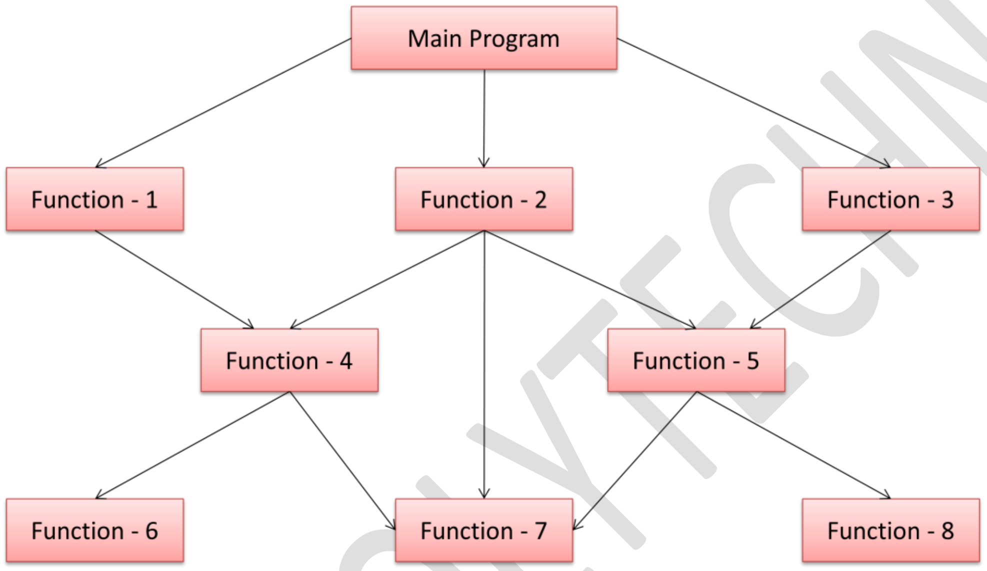
# Unit –1: Principles of Object Oriented Programming

## 1.1 Overview of Structure Programming language

❖ A structure language is a type of computer programming language that specifies a series of well-structured steps and procedures within its programming context to compose a program.

❖ It contains a systematic order of statements, functions and commands to complete a computational task or program.



**Structure of the procedure oriented programs**

❖ Disadvantage of POP:

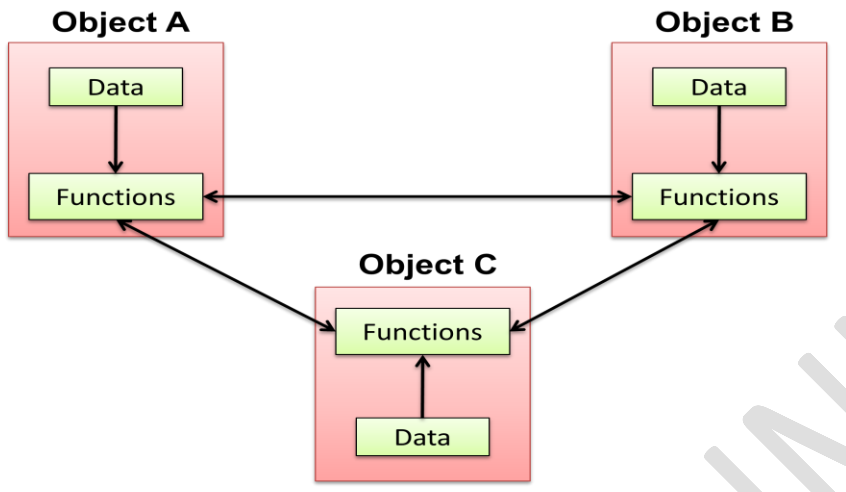
➢ To develop a large program, it is very difficult to identify what data is used by which function.

➢ It does not model (solve) real world problem very well.

## 1.2 The object-oriented Approach

❖ Object-oriented programming (OOP) is a programming paradigm based on the concept of "objects", which may contain data.

❖ OOP treats data as a critical element in the program development and does not allow it to flow freely around the system.



Organization of data and function in OOP

- It ties data more closely to the functions that operate on it, and protects it from accidental modification from outside function.
- OOP allows decomposition of a problem into a number of entities called objects and then builds data and functions around these objects.

## 1.3 Basic Concept of object-oriented Programming:-object, class, Inheritance, Data abstraction, encapsulation, polymorphism, dynamic binding, message passing

**1) Object :**

- Object can be defind as real world entities
- e.g. Person,Cycle,Chair, Pen, Window in computer software
- Objects can be either

  1)Physical Object :Person,Table

  2) Conceptual Object: vehicle, bird

  3) Abstract Object: God
- An object is represented as its property (or attributes) and operations performed on it.

  Object: Window

  Properties:

  size

  type

  position

  Operations:

  change_size( )

  move( )

  minimize( )

  maximize( )

  close( )

**2) Classes**

- A class in C++ is a user defined type or data structure declared with keyword class that has data and functions as its members.
- A class definition starts with the keyword class followed by the class name; and the class body, enclosed by a pair of curly braces.
- E.g.

```
class Box {
        public:
        double length;   // Length of a box
        double breadth;  // Breadth of a box
        double height;   // Height of a box
        };
```

**3) Abstraction**

- Abstraction means keeping necessary while hiding unnecessary.
- Data abstraction is define as providing only essential information to the outside world and hiding their background details.
- For example,
- when a person is working as employee in an organization,then in payroll system of organization,the salary is essential property,while weight is not essential.
- On other side if a person is a member of sportsclub then the weight is essential property,while salary is not essential.

**4) Encapsulation**

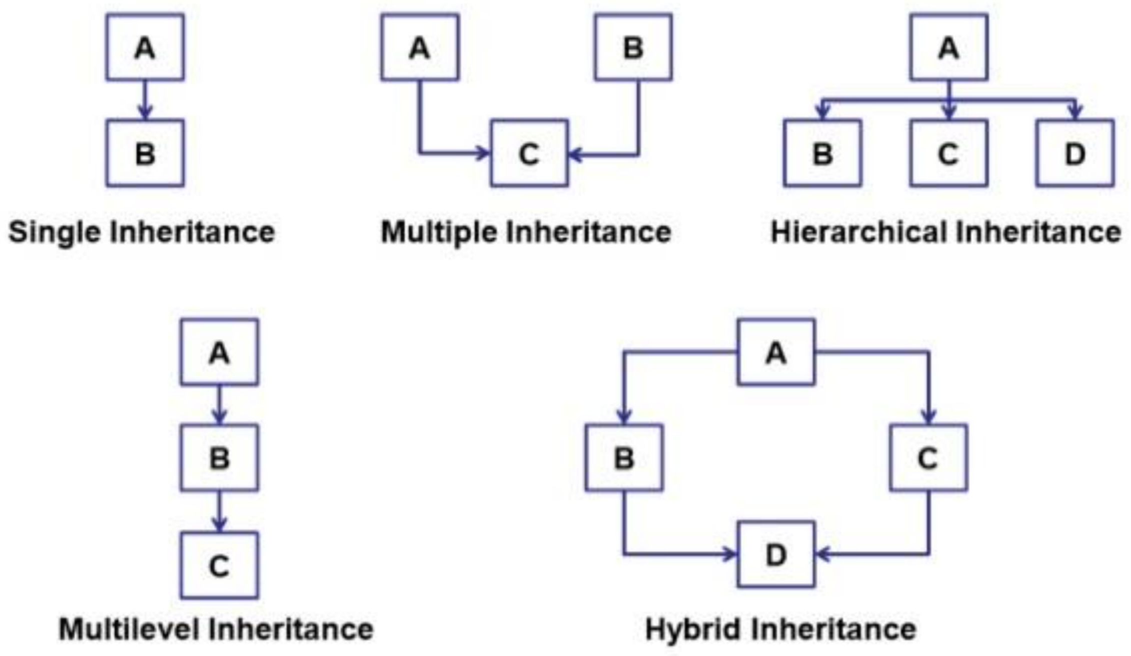
- Putting data and functions together in a single unit is known as encapsulation.
- It serves two purposes.

1. It puts data near to the functions operating on it.
2. It separates the external aspects of the object from internal implementation.

**5) Inheritance**

- Inheritance is a object by which one class gets the properties of other class.
- Class from which property are inherited is called base class.
- Class inheriting the property is called derived class.
- Types of Inheritance:

  1)Single Inheritance
  2) Multilevel Inheritance
  3) Multiple Inheritance
  4) Hierarchical Inheritance
  5) Hybrid Inheritance



**6) Polymorphism:**

- ❖ Polymorphism means different forms or ability to act differently in different situations.
- ❖ For example, + operator is used to add two integer numbers.
- ❖ Some OOP languages are also used + operator to add two strings.

**7) Message passing :**

- ❖ An object oriented program consists of a set of objects that communicate with each other.
- ❖ Steps for Message Passing:
    1. Creating classes that define objects and their behavior.
    2. Creating objects from class definition,
    3. Establishing communication among objects.
- ❖ Objects communicate with one another by sending and receiving information.

## 1.4 Advantages of Object Oriented Programming

- ❖ Data is given primary importance which allows real world entities to be modeled easily.
- ❖ Object correspond to its counterpart in real world system i.e. entities which makes direct correspondence between problem and program. This enables easy understanding of the complex system.
- ❖ Data and functions are closed in a single unit(Class) which make data secured and protects from accidental changes or misuse.
- ❖ Once object is implemented, it can be used in many systems without developing every time from scratch which saves the time and make program more reliable.

- ❖ Encapsulation separates the external and internal aspects of objects allowing modification to the object without effecting rest of the system.
- ❖ Inheritance allows real world relations to be modeled easily. It also allows sharing of features among related objects.
- ❖ Polymorphism provides flexibility to user by allowing multiple forms of same thing.
- ❖ Software systems are easy to maintain as concepts of objects clear boundaries making design easier.
- ❖ Grouping of related classes allows reusable component development in form of libraries. It is possible to assemble several such components from different source together to make a software.

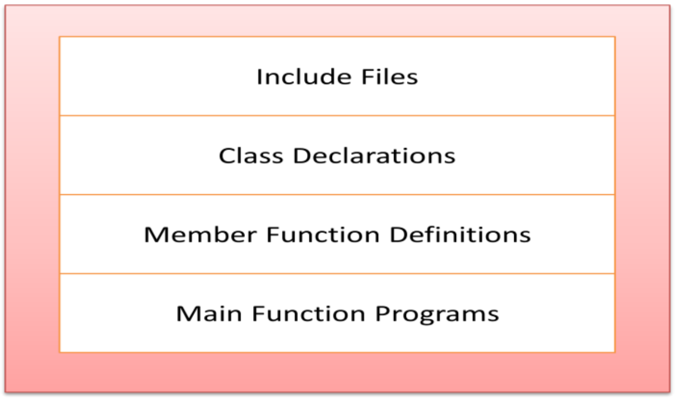
## 1.5 Usage of Object Oriented Programming languages

- ❖ Real-time systems.
- ❖ Simulation and modeling.
- ❖ Object-oriented databases.
- ❖ Hypertext, hypermedia and expertext.
- ❖ AI and expert systems.
- ❖ Neural networks and parallel programming.
- ❖ Decision support and office automation systems.
- ❖ CIM/CAM/CAD systems.

## 1.6 Features of Object Oriented Programming languages

- ❖ Emphasis on data rather than procedure.
- ❖ Programs are divided into objects.
- ❖ Data structures are designed such that they characterize the objects.
- ❖ Functions that operate on the data of an object are tied together in the data structure.
- ❖ Data is hidden and can't be accessed by external function.
- ❖ Objects may communicate with each other through functions.
- ❖ New data and functions can be easily added whenever necessary.

## 1.7 Structure of c++ Program



❖ **Include Files:**

- It is used to access the library function with are defined inside the header file.
- The most commonly used header files are iostream.h, conio.h, stdlib.h.

❖ **Class Declaration:**

- It contains the declaration of class. In which the data members and member function are declared.

❖ **Member function definition:**

- In this section, we can write the definition of the member function which are declared inside the class declaration. This member function definition is known as member function definition outside the class.

❖ **main( ) function:**

- It the function from where the execution of the C++ program is start. In C++, by default the return type of the main function is integer.

## 1.8 Output using cout

- The output operator, commonly known as the insertion operator <<.
- The output operator works on two operands, namely, the cout stream on its left and the expression to be displayed on its right.
- The output operator directs (inserts) the value to cout.
- The sentence in the instruction is enclosed between double quotes ( ” ), because it is constant string of characters.

**Syntax:**

```
cout<<expression
```

**Example:**

```
1)  cout<<”Hello World”;
2)  int a=10;
cout<<a;
```

## 1.9 Directives:-pre-processor directives, header files, The using Directives, Comments

- **pre-processor directives**
- The preprocessors are the directives, which give instructions to the compiler to preprocess the information before actual compilation starts.
- All the statements starting with the # (hash) symbol are known as preprocessor directives in C++.
- There are number of preprocessor directives supported by C++ like #include, #define, #if, #else, #line, etc.

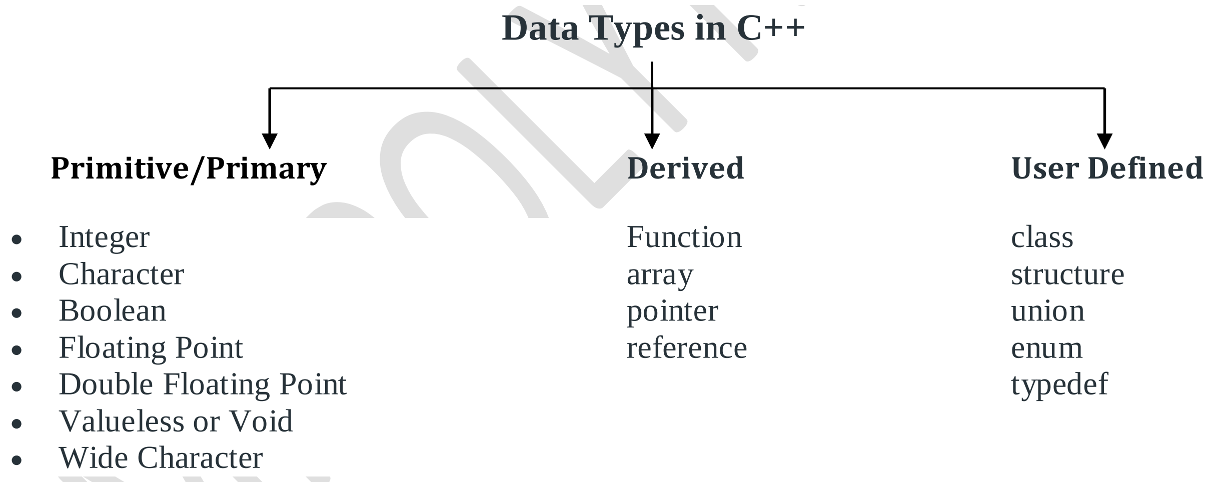
- ❖ **<u>Using Directives</u>**
- ❖ A using directive provides access to all namespace qualifiers and the scope operator.
- ❖ It allows a single name from a specific namespace to be used without explicit qualification in the declaration region in which it appears.
- ❖ The using directive may be applied at the global and local scope but not the class scope.
- ❖ Syntax:

  Using namespace *name*;

- ❖ **<u>Comment</u>**
- ❖ A C++ comment is written in one of the following two ways:
- ❖ // -- used for single line comment
- ❖ /* */ -- used for multi line comment

## 1.10 Basic Data types

- ❖ Data Types in C++ are divided into 3 Types:



**1. Primitive Data Types**:

These data types are built-in or predefined data types and can be used directly by the user to declare variables. example: int, char, float, bool, etc. Primitive data types available in C++ are:

- Integer
- Character
- Boolean
- Floating Point
- Double Floating Point
- Valueless or Void
- Wide Character

**2. Derived Data Types:**

Derived data types that are derived from the primitive or built-in datatypes are referred to as Derived Data Types. These can be of four types namely:

- Function
- Array
- Pointer
- Reference

**3. Abstract or User-Defined Data Types**:

Abstract or User-Defined data types are defined by the user itself. Like, defining a class in C++ or a structure. C++ provides the following user-defined datatypes:

- Class
- Structure
- Union
- Enumeration
- Typedef defined Datatype

## 1.11 Input using cin

- ❖ The cin object in C++ is an object of class iostream. It is used to accept the input from the standard input device i.e. keyboard.
- ❖ It is associated with the standard C input stream stdin.
- ❖ The extraction operator(>>) is used along with the object cin for reading inputs.
- ❖ The extraction operator extracts the data from the object cin which is entered using the keyboard.

**Syntax:**

```
cin  >> variable_name;
```

**Example:**

```
int age;
cin >> age;
```

## 1.12 Overview of operators: Types of operators, scope-resolution operator

- ❖ C++ supports set of operators including all the operators of C.
- ❖ All the C operators work with same meaning in C++.
- ❖ OOP adds many other operators like Scope resolution operator, memory management operators new and delete and some manipulators.

### Scope Resolution Operators (::)

- ❖ Local variable: A variable which is declared inside the function or block is called as the local variable for that function and block.
- ❖ Global variable: A variable which is declared outside the function or block is called as the global variable for that function and block.
- ❖ In C, to access the global version of the variable inside the function and block that is not possible.

- In C++, the global version of variable is accessed inside the function and block using the scope resolution operator.
- Scope resolution operator is used to uncover a hidden variable.
- **Syntax:** :: variable-name
- **Example:**

```
int a = 10;
void main( )
{
        int a = 15;
        cout<< " Value of a =  " << a<<endl;
        cout<< " Value of a =  " << ::a;
}
```

**Output**:

```
Value of a = 15
Value of a = 10
```

## Memory Management Operators

- In C, calloc( ) and malloc( ) function are used to allocate memory dynamically at runtime.
- Similarly, it uses the function free( ) to free dynamically allocated memory.
- In C++, the unary operators **new** and **delete** are used to allocate and free a memory at run-time.

### new operator

- An object can be created by using new and destroyed by using delete operator.
- The new operator can be used to create objects of any type.
- **Syntax:**

```
pointer-variable = new data-type;
```

- **Example:**

```
int *p = new int;
float *q = new float;
```

### delete operator

When a data object is no longer needed, it is destroyed to release the memory space for reuse.

- **Syntax:**

```
delete pointer-variable;
```

- **Example**:

```
delete p;
delete q;
```

## 1.13 Manipulators and Enumeration

### Manipulators

1) endl : Indicate End of Line

2) setw:

- ❖ Manipulators are operators that are used to format the output.
- ❖ Most commonly used manipulators are endl and setw.
- ❖ The endl manipulator is used in the output statement when we want to insert a linefeed.
- ❖ It has the same effect as using the newline character “\n”.
- ❖ The setw manipulator is used to format the output.
- ❖ When we use the setw manipulator the output is formatted in right justified.
- ❖ Before using the setw manipulator we have to include the header file iomanip.
- ❖ Syntax:

  setw(width)

  Where, width specifies the field width.
- ❖ Example:

```
#include<iostream.h>
#include<conio.h>
#include<iomanip.h>
void main( )
{
        clrscr();
        int a=15,b=420,c=3542;
        cout<< “a =”<<setw(4)<<a<<endl;
        cout<< “b =”<<setw(4)<<b<<endl;
        cout<< “c =”< setw(4)<<c<<endl;
        getch();
}
```

**Output:**

```
a =   15
b =  420
c =3542
```

### Enumeration

- ❖ An enumeration is a user-defined data type that consists of integral constants.
- ❖ To define an enumeration, keyword enum is used.
- ❖ It can be assigned some limited values.
- ❖ These values are defined by the programmer at the time of declaring the enumerated type.

❖ **Syntax:**

enum enumerated-type-name { value1, value2, value3…..valueN };

❖ **Example:**

```
enum week { Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday };
int main()
{
   week today;
   today = Wednesday;
   cout << "Day " << today+1;
   return 0;
}
```

**Output:**

```
Day 4
```