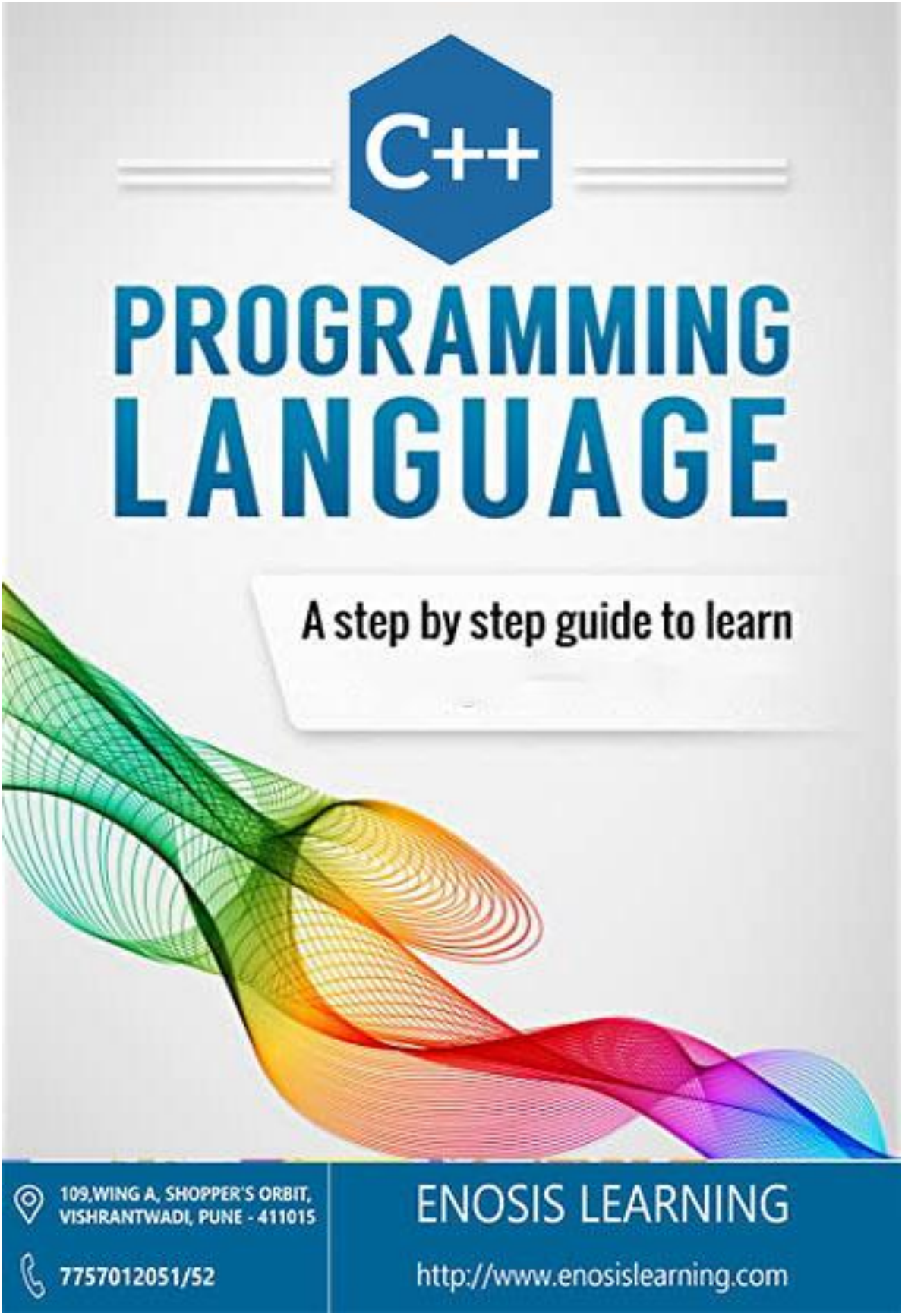

C++
PROGRAMMING
LANGUAGE
A step by step guide to learn
109,WING A, SHOPPER'S ORBIT,
VISHRANTWADI, PUNE - 411015
7757012051/52
ENOSIS LEARNING
http://www.enosislearning.com

# Index

Index

.

# 1. CHAPTER - INTRODUCTION OF C++

## WHAT IS C++

C++ is an object-oriented programming language. It is an extension to C programming C++ is a general purpose, case-sensitive, free-form programming language that supports object-oriented, procedural and generic programming

C++ is a middle-level language, as it encapsulates both high and low level language features.

## C++ HISTORY

**History of C++ language** is interesting to know. Here we are going to discuss brief history of C++ language.

**C++ programming language** was developed in 1980 by Bjarne Stroustrup at bell laboratories of AT&T (American Telephone & Telegraph), located in U.S.A.

**Bjarne Stroustrup** is known as the **founder of C++ language.**


It was develop for adding a feature of **OOP (Object Oriented Programming)**

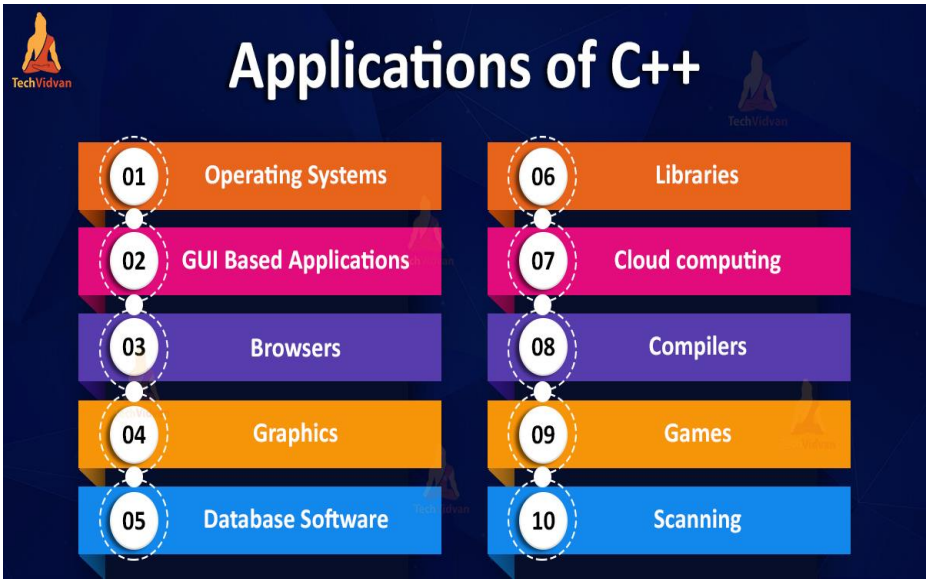
## WHERE WE USE C++ LANGUAGE:

CPP Language is mainly used for

- Design Operating system
- Design Language Compiler
- Design Database
- Utilities
- Application Software

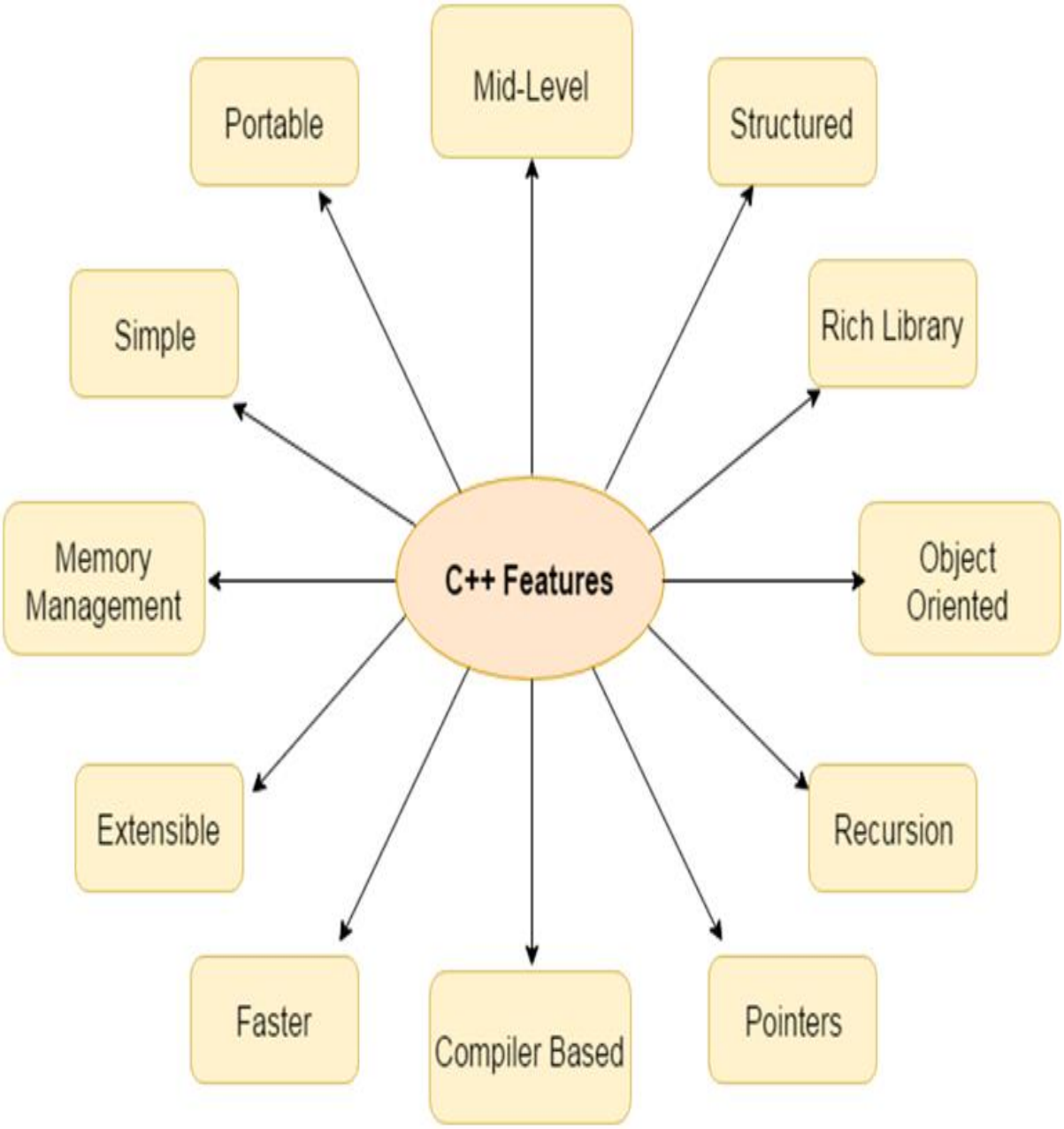
## APPLICATIONS OF C++:

- Mainly C++ Language is used for Develop Desktop application and system software. Some application of C++ language is given below.
- For Develop Graphical related application like computer and mobile games.
- To evaluate any kind of mathematical equation use C++ language.
- C++ Language are also used for design OS. Like window xp.
- Google also use C++ for Indexing
- Internet browsers Firefox are written in C++ programming language c++ are used for design database like MySQL.




## FEATURES OF C++:

C++ is object oriented programming language and it is a very simple and easy language, this language have following features.



# KEYWORDS

| asm | continue | float | new | signed | try |
|---|---|---|---|---|---|
| auto | default | for | operator | sizeof | typedef |
| break | delete | friend | private | static | union |
| case | do | goto | protected | struct | unsigned |
| catch | double | if | public | switch | virtual |
| char | Else | Inline | register | template | void |
| class | enum | int | return | this | volatile |
| const | extern | long | short | throw | while |

C++ Reserved Keywrds

Keywords is also called as reserved words.

# 2. CHAPTER – GETTING STARTED

## DATATYPES IN C++:

They are used to define type of variables and contents used. Data types define the way you use storage in the programs you write. Data types can be built in or abstract.

**Built in Data Types:**

These are the data types which are predefined and are wired directly into the compiler. eg: int, char etc.

**User defined or Abstract data types:**

These are the type, that user creates as a class. In C++ these are classes where as in C it was implemented by structures.

**Basic Built in data types:**

**char for character storage ( 1 byte )**

**int for integral number ( 2 bytes )**

**float single precision floating point ( 4 bytes )**

**double double precision floating point numbers ( 8 bytes )**

Example :

```
char a = 'A';                 // character type
int a = 1;                    // integer type
float a = 3.14159;            // floating point type
double a = 6e-4;              // double type (e is for exponential)
```

## WHAT ARE VARIABLES?

Variable are used in C++, where we need storage for any value, which will change in program. Variable can be declared in multiple ways each with different memory requirements and functioning. Variable is the name of memory location allocated by the compiler depending upon the datatype of the variable.

### Declaration and Initialization:

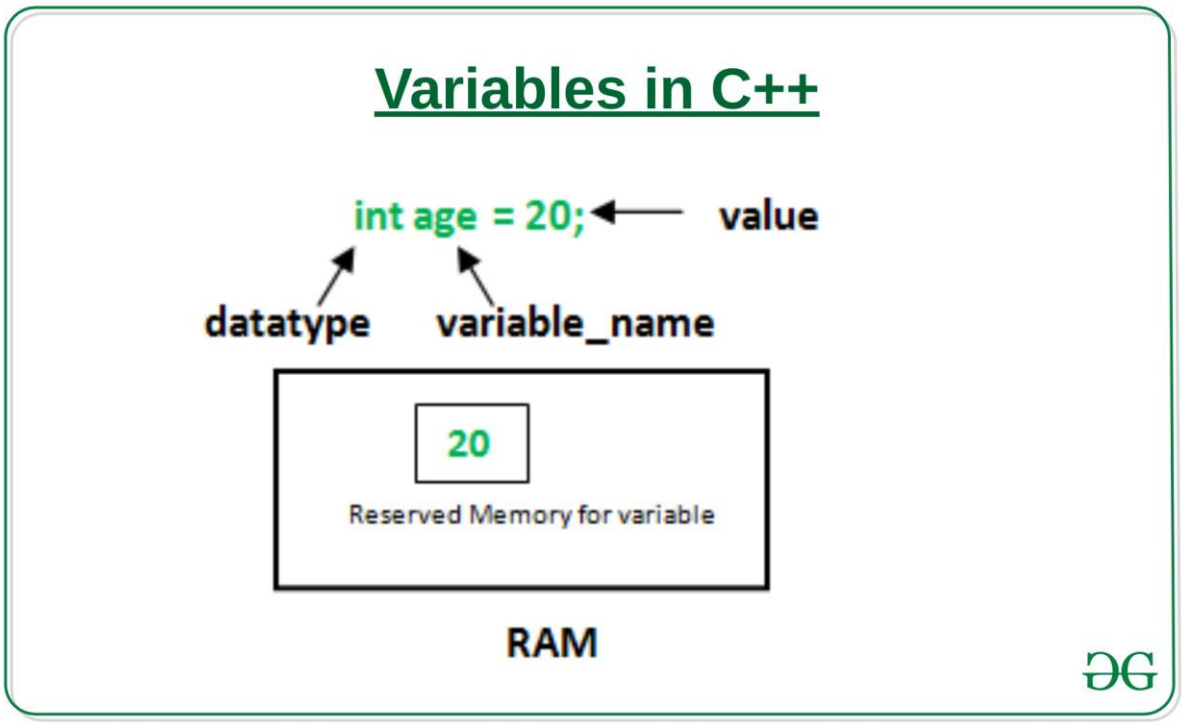
Variable must be declared before they are used. Usually it is preferred to declare them at the starting of the program, but in C++ they can be declared in the middle of program too, but must be done before using them.

Example :

```
int i;          // declared but not initialised

char c;

int i, j, k;    // Multiple declaration
```

Initialization means assigning value to an already declared variable,

```
int i;      // declaration

i = 10;     // initialization
```

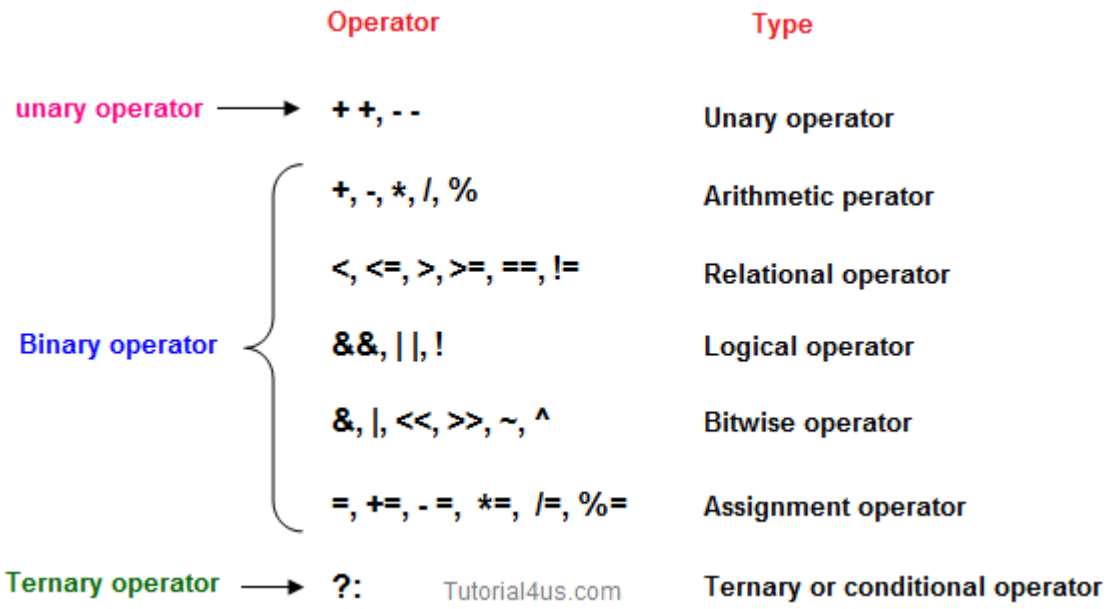


## OPERATORS:

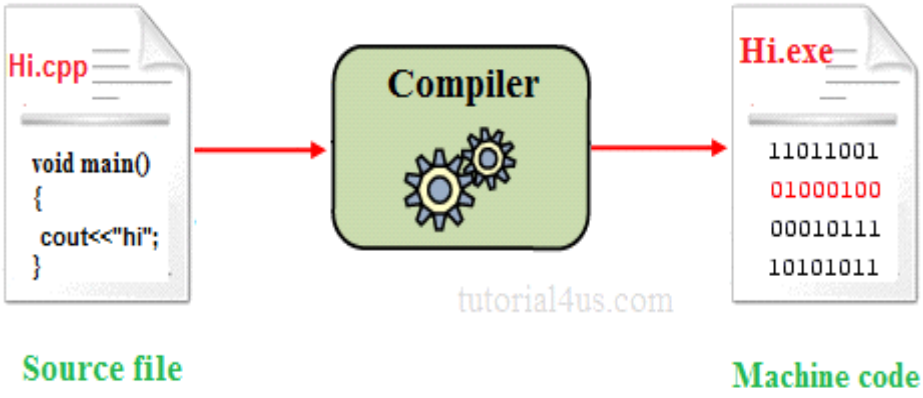
**Operator is symbol that perform some Mathematical Operations.**

Types of operators

1. Assignment Operator
2. Mathematical Operators/Arithmetic Operators
3. Relational Operators
4. Logical Operators
5. Bitwise Operators
6. Comma Operator



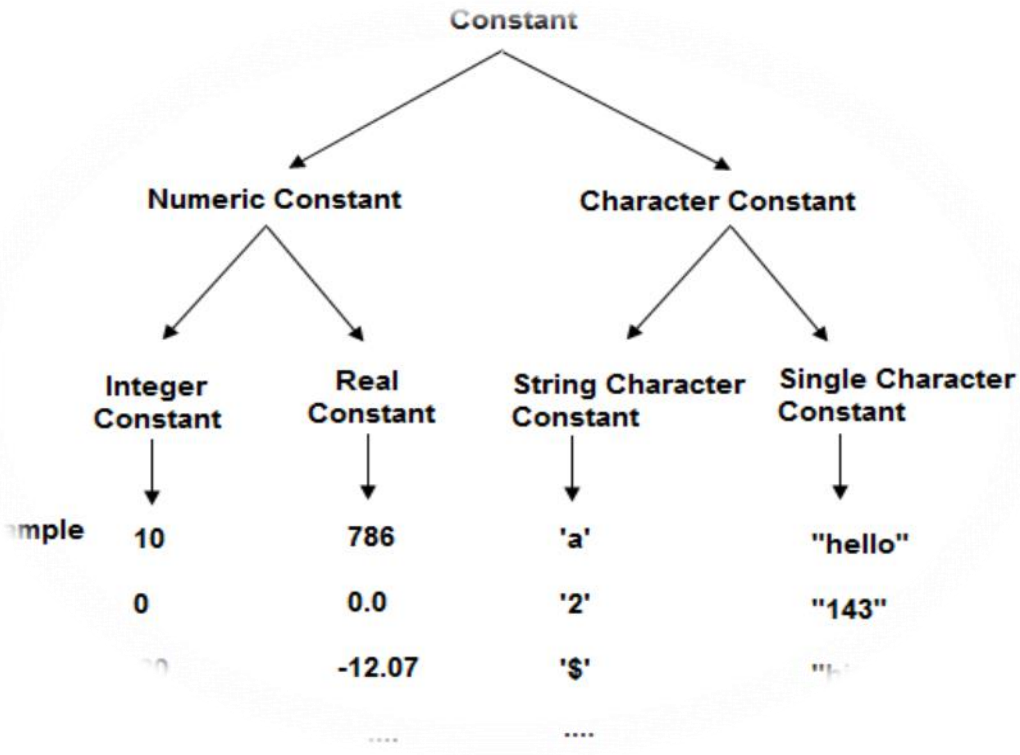
## COMPILER IN C++:




A compiler is system software wnıcn converts programming language code into binary format in single steps. In other words Compiler is a system software which can take input from other any programming language and convert it into lower level machine dependent language.

## CONSTANT IN C++:

It is an identifier whose value can not be changed at the execution time of program. In general constant can be used to represent as fixed values in a C++ program. Constants are classified into following types.



## SCOPE OF VARIABLE IN C++:

In C++ language, a variable can be either of global or local scope.

### Global variable:

Global variables are defined outside of all the functions, generally on top of the program. The global variables will hold their value throughout the life-time of your program.

### Local variable:

A local variable is declared within the body of a function or a block. Local variable only use within the function or block where it is declare.

Example of Global and Local variable

```
#include<iostream.h>
#include<conio.h>
int a;      // global variable
void main()
{
int b;       // local variable
a=10, b=20;
cout<<"Value of a: "<<a;
cout<<"Value of b: "<<b;
getch();
}
```

Output

```
Value of a: 10
Value of b: 20
```

# 3. CHAPTER-STRUCTURE OF C++ PROGRAM:

```
#include<headerfilename.h>    --> include
section
class class_name
{
data members;
user defined method;
{
.....
.....
}
};
returntype main()    --> main method
{
.....
.....
}
```

- #include is a specific preprocessor command that effectively copies and pastes the entire text of the file, specified between the angle brackets, into the source code.
- The file <iostream>, which is a standard file that should come with the C++ compiler, is short for **input-output streams**. This command contains code for displaying and getting an input from the user.
- namespace is a prefix that is applied to all the names in a certain set. iostream file defines two *names* used in this program cout and cin endl.

## COUT AND CIN IN C++:

**cout<< :** cout is used for print message on screen

Example

```
#include<iostream.h>
#include<conio.h>
void main()
{
  int num;
  clrscr();
  cout<<"Hello word !";    // print message on screen
  getch();
}
```

Output:

Hello word !

## Cin:

cin>> : cin is used for get or read value form keyboard.

Example

```
#include<iostream.h>
#include<conio.h>
void main()
{
  int num;
  clrscr();
  cout<<"Enter any number:";
  cin>>num;    // accept one value form keyboard
  cout<<"Number is: "<<num;
  getch();
}
```

# Assignment No: 1

**Question 1** Write a program to print HELLO WORLD on screen.

**Question 2** Write a program to display the following output using a single cout statement.

| Subject | Marks |
|---|---|
| Mathematics | 90 |
| Computer | 77 |
| Chemistry | 69 |

**Question 3** Write a program which accept two numbers and print their sum.

**Question 4** Write a program which accept temperature in Fahrenheit and print it in centigrade.

**Question 5** Write a program which accept principle, rate and time from user and print the simple interest.

**Question 6** Write a program which accepts a character and display its ASCII value.

**Question 7** Write a program to swap the values of two variables.

**Question 8** Write a program to calculate area of circle.

# 4. CHAPTER – DECISION & LOOP STATEMENT

## Decision making statement:

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,

1. if

2. if.....else

3. nested if.

### 1. 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' it will be executed, otherwise 'statement-inside' is skipped and only 'statement-outside' is executed.

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.

## 2. if...else statement:

The general form of a simple if...else statement is,

```
if( expression )
{
  statement-block1;
}
else
{
  statement-block2;
}
```

If the 'expression' is true, the 'statement-block1' is executed, else 'statement-block1' is skipped and 'statement-block2' is executed.

Example :

```
void main( )
{
  int x,y;
  x=15;
  y=18;
  if (x > y )
  {
    cout << "x is greater than y";
  }
  else
  {
    cout << "y is greater than x";}}
```

# Assignment no 2

**Question 1** Any integer is input by the user. Write a program to find out whether it is an odd number or even number.

**Question 2** Find the absolute value of a number entered by the user.

**Question 3** Write a program to calculate the total expenses. Quantity and price per item are input by the user and discount of 10% is offered if the expense is more than 5000.

**Question 4** If the ages of Ram, Sulabh and Ajay are input by the user, write a program to determine the youngest of the three.

**Question 5** Write a program to check whether a triangle is valid or not, when the three angles of the triangle are entered by the user. A triangle is valid if the sum of all the three angles is equal to 180 degrees.

**Question 6** Any year is input by the user. Write a program to determine whether the year is a leap year or not.

# 5. CHAPTER-LOOPS

A sequence of statement is executed until a specified condition is true. This sequence of statement to be executed is kept inside the curly braces { } known as loop body. After every execution of loop body, condition is checked, and if it is found to be true the loop body is executed again. When condition check comes out to be false, the loop body will not be executed.

There are 3 type of loops in C++ language

1. while loop
2. for loop
3. do-while loop

## 1. While loop:

```
variable initialization ;
while (condition)
{
  statements ;
  variable increment or decrement ; }
```

```
//program for demonstrate while loop.
#include <iostream>
using namespace std; // So we can see cout and endl
int main()
{
   int x = 0;    // Don't forget to declare variables

   while ( x < 10 )
{ // While x is less than 10
      cout<< x <<endl;
      x++;                    // Update x so the condition can be met eventually
}
```

```
}
```

## 2. for loop:

For loop is used to execute a set of statement repeatedly until a particular condition is satisfied. We can say it an open ended loop. General format is,

```
for(initialization; condition ; increment/decrement)
{
   statement-block;
}
```

In for loop we have exactly two semicolons, one after initialization and second after condition. In this loop we can have more than one initialization or increment/decrement, separated using comma operator. for loop can have only one condition.

```
#include <iostream>
using namespace std;
int main()
{
       int i, n, factorial = 1;

       cout << "Enter a positive integer: ";
       cin >> n;
       for (i = 1; i <= n; ++i)
 {
   factorial = factorial * i;
       }
       cout<< "Factorial of "<<n<<" = "<<factorial;
       return 0;
}
```

Output: Enter a positive integer: 5

Factorial of 5 = 120

## 3. do while:

In some situations it is necessary to execute body of the loop before testing the condition. Such situations can be handled with the help of do-while loop. do statement evaluates the body of the loop first and at the end, the condition is checked using while statement. General format of do-while loop is,

```
Do
{
  ....
  .....
}
while(condition);
```

```
// C++ program to add numbers until user enters 0
#include <iostream>
using namespace std;
int main()
{
	float number, sum = 0.0;
	do {
		cout<<"Enter a number: ";
		cin>>number;
		sum += number;
	}
	while(number != 0.0);

	cout<<"Total sum = "<<sum;
	return 0;
}
```

Loop Control Statements:

1.break

2.continue

3.goto

**1.break:**

The break statement terminates a loop (for, while and do..while loop) and a switch statement immediately when it appears.

Syntax:

break;

```
#include <iostream>
using namespace std;
int main () {
   // Local variable declaration:
   int a = 10;
   // do loop execution
   do {
      cout << "value of a: " << a << endl;
      a = a + 1;
      if( a > 15) {
         // terminate the loop
         break;
      }
   }while( a < 20 );
   return 0;
}
```

output:

value of a: 10

value of a: 11

value of a: 12

value of a: 13

value of a: 14

value of a: 15

## 2. Continue:

Skip that particular Element.

```
//C++ program to display integer from 1 to 10 except 6 and 9.
#include <iostream>
using namespace std;
int main()
{
	for (int i = 1; i <= 10; ++i)
	{
		if ( i == 6 || i == 9)
		{
			continue;
		}
		cout << i << "\t";
	}
	return 0;
}
```

Output

1 2 3 4 5 7 8 10

## 3. goto:

A goto statement provides an unconditional jump from the goto to a labeled statement in the same function.

The syntax of a goto statement in C++ is:

```
goto label;
..
.
label: statement;
```

## Switch Statement in cpp:

A switch statement allows a variable to be tested for equality against a list of values. Each value is called a case, and the variable being switched on is checked for each case.

**The syntax for a switch statement in C++ is as follows:**

```
switch(expression)
{
    case constant-expression  :
        statement(s);
        break;
    case constant-expression  :
        statement(s);
        break;
    default :
        statement(s);
}
```

```
#include <iostream>
using namespace std;
int main () {
          char grade = 'D';
     switch(grade) {
          case 'A' :
               cout << "Excellent!" << endl;
```

```
            break;
        case 'B' :
        case 'C' :
            cout << "Well done" << endl;
            break;
        case 'D' :
            cout << "You passed" << endl;
            break;
        case 'F' :
            cout << "Better try again" << endl;
            break;
        default :
           cout << "Invalid grade" << endl;
    }

    cout << "Your grade is " << grade << endl;

    return 0;
}
```

**Output:**

**You passed**

**Your grade is D**

# Assignment no 3

**Question 1** Write a program to print number from 1 to 10.

**Question 2** Write a program to calculate the sum of first 10 natural number.

**Question 3** Write a program to find the factorial value of any number entered through the keyboard.

**Question 4** Two numbers are entered through the keyboard. Write a program to find the value of one number raised to the power of another.

**Question 5** Write a program to reveres any given integer number.

**Question 6** Write a program to sum of digits of given integer number.

**Question 7** Write a program to check given number is prime or not.

**Question 8** Write a program to calculate HCF of Two given number.

**Question 9** Write a program to enter the numbers till the user wants and at the end it should display the count of positive, negative and zeros entered.

# CHAPTER 6: FUNCTIONS

## Function in C++:

A function is a group of statements that together perform a specific task. Every C++ program has at least one function, which is main().

### Why use function?

Function are used for divide a large code into module, due to this we can easily debug and maintain the code. For example if we write a calculator programs at that time we can write every logic in a separate function (For addition sum(), for subtraction sub()). Any function can be called many times.

Advantage of Function

- Code Re-usability
- Develop an application in module format.
- Easily to debug the program.
- Code optimization: No need to write lot of code

## inline Function in C++:

Inline Function is powerful concept in C++ programming language. If a function is inline, the compiler places a copy of the code of that function at each point where the function is called at compile time.

To make any function inline function just preceded that function with inline keyword.

### Why use Inline function:

Whenever we call any function many time then, it take a lot of extra time in execution of series of instructions such as saving the register, pushing arguments, returning to calling function. For solve this problem in C++ introduce inline function.

### Advantage of Inline Function

The main advantage of inline function is it make the program faster.

Syntax:

```
inline function_name()
{
    //function body
}
```

```
Example
#include<iostream.h>
#include<conio.h>
inline void show()
{
   cout<<"Hello world";
}
void main()
{
  show();    // Call it like a normal function
  getch();
}
```

Output:

Hello word

## Friend Function in C++:

In C++ a Friend Function that is a "friend" of a given class is allowed access to private and protected data in that class.

A function can be made a friend function using keyword friend. Any friend function is preceded with friend keyword. The declaration of friend function should be made inside the body of class (can be anywhere inside class either in private or public section) starting with keyword friend.

### Why use friend function?

You do not access private or protected data member of any class, to access private and protected data member of any class you need a friend function.

Syntax:

```
class class_name
{
  ......
  friend return type function name(arguments);
}
```

## Example Friend function:

```
Example
#include<iostream.h>
#include<conio.h>
class employee
{
private:
        friend void sal();
};
void sal()
{
int salary=4000;
cout<<"Salary: "<<salary;
}
void main()
{
employee e;
sal();
getch();
}
```

Output:

Salary: 4000

## Function Overloading in C++:

Whenever same method name is exiting multiple times in the same class with different number of parameter or different order of parameters or different types of parameters is known as method overloading.

Syntax:

class class_Name

{

Returntype method()

{

.....

.....

}

Returntype method(datatype1 variable1)

{

.....

.....

}

Returntype method(datatype1 variable1, datatype2 variable2)

{

.....

.....

}

};

Example

```
#include<iostream.h>
#include<conio.h>
class Addition
{
public:
```

```
void sum(int a, int b)
{
cout<<a+b;
}
void sum(int a, int b, int c)
{
cout<<a+b+c;
}
};
void main()
{
clrscr();
Addition obj;
obj.sum(10, 20);
cout<<endl;
obj.sum(10, 20, 30);
}
```

Output

30

60.

# CHAPTER 7: OOPS CONCEPTS.

## Oops Concept in C++:

The main purpose of C++ programming was to add object orientation to the C programming language, which is in itself one of the most powerful programming languages. If any programming language follow below oops concept then that language called object oriented programming language.

- ✓ Object
- ✓ Class
- ✓ Encapsulation
- ✓ Abstraction
- ✓ Inheritance
- ✓ Polymorphism

### Object:

Object is the physical as well as logical entity where as class is the only logical entity.

Class:

Class is a blue print which is containing only list of variables and method and no memory is allocated for them. A class is a group of objects that has common properties.

Encapsulation:

Encapsulation is a process of wrapping of data and methods in a single unit is called encapsulation. Encapsulation is achieved in C++ language by class concept. The main advantage of using of encapsulation is to secure the data from other methods, when we make a data private then these data only use within the class, but these data not accessible outside the class.

Abstraction:

Abstraction is the concept of exposing only the required essential characteristics and behavior with respect to a context.

Hiding of data is known as data abstraction. In object oriented programming

language this is implemented automatically while writing the code in the form of class and object.

Inheritance:

The process of obtaining the data members and methods from one class to another class is known as inheritance. It is one of the fundamental features of object-oriented programming.

Polymorphism:

The process of representing one Form in multiple forms is known as Polymorphism. Here one form represent original form or original method always resides in base class and multiple forms represents overridden method which resides in derived classes.

Access Specifiers in C++:

Access specifiers in C++ define how the members of the class can be accessed. C++ has 3 new keywords introduced, namely.

public

private

protected

| Specifiers | Within Same Class | In Derived Class | Outside the Class |
|---|---|---|---|
| Private | Yes | No | No |
| Protected | Yes | Yes | No |
| Public | Yes | Yes | Yes |

Sitesbay.com

# Class:

Class is a blue print which is containing only list of variables and method and no memory is allocated for them. A class is a group of objects that has common properties.

A class in C++ contains, following properties;

Data Member

Method

Constructor

Object: Object is a instance of class, object has state and behaviors.

An Object in C++ has three characteristics:

State

Behavior

Identity

Real Life Example:

Vehicle class:

Car, bike, truck these all are belongs to vehicle class. These Objects have also different different states and behaviors. For Example car has state - color, name, model, speed, Mileage. as we;; as behaviors - distance travel

Syntax to declare a Class

Syntax:

```
class Class_Name
{
    data member;
    method;
}
```

```
//Simple Example of Class:
#include<iostream.h>
#include<conio.h>
class Employee
{
public:
int salary    // data member
void sal()
{
    cout<<"Enter salary: ";
    cin>>salary;
    cout<<"Salary: "<<salary;
}
};
  void main()
  {
    clrscr();
    Employee e; //creating an object of Employee
    e.sal();
    getch();
  }
```

Output

Enter salary: 4500

Salary: 4500

## Constructor:

**Constructor** is a special member function which will be called implicitly (automatically) whenever an object of class is created. In other words, it is a member function which initializes a class which is called automatically whenever a new instance of a class is created.

## FEATURES OF CONSTRUCTOR

- The same name as the class itself.

- no return type.

```
Syntax:
classname()
{
....
}
```

## Types of Constructor:

1) default constructor.
2) Parameterized constructor.
3) Copy constructor.

### DEFAULT CONSTRUCTOR

Default constructor is the constructor which doesn't take any argument. It has no parameter.

```
Class cube
{
   Int side;
   Public:
       Cube()
      {
            Side=10;
      }
};
Int main()
{
Cube c;
Cout<<c.side;
}
```

```
Output : 10
```

In this case, as soon as the object is created the constructor is called which initializes its data members.

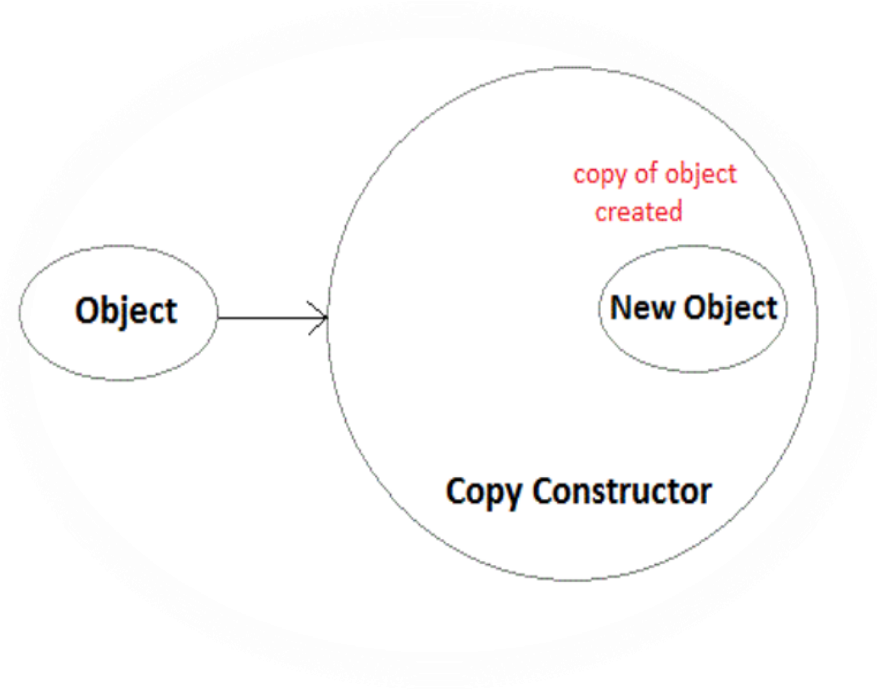
## PARAMETERIZED CONSTRUCTOR:

These are the constructors with parameter. Using this Constructor you can provide different values to data members of different objects, by passing the appropriate values as argument.

```
Class cube
{
   Public:
       Int side;
    Cube()
{
   Side=x;
}
};
Int main()
{
    Cube c1(10);
    Cube c2(30);
    Cube c2(20);
Cout<<c1.side;
Cout<<c2.side;
Cout<<c3.side;
}
```

## COPY CONSTRUCTOR:-

These are special type of Constructors which takes an object as argument, and is used to copy values of data members of one object into other object




```
#include<iostream>
#include<conio.h>
using namespace std;
class Example {
    // Variable Declaration
    int a, b;
public:
    //Constructor with Argument
    Example(int x, int y) {
        // Assign Values In Constructor
        a = x;
        b = y;
        cout << "\nIm Constructor";
    }

    void Display() {
        cout << "\nValues :" << a << "\t" << b;
    }
```

```
};
int main() {
    Example Object(10, 20);
    //Copy Constructor
    Example Object2 = Object;
    // Constructor invoked.
    Object.Display();
    Object2.Display();
    // Wait For Output Screen
    getch();
    return 0;
}
```

# 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 (~).

```
#include <iostream>
using namespace std;
 class Employee
 {
   public:
       Employee()
       {
          cout<<"Constructor Invoked"<<endl;
       }
       ~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
```

**Question 1** Define a class student with the following specification

**Private members** of class student
admno integer
sname 20 character
eng. math, science float
total float
ctotal() a function to calculate eng + math + science with float return type.
**Public member** function of class student
Takedata() Function to accept values for admno, sname, eng, science and invoke ctotal() to calculate total.
Showdata() Function to display all the data members on the screen.

**Question 2** Define a class batsman with the following specifications:

| | |
|---|---|
| **Private members:** | |
| bcode | 4 digits code number |
| bname | 20 characters |
| innings, notout, runs | integer type |
| batavg | it is calculated according to the formula batavg =runs/(innings-notout) |
| calcavg() | Function to compute batavg |
| **Public members:** | |
| readdata() | Function to accept value from bcode, name, innings, notout and invoke the function calcavg() |
| displaydata() | Function to display the data members on the screen. |

**Question 3** Define a class TEST in C++ with following description:
**Private Members**
TestCode of type integer
Description of type string
NoCandidate of type integer
CenterReqd (number of centers required) of type integer
A member function CALCNTR() to calculate and return the number of centers as
(NoCandidates/100+1)
**Public Members**
- A function SCHEDULE() to allow user to enter values for TestCode, Description, NoCandidate & call function CALCNTR() to calculate the number of Centres
- A function DISPTEST() to allow user to view the content of all the data members

Question 4:what is constructor?Explain its types?

1.write a program to declare a class employee having data memebers as name and basic-salary . accept the data for2 emplyees.

2.write a program to declare a class book having data members as title and author accept the for five books and display this accepted data.

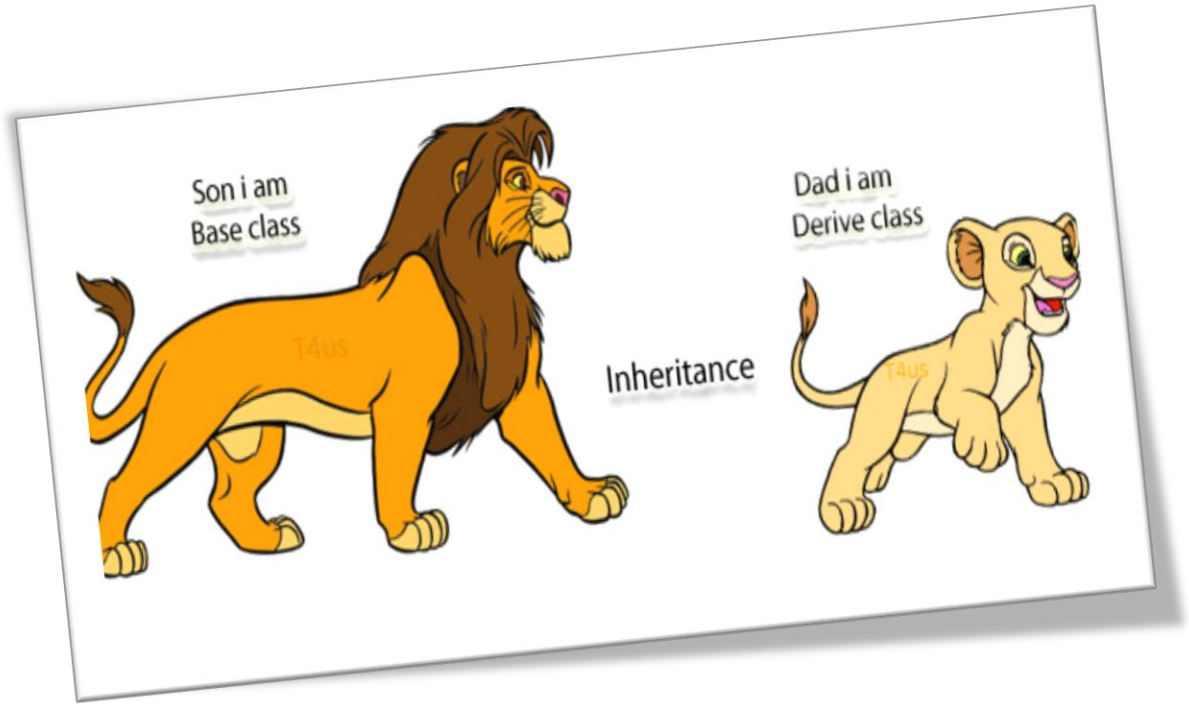
# CHAPTER 8: INHERITANCE.

## Inheritance in C++:

The process of obtaining the data members and methods from one class to another class is known as inheritance.

Real Life Example:

The best example in inheritance is parent child.




Syntax:

```
class subclass_name : superclass_name
{
    // data members
   // methods
}
```

### Advantage of inheritance

- Application development time is less.
- Application takes less memory.
- Application execution time is less.
- Application performance is enhancing (improved).
- Redundancy (repetition) of the code is reduced or minimized so that we get consistence results and less storage cost.

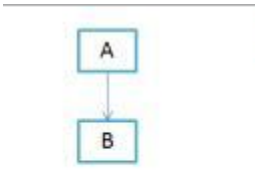
### Types of Inheritance:

Based on number of ways inheriting the feature of base class into derived class it have five types they are:

1. Single inheritance
2. Multiple inheritance
3. Hierarchical inheritance
4. Multiple inheritance
5. Hybrid inheritance.

## 1) SINGLE INHERITANCE

**Single inheritance** is damn easy to understand. When a class extends another one class only then we call it a single inheritance. The below flow diagram shows that class B extends only one class which is A. Here A is a **parent class** of B and B would be a **child class** of A.

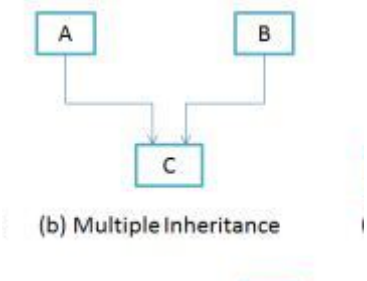



(a) Single Inheritance

```
Class A
{
   public void methodA()
   {
     Cout<<”base class method”;
   }
};

Class B:public A
{
   public void methodB()
   {
     Cout<<"Child class method";
   }
};
   void main()
   {
     B obj;
     obj.methodA(); //calling super class method
     obj.methodB(); //calling local method
  }
}
```

## 2) MULTIPLE INHERITANCE

**“Multiple Inheritance”** refers to the concept of one class extending (Or inherits) more than one base class. The inheritance we learnt earlier had the concept of one base class or parent. In C++ programming, a class can be derived from more than one parents.



(b) Multiple Inheritance

```
#include<iostream>
using namespace std;
 class A
{
public:
   A()
```

```
 { cout << "A's constructor called" << endl;
}
};
 class B
{
public:
  B()  {
cout << "B's constructor called" << endl;
 }
};

class C: public B, public A  // Note the order
{
public:
  C()  { cout << "C's constructor called" << endl; }
};

int main()
{
    C c;
    return 0;
}
```

Output:

```
B's constructor called

A's constructor called

C's constructor called
```

## 3) MULTILEVEL INHERITANCE:

In C++ programming, not only you can derive a class from the base class but you can also derive a class from the derived class. This form of inheritance is known as multilevel inheritance.

```
class A

{

... .. ...

};

class B: public A
```

```
{

... .. ...

};

class C: public B

{

... ... ...

};
```

## EXAMPLE:

```
#include <iostream>
using namespace std;

class A
{
    public:
      void display()
      {
          cout<<"Base class content.";
      }
};

class B : public A
{

};

class C : public B
```

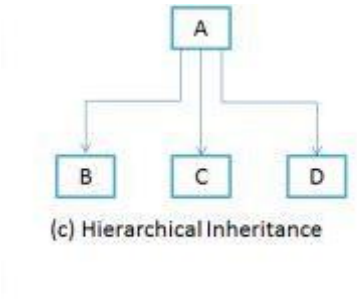
```
{


};


int main()
{
    C obj;
    obj.display();
    return 0;
}
```

## 4) HIERARCHICAL INHERITANCE:

If more than one class is inherited from the base class, it's known as hierarchical inheritance. In hierarchical inheritance, all features that are common in child classes are included in the base class.

It is just like tree.

For example: Physics, Chemistry, Biology are derived from Science class.



(c) Hierarchical Inheritance

## 5) HYBRID INHERITANCE

In simple terms you can say that Hybrid inheritance is a combination of **Single** and **Multiple inheritance.and the combination of two inheritance.**

**Example on Hybrid inheritance:**

```
#include<iostream.h>
#include<conio.h>
class arithmetic
{
protected:
int num1, num2;
public:
void getdata()
{
cout<<"For Addition:";
cout<<"\nEnter the first number: ";
cin>>num1;
cout<<"\nEnter the second number: ";
cin>>num2;
}
};
class plus:public arithmetic
{
protected:
int sum;
public:
void add()
{
sum=num1+num2;
}
};
class minus
{
protected:
int n1,n2,diff;
public:
void sub()
{
cout<<"\nFor Subtraction:";
cout<<"\nEnter the first number: ";
cin>>n1;
cout<<"\nEnter the second number: ";
cin>>n2;
diff=n1-n2;
}
};
class result:public plus, public minus
{
public:
void display()
```

```
{
cout<<"\nSum of "<<num1<<" and "<<num2<<"= "<<sum;
cout<<"\nDifference of "<<n1<<" and "<<n2<<"= "<<diff;
}
};
void main()
{
clrscr();
result z;
z.getdata();
z.add();
z.sub();
z.display();
getch();
}
```

# Assignments no 5

1. What is inheritance?and which are the types of it?

2. What is multiple inheritace Explain with example.

3. What is hybrid inheritance.explain with example.

4. A base class contain the data of employee name,empno. The derived class contain the basic salary. The derived class has been declared as array of objects.(single inheritance).

5. A base class is student.class test is intermediate base class and result is childe class. The class result inherits the details of marks obtained in the test and the roll noof students from base class through multilevel inheritance.

6. Consider an example to print bio-data of a diploma student having personal and academic information(multiple inheritance).

# 9. CHAPTER – POLYMARPHISM

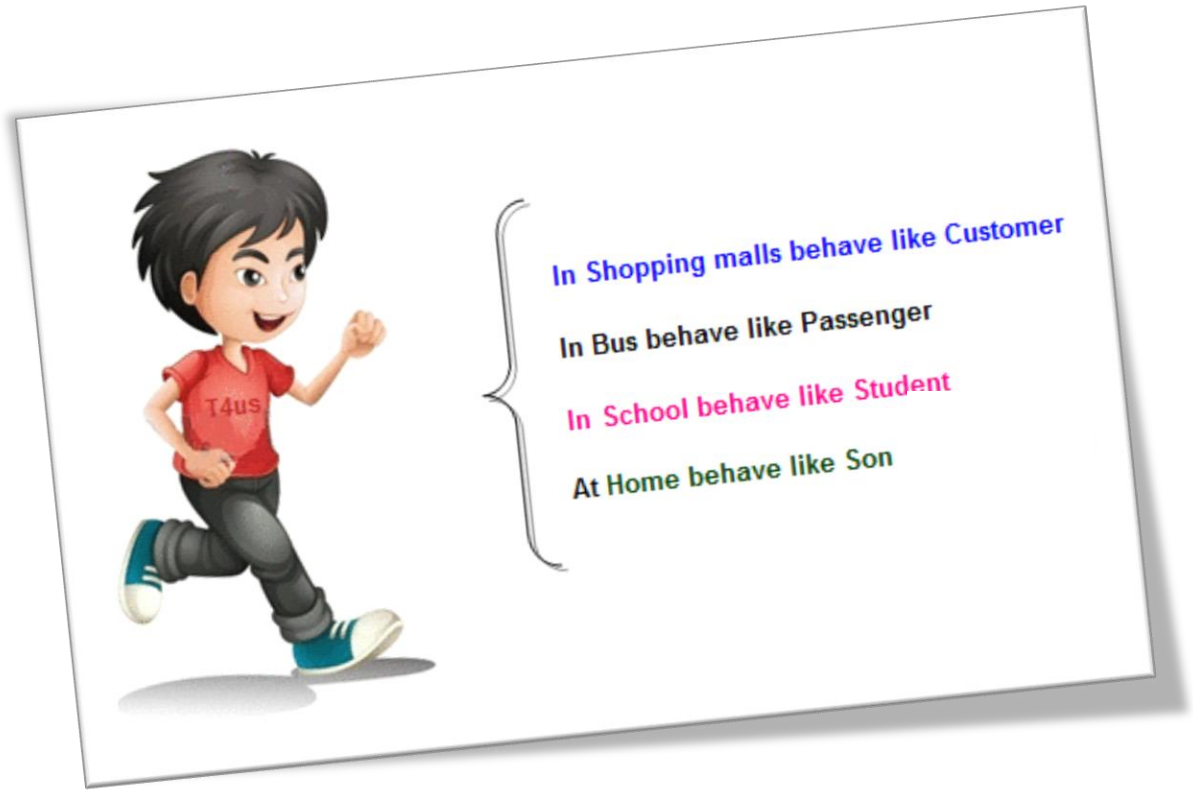
## POLYMORPHISM IN C++:

The process of representing one Form in multiple forms is known as Polymorphism. Here one form represent original form or original method always resides in base class and multiple forms represents overridden method which resides in derived classes.

Polymorphism is derived from 2 greek words: poly and morphs. The word "poly" means many and morphs means forms. So polymorphism means many forms.

### REAL LIFE EXAMPLE OF POLYMORPHISM IN C++:

Suppose if you are in class room that time you behave like a student, when you are in market at that time you behave like a customer, when you at your home at that time you behave like a son or daughter, Here one person have different-different behaviors.




## TYPE OF POLYMORPHISM

1. Compile time polymorphism
2. Run time polymorphism

## COMPILE TIME POLYMORPHISM

In C++ programming you can achieve compile time polymorphism in two way, which is given below:

Method overloading

Method overriding

## METHOD OVERLOADING IN C++:

Whenever same method name is exiting multiple times in the same class with different number of parameter or different order of parameters or different types of parameters is known as method overloading. In below example method "sum()" is present in Addition class with same name but with different signature or arguments.

```
//Method Overloading
#include<iostream.h>
#include<conio.h>

class Addition
{
public:
void sum(int a, int b)
{
cout<<a+b;
}
void sum(int a, int b, int c)
{
cout<<a+b+c;
}
};
void main()
{
clrscr();
Addition obj;
obj.sum(10, 20);
cout<<endl;
obj.sum(10, 20, 30);
}
```

Output:

30

60

## METHOD OVERRIDING IN C++:

Define any method in both base class and derived class with same name, same parameters or signature, this concept is known as method overriding. In below example same method "show()" is present in both base and derived class with same name and signature.

```
Example of Method Overriding in C++
#include<iostream.h>
#include<conio.h>
class Base
{
  public:
  void show()
  {
    cout<<"Base class";
  }
};
class Derived:public Base
{
  public:
  void show()
  {
    cout<<"Derived Class";
  }
}

int main()
{
  Base b;             //Base class object
  Derived d;    //Derived class object
  b.show();       //Early Binding Ocuurs
  d.show();
getch();
}
```

Output:

Base class

Derived Class

## Run time polymorphism:

In C++ Run time polymorphism can be achieve by using virtual function.

## VIRTUAL FUNCTION IN C++:

A virtual function is a member function of class that is declared within a base class and re-defined in derived class.

When you want to use same function name in both the base and derived class, then the function in base class is declared as virtual by using the virtual keyword and again re-defined this function in derived class without using virtual keyword.

Syntax

virtual return type function name()

{

.....

.....

}

```
Virtual Function Example
#include<iostream.h>
#include<conio.h>

class A
{
public:
  virtual void show()
  {
    cout<<"Hello base class";
  }
};
```

```
class B : public A
{
  public:
    void show()
    {
      cout<<"Hello derive class";
  }
};

void main()
{
clrsct();
A aobj;
B bobj;
A *bptr;
bptr=&aobj;
bptr->show();    // call base class function

bptr=&bobj;
bptr->show();    // call derive class function
getch();
}
```

Output:

Hello base class

Hello derive class

## ENCAPSULATION IN C++

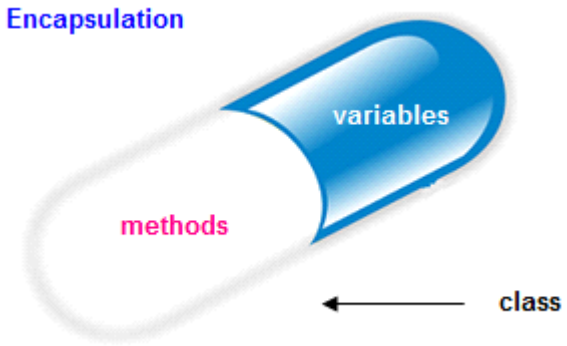
Encapsulation is a process of wrapping of data and methods in a single unit. It is achieved in C++ language by class concept.

Combining of state and behavior in a single container is known as encapsulation. In C++ language encapsulation can be achieve using class keyword, state represents declaration of variables on attributes and behavior represents operations in terms of method.

## ADVANTAGE OF ENCAPSULATION:

The main advantage of using of encapsulation is to secure the data from other methods, when we make a data private then these data only use within the class, but these data not accessible outside the class.

Real Life Example of Encapsulation in C++




The common example of encapsulation is Capsule. In capsule all medicine are encapsulated in side capsule.

Benefits of encapsulation

Provides abstraction between an object and its clients.

Protects an object from unwanted access by clients.

Example: A bank application forbids a client to change an Account's balance.

```
Example of Encapsulation in C++
#include<iostream.h>
#include<conio.h>

class sum
{
private: int a,b,c;
public:
void add()
{
clrscr();
```

```
cout<<"Enter any two numbers: ";
cin>>a>>b;
c=a+b;
cout<<"Sum: "<<c;
}
};
void main()
{
sum s;
s.add();
getch();
}
```

Output:

Enter any two number:

4

5

Sum: 9

## ABSTRACTION IN C++:

Abstraction is the concept of exposing only the required essential characteristics and behavior with respect to a context.

Hiding of data is known as data abstraction. In object oriented programming language this is implemented automatically while writing the code in the form of class and object.

Real life example of Abstraction in C++


Abstraction shows only important things to the user and hides the internal details for example when we ride a bick, we only know about how to ride bick but can not know about how it work ? and also we do not know internal functionality of bick.

Example of Abstraction in C++

```
#include<iostream.h>
#include<conio.h>

class sum
{
// hidden data from outside world
private: int a,b,c;

public:
void add()
{
clrscr();
cout<<"Enter any two numbers: ";
cin>>a>>b;
c=a+b;
cout<<"Sum: "<<c;
}
};
void main()
```

```
{
sum s;
s.add();
getch();
}
```

Output

Enter any two number:

4

5

Sum: 9

Note: Data abstraction can be used to provide security for the data from the unauthorized methods.

## EXCEPTION HANDLING IN C++:

The process of converting system error messages into user friendly error message is known as Exception handling. This is one of the powerful feature of C++ to handle run time error and maintain normal flow of C++ application.

### Exception:

An exception is an event, which occurs during the execution of a program, that disrupts the normal flow of the program's Instructions.

### Handling the Exception:

Handling the exception is nothing but converting system error message into user friendly error message. Use Three keywords for Handling the Exception in C++ Language, they are;

try

catch

throw

## Syntax for handling the exception:

Example of Exception Handling in C++

```
try
{
    // causes executions code
}
catch( ExceptionName e1 )
{
    // catch block
}
catch( ExceptionName e2 )
{
    // catch block
}
catch( ExceptionName eN )
{
    // catch block
}
```

## Try Block:

It is one of the block in which we write the block of statements which causes executions at run time in other words try block always contains problematic statements.

### Catch block:

It is one of the block in which we write the block of statements which will generates user friendly error messages in other words catch block will suppose

system error messages.

**Example without Exception Handling:**

Example

```
#include<iostream.h>
#include<conio.h>
void main()
{
int a, ans;
a=10;
ans=a/0;
cout<<"Result: "<<ans;
}
```

Output

Abnormally terminate program

**Example of Exception Handling:**

Example

```
#include<iostream.h>
#include<conio.h>
void main()
{
int a=10, ans=0;
try
{
ans=a/0;
}
catch(int i)
{
cout<<"Denominator not be zero";
}
}
```

**Output:**

Denominator not be zero

# 10. CHAPTER – FILE HANDLING

## FILE HANDLING IN C++:

File handling concept in C++ language is used for store a data permanently in computer. Using file handling we can store our data in Secondary memory (Hard disk).

Why use File Handling,

- For permanet storage.
- The transfer of input - data or output - data from one computer to another can be easily done by using files.

For read and write from a file you need another standard C++ library called fstream, which defines three new data types:

| Datatype | Description |
|---|---|
| ofstream | This is used to create a file and write data on files |
| Ifstream | This is used to read data from files |
| Fstream | This is used to both read and write data from/to files |

### File Opening mode:

| Mode | Meaning | Purpose |
|---|---|---|
| ios :: out | Write | Open the file for write only. |
| ios :: in | read | Open the file for read only. |

| | | |
|---|---|---|
| ios :: app | Appending | Open the file for appending data to end-of-file. |
| ios :: ate | Appending | take us to the end of the file when it is opened. |

How to achieve File Handling

For achieving file handling in C++ we need follow following steps

- Naming a file
- Opening a file
- Reading data from file
- Writing data into file
- Closing a file

## Defining and Opening a File

The function open() can be used to open multiple files that use the same stream object.

Syntax:

file-stream-class stream-object;

stream-object.open("filename");

Example

ofstream outfile; // create stream

outfile . open ("data1"); // connect stream to data1

```
#include<fstream>
#include<iostream>
using namespace std;
int main()
{
        int a;
        cout<<"enter a";
```

```
        cin>>a;
        ofstream outfile;
        outfile.open("abc.txt",ios::app);
        outfile<<a<<"\n";
        outfile.close();
}
```

Reading the fILE:

```
#include<iostream>
#include<fstream>
using namespace std;
int main()
{
        int a;
        ifstream infile;
        infile.open("FileHandling.txt");
        infile>>a;
        cout<<a;
        infile.close();
}
```