<"B's constructor
called";
        }
        static A get(){
                return a;
        }
};
A B::a;
int main(int argc, char const *argv[])
{
        B b;
        A a1 = b.get();
        A a2 = b.get();
        A a3 = b.get();
}
```

a) 3
b) 4
c) 2
d) 1

Answer: d
Explanation: As the object is defined ony once in the program at line A B::a, so the constructor of A is called only once. For objects a1, a2 and a3 copy constructor is called so the string will not be printed for them.

281. What happens if a user forgets to define a constructor inside a class?
a) Error occurs
b) Segmentation fault
c) Objects are not created properly
d) Compiler provides a default constructor to avoid faults/errors

Answer: d
Explanation: The C++ compiler always provides a default constructor if one forgets to define a constructor inside a class.

282. How many parameters does a default constructor require?
a) 1
b) 2
c) 0
d) 3

Answer: c
Explanation: A default constructor does not require any parameters for object creation that’s why sometimes we declare an object without any parameters.

283. How constructors are different from other member functions of the class?
a) Constructor has the same name as the class itself
b) Constructors do not return anything
c) Constructors are automatically called when an object is created
d) All of the mentioned

Answer: d
Explanation: All the above mention are the reasons where constructor differs from other normal member functions of a class.

284. How many types of constructors are there in C++?
a) 1
b) 2
c) 3
d) 4

Answer: c
Explanation: There are three types of constructors in C++ namely default, parameterized and copy constructor.

283. What is the difference between constructors and destructors?
a) They have a different function name
b) Constructors does not have return type whereas destructors do have
c) Constructors allow function parameters whereas destructors do not
d) Constructors does not function parameters

284. How many Destructors are allowed in a Class?
a) 1
b) 2
c) 3
d) Any number

Answer: a
Explanation: A class in C++ allows only one destructor, which is called whenever the lifetime of an object ends.

285. Which of the following constructors are provided by the C++ compiler if not defined in a class?
a) Default constructor
b) Assignment constructor
c) Copy constructor
d) All of the mentioned

Answer: d
Explanation: If a programmer does not define the above constructors in a class the C++ compiler by default provides these constructors to avoid error on basic operations.

286. When a copy constructor is called?
a) When an object of the class is returned by value
b) When an object of the class is passed by value to a function
c) When an object is constructed based on another object of the same class
d) All of the mentioned

Answer: d
Explanation: Copy constructor is called in all the above-mentioned criteria because in all the above cases we are somehow trying to copy one object into another.

287. Pick the other name of operator function.
a) function overloading
b) operator overloading
c) member overloading
d) object overloading

Answer: b
Explanation: Operator function means operation defined for that operator so if user defines a function for an operator then that is called operator overloading i.e. overloading already present operator function.

288. Which of the following operators can't be overloaded?
a) ::

b) +
c) –
d) []

Answer: a
Explanation: :: operator cannot be overloaded because this operator operates on names rather than values and C++ has no syntax for writing codes that works on names than values so using syntax these operators cannot be overloaded.

289. How to declare operator function?
a) operator sign
b) operator
c) name of the operator
d) name of the class

Answer: a
Explanation: We have to declare the operator function by using the operator, operator sign. Example "operator +" where the operator is a keyword and + is the symbol need to be overloaded.

290. Which of the following statements is NOT valid about operator overloading?
a) Only existing operators can be overloaded
b) The overloaded operator must have at least one operand of its class type
c) The overloaded operators follow the syntax rules of the original operator
d) None of the mentioned

Answer: b
Explanation: The overloaded operator must not have at least one operand of its class type.

291. Operator overloading is _____
a) making c++ operator works with objects
b) giving new meaning to existing operator
c) making the new operator
d) adding operation to the existing operators

Answer: d
Explanation: Operator overloading is the way adding operation to the existing operators.

292. Which operator is having the right to left associativity in the following?
a) Array subscripting
b) Function call
c) Addition and subtraction
d) Type cast

Answer: d
Explanation: There are many rights to left associativity operators in C++, which means they are evaluation is done from right to left. Type Cast is one of them. Here is a link of the associativity of operators: https://github.com/MicrosoftDocs/cpp-docs/blob/master/docs/cpp/cpp-built-in-operators-precedence-and-associativity.md

293. Which operator is having the highest precedence?
a) postfix
b) unary
c) shift
d) equality

Answer: a
Explanation: The operator which is having the highest precedence is postfix and lowest is equality.

294. What is this operator called ?:?
a) conditional

b) relational
c) casting operator
d) unrelational

Answer: a
Explanation: In this operator, if the condition is true means, it will return the first operator, otherwise second operator.

295. What will be the output of the following C++ code?

```
1.  #include <iostream>
2.  using namespace std;
3.  int main()
4.  {
5.     int a;
6.     a = 5 + 3 * 5;
7.     cout << a;
8.     return 0;
9.  }
```

a) 35
b) 20
c) 25
d) 30

Answer: b
Explanation: Because the * operator is having highest precedence, So it is executed first and then the + operator will be executed.
Output:

```
$ g++ op1.cpp
$ a.out
20
```

296. What is the use of dynamic_cast operator?
a) it converts virtual base class to derived class
b) it converts the virtual base object to derived objects
c) it will convert the operator based on precedence
d) it converts the virtual base object to derived class

Answer: a
Explanation: Because the dynamic_cast operator is used to convert from base class to derived class.

297. What will be the output of the following C++ code?

```
1.  #include <iostream>
2.  using namespace std;
3.  int main()
4.  {
5.     int a = 5, b = 6, c, d;
6.     c = a, b;
7.     d = (a, b);
8.     cout << c << ' ' << d;
9.     return 0;
10. }
```

6 a) 5
5 b) 6
7 c) 6
8 d) 6

Answer: a
Explanation: It is a separator here. In C, the value a is stored in c and in d the value b is stored in d because of the bracket.
Output:

```
$ g++ op3.cpp
$ a.out
5  6
```

298. What will be the output of the following C++ code?

```
1.  #include <iostream>
2.  using namespace std;
3.  int main()
4.  {
```

```
5.      int i, j;
6.      j = 10;
7.      i = (j++, j + 100, 999 + j);
8.      cout << i;
9.      return 0;
10.  }
```

a) 1000
b) 11
c) 1010
d) 1001

Answer: c
Explanation: j starts with the value 10. j is then incremented to 11. Next, j is added to 100. Finally, j (still containing 11) is added to 999 which yields the result 1010.
Output:

```
$ g++ op2.cpp
$ a.out
1010
```

299. What will be the output of the following C++ code?

```
1.   #include <iostream>
2.   using namespace std;
3.   int main ()
4.   {
5.      int x, y;
6.      x = 5;
7.      y = ++x * ++x;
8.      cout << x << y;
9.      x = 5;
10.     y = x++ * ++x;
11.     cout << x << y;
12.     return 0;
13.  }
```

a) 749735
b) 736749
c) 367497
d) 367597

Answer: a
Explanation: Because of the precedence the pre-increment and post increment operator, we got the output as 749736.
Output:

```
$ g++ op.cpp
$ a.out
749735
```

300. What will be the output of the following C++ code?

```
1.   #include <iostream>
2.   using namespace std;
3.   int main()
4.   {
5.      int a = 5, b = 6, c;
6.      c = (a > b) ? a : b;
7.      cout << c;
8.      return 0;
9.   }
```

a) 6
b) 5
c) 4
d) 7

Answer: a
Explanation: Here the condition is false on conditional operator, so the b value is assigned to c.
Output:

```
$ g++ op1.cpp
$ a.out
6
```

301. What will be the output of the following C++ code?

```
1.   #include <iostream>
2.   using namespace std;
3.   main()
4.   {
5.      double a = 21.09399;
```

```
6.      float b = 10.20;
7.      int c ,d;
8.      c = (int) a;
9.      d = (int) b;
10.     cout << c <<' '<< d;
11.     return 0;
12.  }
```

10 a) 20
21 b) 10
10 c) 21
20 d) 10

Answer: c
Explanation: In this program, we are casting the operator to integer, So it is printing as 21 and 10.
Output:

```
$ g++ op5.cpp
$ a.out
21        10
```

302. How are many sequences of statements present in c++?
a) 4
b) 3
c) 5
d) 6

Answer: c
Explanation: There are five sequences of statements. They are Preprocessor directives, Comments, Declarations, Function Declarations, Executable statements.

303. The if..else statement can be replaced by which operator?
a) Bitwise operator
b) Conditional operator
c) Multiplicative operator
d) Addition operator

Answer: b
Explanation: In the conditional operator, it will predicate the output using the given condition.

304. The switch statement is also called as?
a) choosing structure
b) selective structure
c) certain structure
d) bitwise structure

Answer: b
Explanation: The switch statement is used to choose the certain code to execute, So it is also called as selective structure.

305. What will be the output of the following C++ code?

```
1.   #include <iostream>
2.   using namespace std;
3.   int main ()
4.   {
5.      int n;
6.      for (n = 5; n > 0; n--)
7.      {
8.         cout << n;
9.         if (n == 3)
10.           break;
11.     }
12.     return 0;
13.  }
```

a) 543
b) 54
c) 5432
d) 53

Answer: a
Explanation: In this program, We are printing the numbers in reverse order but by using break statement we stopped printing on 3.
Output:

```
$ g++ stat.cpp
$ a.out
```

```
543
```

306. What will be the output of the following C++ code?

```
1.  #include <iostream>
2.  using namespace std;
3.  int main()
4.  {
5.     int a = 10;
6.     if (a < 15)
7.     {
8.        time:
9.        cout << a;
10.       goto time;
11.    }
12.    break;
13.    return 0;
14. }
```

a) 1010
b) 10
c) infinitely print 10
d) compile time error

Answer: d
Explanation: Because the break statement need to be presented inside a loop or a switch statement.

307. What will be the output of the following C++ code?

```
1.  #include <iostream>
2.  using namespace std;
3.  int main()
4.  {
5.     int n = 15;
6.     for ( ; ;)
7.     cout << n;
8.     return 0;
9.  }
```

a) error
b) 15
c) infinite times of printing n
d) none of the mentioned

Answer: c
Explanation: There is not a condition in the for loop, So it will loop continuously.

308. What will be the output of the following C++ code?

```
1.  #include <iostream>
2.  using namespace std;
3.  int main()
4.  {
5.     int i;
6.     for (i = 0; i < 10; i++);
7.     {
8.        cout << i;
9.     }
10.    return 0;
11. }
```

a) 0123456789
b) 10
c) 012345678910
d) compile time error

Answer: b
Explanation: for loop with a semicolon is called as body less for loop. It is used only for incrementing the variable values. So in this program the value is incremented and printed as 10.
Output:

```
$ g++ stat2.cpp
$ a.out
10
```

309. How many types of loops are there in C++?
a) 4
b) 2

c) 3
d) 1

Answer: a
Explanation: There are four types of loop. They are the while, do while, nested, for the loop.

310. Which looping process is best used when the number of iterations is known?
a) for
b) while
c) do-while
d) all looping processes require that the iterations be known

Answer: a
Explanation: Because in for loop we are allowed to provide starting and ending conditions of loops, hence fixing the number of iterations of loops, whereas no such things are provided by other loops.

311. Which of the following is the default return value of functions in C++?
a) int
b) char
c) float
d) void

Answer: a
Explanation: C++ uses int as the default return values for functions. It also restricts that the return type of the main function must be int.

312. What happens to a function defined inside a class without any complex operations (like looping, a large number of lines, etc)?
a) It becomes a virtual function of the class
b) It becomes a default calling function of the class
c) It becomes an inline function of the class
d) The program gives an error

Answer: c
Explanation: Any function which is defined inside a class and has no complex operations like loops, a large number of lines then it is made inline.

313. What is an inline function?
a) A function that is expanded at each call during execution
b) A function that is called during compile time
c) A function that is not checked for syntax errors
d) A function that is not checked for semantic analysis

Answer: a
Explanation: Inline function is those which are expanded at each call during the execution of the program to reduce the cost of jumping during execution.

314. An inline function is expanded during _____
a) compile-time
b) run-time
c) never expanded
d) end of the program

Answer: a
Explanation: An inline function is expanded during the compile-time of a program.

315. How many ways of passing a parameter are there in c++?
a) 1
b) 2

c) 3
d) 4

Answer: c
Explanation: There are three ways of passing a parameter. They are pass by value,pass by reference and pass by pointer.

316. Which is used to keep the call by reference value as intact?
a) static
b) const
c) absolute
d) virtual

Answer: b
Explanation: Because const will not change the value of the variables during the execution.

317. By default how the value are passed in c++?
a) call by value
b) call by reference
c) call by pointer
d) call by object

Answer: a
Explanation: None.

318. What will be the output of the following C++ code?

```
1.  #include <iostream>
2.  using namespace std;
3.  void copy (int& a, int& b, int&
    c)
4.  {
5.      a *= 2;
6.      b *= 2;
7.      c *= 2;
8.  }
9.  int main ()
10. {
11.     int x = 1, y = 3, z = 7;
12.     copy (x, y, z);
13.     cout << "x =" << x << ", y ="
    << y << ", z =" << z;
14.     return 0;
15. }
```

a) 2 5 10
b) 2 4 5
c) 2 6 14
d) 2 4 9

Answer: c
Explanation: Because we multiplied the values by 2 in the copy function.
Output:

```
$ g++ arg6.cpp
$ a.out
x = 2,y = 6,z = 14
```

319. What will happen when we use void in argument passing?
a) It will not return value to its caller
b) It will return value to its caller
c) Maybe or may not be return any value to its caller
d) It will return value with help of object

Answer: a
Explanation: As void is not having any return value, it will not return the value to the caller.

320. What will be the output of the following C++ code?

```
1.  #include <iostream>
2.  using namespace std;
3.  void Sum(int a, int b, int & c)
4.  {
5.      a = b + c;
6.      b = a + c;
7.      c = a + b;
```

```
8.      }
9.      int main()
10.     {
11.        int x = 2, y =3;
12.        Sum(x, y, y);
13.        cout << x << " " << y;
14.        return 0;
15.     }
```

a) 2 3
b) 6 9
c) 2 15
d) compile time error

Answer: c
Explanation: We have passed three values and it will manipulate according to the given condition and yield the result as 2 15
Output:

```
$ g++ arg.cpp
$ a.out
2 15
```

321. What is Inheritance in C++?
a) Wrapping of data into a single class
b) Deriving new classes from existing classes
c) Overloading of classes
d) Classes with same names

Answer: b
Explanation: Inheritance is the concept of OOPs in which new classes are derived from existing classes in order to reuse the properties of classes defined earlier.

322. How many specifiers are used to derive a class?
a) 1
b) 2
c) 3
d) 4

Answer: c
Explanation: There are 3 specifiers used to derive a class. They are private, protected and public.

323. Which specifier makes all the data members and functions of base class inaccessible by the derived class?
a) private
b) protected
c) public
d) both private and protected

Answer: a
Explanation: Private access specifier is used to make all the data members and functions of the base class inaccessible.

324. If a class is derived privately from a base class then

_____

a) no members of the base class is inherited
b) all members are accessible by the derived class
c) all the members are inherited by the class but are hidden and cannot be accessible
d) no derivation of the class gives an error

Answer: c
Explanation: Whenever a class is derived, all the members of the base class is inherited by the derived class but are not accessible by the derived class.

325. What is a virtual function in C++?
a) Any member function of a class
b) All functions that are derived from the base class
c) All the members that are accessing base class data members
d) All the functions which are declared in

the base class and is re-defined/overridden by the derived class

Answer: d
Explanation: Virtual function is a function that is declared inside the base class and is re-defined inside the derived class.

326. Which is the correct syntax of declaring a virtual function?
a) virtual int func();
b) virtual int func(){};
c) inline virtual func();
d) inline virtual func(){};

Answer: a
Explanation: To make a function virtual function we just need to add virtual keyword at the starting of the function declaration.

327. What will be the output of the following C++ code?

```
#include <iostream>
#include <string>
using namespace std;
class A{
        float d;
  public:
        virtual void func(){
                cout<<"Hello this is class A\n";
        }
};

class B: public A{
        int a = 15;
public:
        void func(){
                cout<<"Hello this is class B\n";
        }
};

int main(int argc, char const *argv[])
{
        B b;
        b.func();
        return 0;
}
```

a) Hello this is class B
b) Hello this is class A
c) Error
d) Segmentation fault

Answer: a
Explanation: Normal execution of the program and object calls func() from class B.

328. What will be the output of the following C++ code?

```
#include <iostream>
#include <string>
using namespace std;
class A
{
        float d;
    public:
        virtual void func(){
                cout<<"Hello this is class A\n";
        }
};

class B: public A
{
        int a = 15;
  public:
        void func(){
                cout<<"Hello this is class B\n";
        }
};

int main(int argc, char const *argv[])
```

```
{
        A *a;
        a->func();
        return 0;
}
```

a) Hello this is class A
b) Hello this is class B
c) Error
d) Segmentation Fault

Answer: d
Explanation: As object 'a' is a pointer object and we know every pointer needs to be initialised memory before use. Hence segmentation fault. Use A *a = new A(); to initialise memory to the object.

329. Which statement is incorrect about virtual function.
a) They are used to achieve runtime polymorphism
b) They are used to hide objects
c) Each virtual function declaration starts with the virtual keyword
d) All of the mentioned

Answer: b
Explanation: Virtual function are used to achieve runtime polymorphism by calling the right function during runtime. Their declaration starts with a virtual keyword.

330. The concept of deciding which function to invoke during runtime is called _____
a) late binding
b) dynamic linkage
c) static binding
d) both late binding and dynamic linkage

Answer: d
Explanation: The concept of deciding which function to invoke during runtime is called late binding or dynamic linkage. Late binding because function binding to the object is done during runtime. Dynamic linkage because this binding is done during runtime.

331. What is a pure virtual function?
a) A virtual function defined inside the base class
b) A virtual function that has no definition relative to the base class
c) A virtual function that is defined inside the derived class
d) Any function that is made virtual

Answer: b
Explanation: A virtual function that has no definition relative to the base class is called a pure virtual function.

332. Which attribute specifies a unique alphanumeric identifier to be associated with an element?
a) class
b) id
c) article
d) html

Answer: b
Explanation: HTML is Hyper Text Markup Language which is used to create web pages and applications. The id attribute is most used to point to a style in a style sheet, and by JavaScript (via the HTML DOM) to manipulate the element with the specific id. Class is a name given to HTML elements which can be used by CSS and JavaScript for styling the web pages. A self-contained content is called attribute.

333. The _____ attribute specifies an inline style associated with an element,

which determines the rendering of the affected element.
a) dir
b) style
c) class
d) article

Answer: b
Explanation: Style attribute specifies an inline style for an element. The style attribute will override any style set globally, e.g. styles specified in the style tag or in an external style sheet. A self-contained content is called attribute. Class is a name given to HTML elements which can be used by CSS and JavaScript for styling the web pages. List of directory files is given by dir tag which is not supported in HTML5.

334. Which attribute is used to provide an advisory text about an element or its contents?
a) tooltip
b) dir
c) title
d) head

Answer: c
Explanation: The extra information about an element is specified by title tag. The information is most often shown as a tooltip text when the mouse moves over the element. List of directory files is given by dir tag which is not supported in HTML5. Tooltip or else infotip is a graphical user interface of an element. Container of metadata is called head.

335. The _____ attribute sets the text direction as related to the lang attribute.
a) lang
b) sub
c) dir
d) ds

Answer: c
Explanation: The dir attribute specifies the text direction of the element's content. List of directory files is given by dir tag which is not supported in HTML5. The language of an element's content is given by lang attribute. The subscript text is defined by sub attribute.

336. Which of the following is the attribute that specifies the column name from the data source object that supplies the bound data?
a) dataFormatAs
b) datafld
c) disabled
d) datasrc

Answer: b
Explanation: DataFormatAs specifies how data is rendered. The identifier for data source is set by dataSrc. Datafld attribute specifies the column name from the data source object that supplies the bound data. This attribute is specific to Microsoft's data binding. A Disabled is a boolean attribute which specifies that <input> element should be disabled.

337. Which of the following is the attribute that indicates the name of the data source object that supplies the data that is bound to this element?
a) dataFormatAs
b) datafld

c) disabled
d) datasrc

Answer: d
Explanation: The identifier for data source is set by dataSrc. When the dataSrc property is applied to a tabular data consumer, the entire data set is repeated by the consuming elements. DataFormatAs specifies how data is rendered. A Disabled is a boolean attribute which specifies that <input> element should be disabled.

338. Which of the following is the attribute that specifies additional horizontal space, in pixels, to be reserved on either side of an embedded item like an iframe, applet, image, and so on?
a) height
b) hspace
c) hidefocus
d) datasrc

Answer: b
Explanation: Height of element is pixels is specified by height attribute. The hspace attribute specifies the whitespace on left and right side of an object. The hidefocus specifies whether a focused rectangle is drawn around an object. The identifier for data source is set by dataSrc.

339. The accesskey attribute specifies a keyboard navigation accelerator for the element.
a) True
b) False

Answer: a
Explanation: The accesskey attribute specifies a shortcut key to activate/focus an element. It specifies a keyboard navigation accelerator for the element. We can use accesskey attribute in forms or links.

340. Which feature was already introduced before HTML5?
a) Canvas/SVG
b) Video
c) Geolocation
d) Frames

Answer: d
Explanation: Until recently you could not draw on the web and graphics were not very interactive but canvas/SVG which were introduced by HTML5 solved all the problems. Video, Geolocation were also new features introduced by HTML5. Frames were already there before the launch of HTML5.

341. Which tag is used with JavaScript?
a) <canvas>
b) <table>
c) <article>
d) <footer>

Answer: a
Explanation: <canvas> tag is basically used for graphics via scripting i.e. usually with Java Script (scripting language, basically used to create animations). Other tags like <table>, <article>, <footer> can be used simply with HTML.

342. What application can one create even before the introduction of HTML5?
a) Web applications
b) Mobile applications

c) Forms
d) Browser based games

Answer: c
Explanation: With the help of HTML5 and JavaScript it became possible to create excellent mobile applications, browser based games, web applications and many more other applications. Forms were already introduced before HTML5.

343. JavaScript Code can be called by using _____
a) RMI
b) Triggering Event
c) Preprocessor
d) Function/Method

Answer: d
Explanation: JavaScript code can be called by making a function call to the element on which JavaScript has to be run. There are many other methods like onclick, onload, and onsubmit etc.

344. The type of a variable that is volatile is _____
a) Volatile variable
b) Mutable variable
c) Immutable variable
d) Dynamic variable

Answer: b
Explanation: The variables whose values can be changed are called mutable variable types. In JavaScript, only objects and arrays are mutable, not primitive values.

345. A hexadecimal literal begins with _____
a) 00
b) 0x
c) 0X
d) Both 0x and 0X

Answer: d
Explanation: Generally, X or x denotes hexadecimal values. So, any integer literal that begins with 0X or 0x denotes a hexadecimal number.

346. The generalised syntax for a real number representation is _____
a) [digits][.digits][(E|e)[(+|-)]digits]
b) [digits][+digits][(E|e)[(+|-)]digits]
c) [digits][(E|e)[(+|-)]digits]
d) [.digits][digits][(E|e)[(+|-)]digits]

Answer: a
Explanation: Floating-point literals may also be represented using exponential notation: a real number followed by the letter e (or E), followed by an optional plus or minus sign, followed by an integer exponent. This notation represents the real number multiplied by 10 to the power of the exponent.

347. JavaScript _____ when there is an indefinite or an infinite value during an arithmetic computation.
a) Prints an exception error
b) Prints an overflow error
c) Displays “Infinity”
d) Prints the value as such

Answer: c
Explanation: When the result of a numeric operation is larger than the largest representable number (overflow),

JavaScript prints the value as Infinity. Similarly, when a negative value becomes larger than the largest representable negative number, the result is negative infinity. The infinite values behave as you would expect: adding, subtracting, multiplying, or dividing them by anything results in an infinite value (possibly with the sign reversed).

348. Which of the following is not considered as an error in JavaScript?
a) Syntax error
b) Missing of semicolons
c) Division by zero
d) Missing of Bracket

Answer: c
Explanation: Division by zero is not an error in JavaScript: it simply returns infinity or negative infinity. There is one exception, however: zero divided by zero does not have a well defined value, and the result of this operation is the special not-a-number value, printed as NaN.

349. The escape sequence '\f' stands for
_____
a) Floating numbers
b) Representation of functions that returns a value
c) \f is not present in JavaScript
d) Form feed

Answer: d
Explanation: \f is the JavaScript escape sequence that stands for Form feed (\u000C). It skips to the start of the next page. (Applies mostly to terminals where the output device is a printer rather than a VDU).

350. The snippet that has to be used to check if "a" is not equal to "null" is
_____
a) if(a!=null)
b) if (!a)
c) if(a!null)
d) if(a!==null)

Answer: d
Explanation: A strict comparison (e.g., ===) is only true if the operands are of the same type and the contents match. The more commonly-used abstract comparison (e.g. ==) converts the operands to the same type before making the comparison. The not-equal operator !== compares 0 to null and evaluates to either true or false.

351. The statement a===b refers to
_____
a) Both a and b are equal in value, type and reference address
b) Both a and b are equal in value
c) Both a and b are equal in value and type
d) There is no such statement

Answer: c
Explanation: "===" operator is known as the strict comparison operator. A strict comparison (===) is only true if the operands are of the same type and the contents match.

352. Assume that we have to convert "false" that is a non-string to string. The command that we use is (without invoking the "new" operator).

a) false.toString()
b) String(false)
c) String newvariable=”false”
d) Both false.toString() and String(false)

Answer: d
Explanation: The three approaches for converting to string are: value.toString(),”” + value and String(value). A non-string can be converted in two ways without using a new operator false.toString () and String(false).

353. XML is a _____ markup language.
a) meta
b) beta
c) octa
d) peta

Answer: a
Explanation: Generally speaking, a meta language is a language used to describe a language. XML is a metalanguage that is used to describe a markup language.

354. XML uses _____ principle to formally describe the data.
a) DDL
b) DTD
c) DML
d) None of the mentioned

Answer: b
Explanation: A document type definition (DTD) is a set of markup declarations that define a document type for an SGML-family markup language (SGML, XML, HTML). A Document Type Definition (DTD) defines the legal building blocks of an XML document. It defines the document structure with a list of legal elements and attributes.

355. Which among the following are true for an Extensible markup language?
a) Human Readable/ Machine Readable
b) Extended from SGML
c) Developed by www consortium
d) All of the mentioned

Answer: d
Explanation: XML is an open format markup language with a filename extension of .xml.

356. Which of them have XML as their default format?
a) IWork
b) LibreOffice
c) OpenOffice
d) All of the mentioned

Answer: d
Explanation: More that hundred of document formats using XML syntax have been developed, including RSS, Atom, SOAP and XHTML.

357. A DTD is associated with a XML file by means of _____
a) Function
b) <!DOCTYPE>
c) Macros
d) None of the mentioned

Answer: b
Explanation: A document type definition defines the legal building blocks of an XML document .

358. Which of the following is not an example of electronic mark up?

a) HTML
b) LaTeX
c) PostScript
d) None of the mentioned

Answer: d
Explanation: There are three categories of electronic markup: presentational, procedural, and descriptive markup. Examples are XML, HTML, LaTeX, etc.

359. troff and nroff are _____ in Unix.
a) functions
b) typesetting tools
c) System sofwares
d) None of the mentioned

Answer: b
Explanation: Early examples of computer markup languages can be found in typesetting tools like troff and nroff in Unix.

360. Which of the following is not OOPS concept in Java?
a) Inheritance
b) Encapsulation
c) Polymorphism
d) Compilation

Answer: d
Explanation: There are 4 OOPS concepts in Java. Inheritance, Encapsulation, Polymorphism and Abstraction.

361. Which of the following is a type of polymorphism in Java?
a) Compile time polymorphism
b) Execution time polymorphism
c) Multiple polymorphism
d) Multilevel polymorphism

Answer: a
Explanation: There are two types of polymorphism in Java. Compile time polymorphism (overloading) and runtime polymorphism (overriding).

362. When does method overloading is determined?
a) At run time
b) At compile time
c) At coding time
d) At execution time

Answer: b
Explanation: Overloading is determined at compile time. Hence, it is also known as compile time polymorphism.

363. What is the range of short data type in Java?
a) -128 to 127
b) -32768 to 32767
c) -2147483648 to 2147483647
d) None of the mentioned

Answer: b
Explanation: Short occupies 16 bits in memory. Its range is from -32768 to 32767.

364. What is the range of byte data type in Java?
a) -128 to 127
b) -32768 to 32767
c) -2147483648 to 2147483647
d) None of the mentioned

Answer: a
Explanation: Byte occupies 8 bits in memory. Its range is from -128 to 127.

365. How constructor can be used for a servlet?
a) Initialization
b) Constructor function
c) Initialization and Constructor function
d) Setup() method

Answer: c
Explanation: We cannot declare constructors for interface in Java. This means we cannot enforce this requirement to any class which implements Servlet interface.
Also, Servlet requires ServletConfig object for initialization which is created by container.

366. Can servlet class declare constructor with ServletConfig object as an argument?
a) True
b) False

Answer: b
Explanation: ServletConfig object is created after the constructor is called and before init() is called. So, servlet init parameters cannot be accessed in the constructor.

367. What is the difference between servlets and applets?
i. Servlets execute on Server; Applets execute on browser
ii. Servlets have no GUI; Applet has GUI
iii. Servlets creates static web pages; Applets creates dynamic web pages
iv. Servlets can handle only a single request; Applet can handle multiple requests
a) i, ii, iii are correct
b) i, ii are correct
c) i, iii are correct
d) i, ii, iii, iv are correct

Answer: b
Explanation: Servlets execute on Server and doesn't have GUI. Applets execute on browser and has GUI.

368. Which of the following code is used to get an attribute in a HTTP Session object in servlets?
a) session.getAttribute(String name)
b) session.alterAttribute(String name)
c) session.updateAttribute(String name)
d) session.setAttribute(String name)

Answer: a
Explanation: session has various methods for use.

369. Which of these functions is called to display the output of an applet?
a) display()
b) paint()
c) displayApplet()
d) PrintApplet()

Answer: b
Explanation: Whenever the applet requires to redraw its output, it is done by using method paint().

370. Which of these methods can be used to output a string in an applet?
a) display()
b) print()
c) drawString()
d) transient()

Answer: c
Explanation: drawString() method is defined

in Graphics class, it is used to output a string in an applet.

371. Which of these methods is a part of Abstract Window Toolkit (AWT) ?
a) display()
b) paint()
c) drawString()
d) transient()

Answer: b
Explanation: paint() is an abstract method defined in AWT.

372. Which of these modifiers can be used for a variable so that it can be accessed from any thread or parts of a program?
a) transient
b) volatile
c) global
d) No modifier is needed

Answer: b
Explanation: The volatile modifier tells the compiler that the variable modified by volatile can be changed unexpectedly by other part of the program. Specially used in situations involving multithreading.

373. The most commonly used input device is
a) Mouse
b) Keyboard
c) Scanner
d) Printer

Answer: b
Explanation: Keyboard is the most commonly used input device.

374. Which keys allows user to enter frequently used operations in a single key stroke?
a) Function keys
b) Cursor control keys
c) Trackball
d) Control keys

Answer: a
Explanation: Function keys are used to access frequently used areas.

375. _____ are used to measure dial rotations.
a) Potentiometers
b) Volta meter
c) Parameter
d) Only a

Answer: d
Explanation: Potentiometer measures the dial rotations.

376. The device which is used to position the screen cursor is
a) Mouse
b) Joystick
c) Data glove
d) Both a and c

Answer: a
Explanation: A mouse is small hand-held box used to position the screen cursor.

377. _____ is used for detecting mouse motion.
a) Optical sensor
b) Rollers on the bottom of mouse
c) Both a and b
d) Sensor

Answer: c
Explanation: Rollers and optical sensors are used to record the amount and direction of movement.

378. The Cartesian slope-intercept equation for a straight line is
a) y = m.x + b
b) y = b.x + m
c) y = x.x + m
d) y = b + m.m

Answer: a
Explanation: Equation for a straight line is y = m.x + b.

379. For lines with slope magnitude |m|<1, ?x can be_____
a) A set corresponding vertical deflection
b) A set proportional to a small horizontal deflection voltage
c) Only a
d) All of the mentioned

Answer: b
Explanation:?x can be a set proportional to a small horizontal deflection voltage only if slope magnitude |m| < 1.

380. On raster system, lines are plotted with
a) Lines
b) Dots
c) Pixels
d) None of the mentioned

Answer: c
Explanation: Using pixels lines can be plotted.

381. Expansion of line DDA algorithm is
a) Digital difference analyzer
b) Direct differential analyzer
c) Digital differential analyzer
d) Data differential analyzer

Answer: c
Explanation: DDA stands for digital differential analyzer.

382. Which algorithm is a faster method for calculating pixel positions?
a) Bresenham's line algorithm
b) Parallel line algorithm
c) Mid-point algorithm
d) DDA line algorithm

Answer: d
Explanation: The DDA is a faster method for calculating pixel positions.

383. The function which references a single attribute that specifies how a primitive is to be displayed with that attribute setting is called
a) Individual attribute
b) Unbundled attribute
c) Bundled attribute
d) A or B

Answer: d
Explanation: Individual attribute are also known as unbundled attribute.

384. A particular set of attribute values for a primitive on each output device is chosen by specifying appropriate table index is known as?
a) Individual attribute
b) Unbundled attribute

c) Bundled attribute
d) A or B

Answer: c
Explanation: Bundle attributes specifies group of attribute values. And these values can be bundled into the workstation table.

385. A table for which, a primitive defines groups of attribute values to be used when displaying that primitive on a particular output device is called
a) Bundle table
b) Index table
c) Both a and b
d) None of these

Answer: a
Explanation: None.

386. The choice between a bundled attribute or an unbundled attribute is made by
a) Setting switch
b) Setting bundle table
c) Index table
d) Only a

Answer: d
Explanation: By setting switch the user can change their choice between bundled attribute or an unbundled attribute.

387. A translation is applied to an object by
a) Repositioning it along with straight line path
b) Repositioning it along with circular path
c) Only b
d) All of the mentioned

Answer: a
Explanation: A translation is applied to an object by repositioning it along with straight line path from one location to another.

388. We translate a two-dimensional point by adding
a) Translation distances
b) Translation difference
c) X and Y
d) Only a

Answer: d
Explanation: We can translate 2D point by adding translation distances dx and dy.

389. The translation distances (dx, dy) is called as
a) Translation vector
b) Shift vector
c) Both a and b
d) Neither a nor b

Answer: c
Explanation: The translation distances (dx, dy) from its original position is called as translation vector or shift vector.

390. In 2D-translation, a point (x, y) can move to the new position (x', y') by using the equation
a) x'=x+dx and y'=y+dx
b) x'=x+dx and y'=y+dy
c) X'=x+dy and Y'=y+dx
d) X'=x-dx and y'=y-dy

Answer: b
Explanation: By adding translation distance dx and dy to its originsl position (x, y) we can obtain a new position (x', y').

391.The two-dimensional translation equation in the matrix form is
a) P'=P+T
b) P'=P-T
c) P'=P*T
d) P'=p

Answer: a
Explanation: The 2D translation equation is P'=P+T.

392. A two dimensional rotation is applied to an object by
a) Repositioning it along with straight line path
b) Repositioning it along with circular path
c) Only b
d) Any of the mentioned

Answer: c
Explanation: A two dimensional rotation is applied to an object by repositioning it along with circular path.

393. To generate a rotation , we must specify
a) Rotation angle Θ
b) Distances dx and dy
c) Rotation distance
d) All of the mentioned

Answer: a
Explanation: Generate a rotation, we must specify rotation angle Θ of the rotation point or pivot point which the object is to be rotated.

394. Positive values for the rotation angle Θ defines
a) Counterclockwise rotations about the end points
b) Counterclockwise translation about the pivot point
c) Counterclockwise rotations about the pivot point
d) Negative direction

Answer: c
Explanation: A positive value for the rotation angle Θ defines counterclockwise rotations about the pivot point.

395. The rotation axis that is perpendicular to the xy plane and passes through the pivot point is known as
a) Rotation
b) Translation
c) Scaling
d) Shearing

Answer: a
Explanation: The rotation transformation is also described as a rotation about a rotation axis that is perpendicular to the xy plane and passes through the pivot point.

396. The original coordinates of the point in polor coordinates are
a) X'=r cos (Φ +Θ) and Y'=r cos (Φ +Θ)
b) X'=r cos (Φ +Θ) and Y'=r sin (Φ +Θ)
c) X'=r cos (Φ -Θ) and Y'=r cos (Φ -Θ)
d) X'=r cos (Φ +Θ) and Y'=r sin (Φ -Θ)

Answer: b
Explanation: The original coordinates of the point in polor coordinates are X'=r cos (Φ +Θ) and Y'=r sin (Φ +Θ).

397. The two-dimensional rotation equation in the matrix form is

a) P'=P+T
b) P'=R*P
c) P'=P*P
d) P'=R+P

Answer: b
Explanation: The 2D translation equation is P'=R*P.

398. The transformation that is used to alter the size of an object is
a) Scaling
b) Rotation
c) Translation
d) Reflection

Answer: a
Explanation: Scaling is used to alter the size of an object.

399. The two-dimensional scaling equation in the matrix form is
a) P'=P+T
b) P'=S*P
c) P'=P*R
d) P'=R+S

Answer: b
Explanation: The 2d scaling equation is P'=S*P.

400. Scaling of a polygon is done by computing
a) The product of (x, y) of each vertex
b) (x, y) of end points
c) Center coordinates
d) Only a

Answer: d
Explanation: Scaling of a polygon is done by computing the product of (x, y) of each vertex with scaling factor sx and sy to produce the transformation coordinates (Xnew, Ynew).

401. If the scaling factors values sx and sy < 1 then
a) It reduces the size of object
b) It increases the size of object
c) It stunts the shape of an object
d) None

Answer: a
Explanation: If the scaling factors values sx and sy < 1 then it reduces the size of object.

402. If the scaling factors values sx and sy are assigned to the same value then
a) Uniform rotation is produced
b) Uniform scaling is produced
c) Scaling cannot be done
d) Scaling can be done or cannot be done

Answer: b
Explanation: When sx and sy are assigned the same value then uniform scaling is produced that maintains relative object proportions.

403.The matrix representation for translation in homogeneous coordinates is
a) P'=T+P
b) P'=S*P
c) P'=R*P
d) P'=T*P

Answer: d
Explanation: The matrix representation for translation is P'=T*P.

104. The matrix representation for scaling in homogeneous coordinates is

a) P'=S*P
b) P'=R*P
c) P'=dx+dy
d) P'=S*S

Answer: a
Explanation: The matrix representation for scaling is P'=S*P.

405. The matrix representation for rotation in homogeneous coordinates is
a) P'=T+P
b) P'=S*P
c) P'=R*P
d) P'=dx+dy

Answer: c
Explanation: The matrix representation for rotation is P'=R*P.

406. What is the use of homogeneous coordinates and matrix representation?
a) To treat all 3 transformations in a consistent way
b) To scale
c) To rotate
d) To shear the object

Answer: a
Explanation: To treat all 3 transformations in a consistent way, we use homogeneous coordinates and matrix representation.

407. Two successive translations are_____
a) Multiplicative
b) Inverse
c) Subtractive
d) Additive

Answer: d
Explanation: Successive translations are additive.
P'= T(tx1, ty1) .[T(tx2, ty2)] P
= {T(tx1, ty1). T(tx2, ty2)}.P
Or T(tx1, ty1). T(tx2, ty2) = T(tx1+tx2 , ty1 + ty2).

408. Two successive translations are not commutative.
a) True
b) False

Answer: b
Explanation: According to commutative property, the order does not matter. Same as in the case of successive translations. Hence we can say that two successive translations are commutative.

409. General pivot point rotation can be expressed as _____
a) T(zr,yr).R(θ).T(-zr,-yr) = R(xr,yr,θ)
b) T(xr,yr).R(θ).T(-xr,-yr) = R(xr,yr,θ)
c) T(xr,yr).R(θ).T(-xr,-yr) = R(zr,yr,θ)
d) T(xr,yr).R(θ).T(-xr,-yr) = R(xr,yr,Q)

Answer: b
Explanation: Since the first two parameters are in 2D, hence only 'x' and 'y' can be variable along with 'θ'. In other options, there is one more parameter 'z'.

410. Which of the following is NOT correct? (A, B and C are matrices)
a) A.B = B.A
b) A.B.C = (A.B).C = A.(B.C)
c) C(A+B) = C.A + C.B
d) 1 A = A 1

Answer: a
Explanation: Matrix multiplication does not commute. We cannot switch the order of the factors and expect to end up with the same result. Hence, A.B ≠ B.A.

411. Reflection about the line y=0, the axis, is accomplished with the transformation matrix with how many elements as '0'?
a) 8
b) 9
c) 4
d) 6

Answer: d
Explanation: The matrix used for reflection about y=0 is an identity matrix with 6 '0's and two '1's and one element as '-1'.

412. In a rotation, by how much angle is the object rotated?
a) 45 degree
b) 90 degree
c) 180 degree
d) 360 degree

Answer: c
Explanation: Reflection is the mirror image of the original object. It rotates the object 1890 degrees. The left side image is formed into right side and vice versa.

413. Reflection is a special case of rotation.
a) True
b) False

Answer: b
Explanation: Reflection is a special case of rotation of 180° about a line in xy plane passing through the origin.

414. If two pure reflections about a line passing through the origin are applied successively the result is

_____

a) Pure rotation
b) Quarter rotation
c) Half rotation
d) True reflection

Answer: a
Explanation: When we apply reflection one time, it rotates the image by 180 degrees. So, if we repeat it 2 times the total reflection will be of 360 degrees.

415. What is the determinant of the pure reflection matrix?
a) 1
b) 0
c) -1
d) 2

Answer: c
Explanation: The pure reflection matrix is:
[T] = | 1 0 |
| 0 -1|
So, the determinant will be = (1) (-1) – (0) (0) = 0.

416. Shearing is also termed as

_____

a) Selecting
b) Sorting
c) Scaling
d) Skewing

Answer: d
Explanation: In the case of shear only one coordinate changes its coordinates and

other preserves its values, that's why it is also called skewing.

417. Shearing and reflection are types of translation.
a) True
b) False

Answer: b
Explanation: Shearing and reflection are not types of translation. They are types of transformation. Even translation is also a type of transformation.

418. Which of this is compulsory for 2D reflection.
a) Reflection plane.
b) Origin
c) Reflection axis
d) Co-ordinate axis.

Answer: c
Explanation: Reflection axis is the axis with respect to which reflection is done. In 3D, it is reflection plane rather than reflection axis.

419. A view is selected by specifying a sub-area of the _____ picture area.
a) half
b) total
c) full
d) quarter

Answer: b
Explanation: We consider a formal mechanism of view, that is, which part of the picture is to be displayed. That's why we select a view by specifying a sub-area of the total picture area.

420. Co-ordinates are ranging according to the screen resolution.
a) True
b) False

Answer: a
Explanation: When we display a scene, only those objects which have a particular window are displayed. So for that mechanism to work, co-ordinates are made to range themselves according to the screen resolution.

422. Any convenient co-ordinate system or Cartesian co-ordinates which can be used to define the picture is called _____
a) spherical co-ordinates
b) vector co-ordinates
c) viewport co-ordinates
d) world co-ordinates

Answer: d
Explanation: World Coordinate Systems (WCS) are the type of coordinate systems which describe the physical coordinates associated with a data array, such as sky coordinates. It is also used to denote wavelengths of a spectrum and to draw astronomical images.

423. The object space or the space in which the application model is defined is called _____
a) World co-ordinate system
b) Screen co-ordinate system
c) World window
d) Interface window

Answer: a
Explanation: World Coordinate System also

called as WCS is any coordinate systems that describe the physical coordinates associated with a data array. They also used for an astronomical image, or for determining the wavelength scale for a spectrum.

424. What is the name of the space in which the image is displayed?
a) World co-ordinate system
b) Screen co-ordinate system
c) World window
d) Interface window

Answer: b
Explanation: The coordinate system of the screen is a Cartesian coordinate system. The origin (0,0) is at the top left of the screen. Point is denoted by (x,y), where x is x co-ordinate and y is y co-ordinate.

425. What is the rectangle in the world defining the region that is to be displayed?
a) World co-ordinate system
b) Screen co-ordinate system
c) World window
d) Interface window

Answer: c
Explanation: The world window specifies which part of the window needs to be drawn. It also defines which part of the window should be drawn and which part outside the window should not be drawn and should be clipped away.

426. What is the primary use of clipping in computer graphics?
a) adding graphics
b) removing objects and lines
c) zooming
d) copying

Answer: b
Explanation: The primary use of clipping in computer graphics is to remove objects, lines, or line segments that are outside the viewing pane.

427. A polygon can be clipped using clipping operations.
a) True
b) False

Answer: a
Explanation: A polygon can also be clipped by specifying the clipping window. Sutherland Hodgeman polygon clipping algorithm is used for polygon clipping.

428. Which vertex of the polygon is clipped first in polygon clipping?
a) top right
b) bottom right
c) bottom left
d) top left

Answer: d
Explanation: In polygon clipping, first the polygon is clipped against the left edge of the polygon window to get new vertices of the polygon. So, it is the top left which is clipped first.

429. How many methods of text clipping are there?
a) 5
b) 4
c) 3
d) 2

Answer: c
Explanation: There are three methods for text clipping which are –
1) All or none string clipping 2) All or none character clipping 3) Text clipping.

430. A bitmap is collection of _____ that describes an image.
a) bits
b) colors
c) algorithms
d) pixels

Answer: d
Explanation: A bitmap is a collection of pixels that describe an image. It is a type of computer graphics that the computer uses to store and display pictures.

431. We can change the size or resize the bitmap image.
a) True
b) False

Answer: b
Explanation: We can't resize the bitmap image. When the bitmap image is resized, the image pixels get distorted. It is one of the main disadvantages of the bitmap.

432. In line clipping, the portion of line which is _____ of window is cut and the portion that is _____ the window is kept.
a) outside, inside
b) inside, outside
c) exact copy, different
d) different, an exact copy

Answer: a
Explanation: Line clipping follows the same algorithm that is in the case of point clipping. So, in line clipping also, we will cut the portion of the line which is outside of the window and keep only the portion that is inside the window.

433. Cohen-Sutherland clipping is an example of _____
a) polygon clipping
b) text clipping
c) line clipping
d) curve clipping

Answer: c
Explanation: It is a type of algorithm which is used for line clipping or in other words it is line clipping algorithm. Other examples of line clipping algorithms are a Liang-Barsky algorithm and Cyrus-Beck algorithm.

434. The Cohen-Sutherland algorithm divides the region into _____ number of spaces.
a) 8
b) 6
c) 7
d) 9

Answer: d
Explanation: The Cohen-Sutherland algorithm divides a two-dimensional space into 9 regions and then efficiently determines the lines and portions of lines that are visible. The portions are visible in the central region of interest.

435. What is the name of the small integer which holds a bit for the result of every

plane test?
a) setcode
b) outcode
c) incode
d) bitcode

Answer: b
Explanation: A small integer holding a bit for the result of every plane test failed in clipping is termed as outcode. Primitives may be trivially rejected if the bitwise of all its vertices outcodes is non zero.

436. An outcode can have ____ bits for two-dimensional clipping and ____ bits for three-dimensional clipping.
a) 4,6
b) 6,8
c) 2,4
d) 1,3

Answer: a
Explanation: The outcode will have 4 bits for two-dimensional clipping, or 6 bits in the three-dimensional case. The first bit is set to 1 if the point is above the viewport. The bits in the 2D outcode represent: top, bottom, right, left.