

# Unit 1

## Introduction to Object Oriented Programming



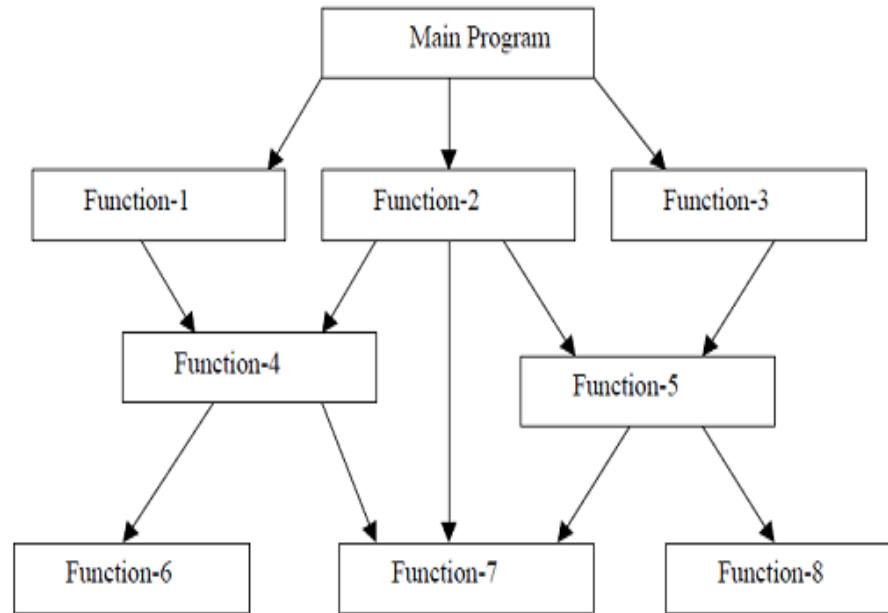
# Procedure Oriented Programming(POP)

- The procedure oriented programming is the programming language in which the procedure is important and a specific procedure is followed by the program.
- For example, C, COBOL, and FORTRAN
- In POP the problem is presented as the sequence of operation to be done such as accepting values, calculating, displaying etc, number of procedures are written to accomplish these operations.



# Procedure Oriented Programming(POP)

- Program is decomposed into smaller parts known as functions.



Structure of procedural oriented programs



# Procedure Oriented Programming(POP)

- A program in a procedural language is a list of instructions where each statement tells the computer to do something.
- It focuses on procedure(function) and algorithm.
- It follows the top-down approach.
- Most of the functions share global data.
- Data moves openly around the functions easily.
- It focuses on process rather than data.



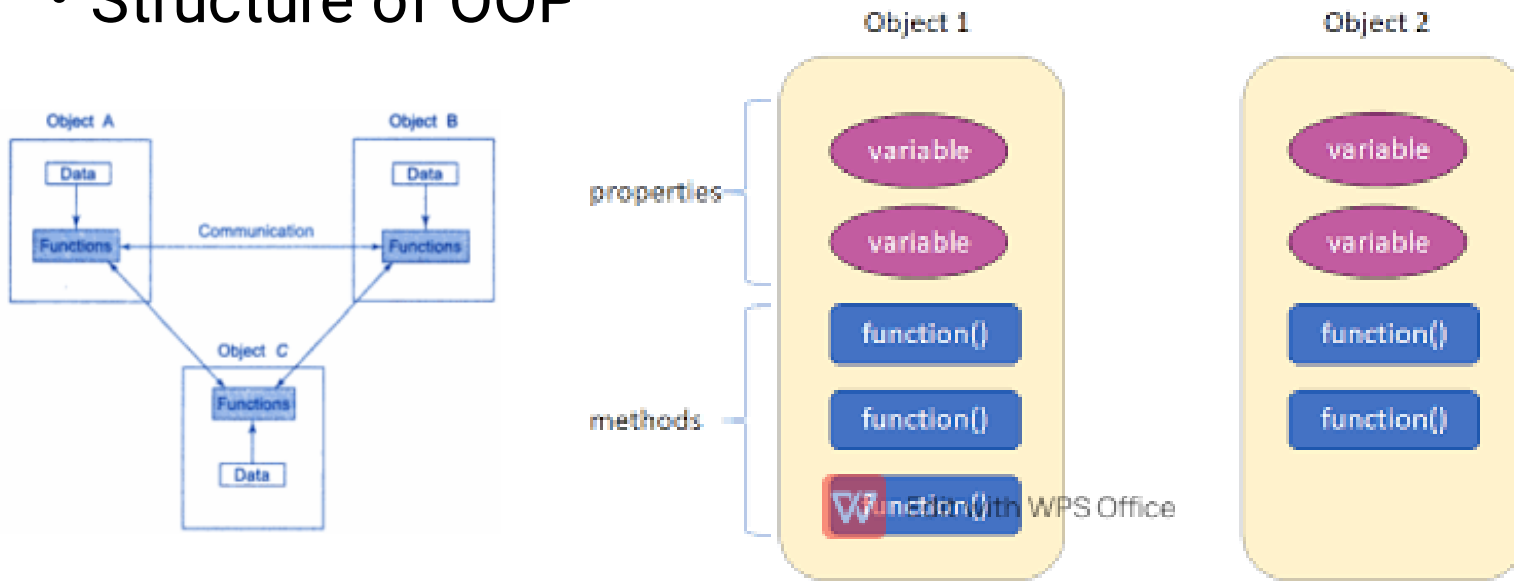
# Drawbacks

- It emphasis on doing things. Data is given a second class status even though data is the reason for the existence of the program.
- Since every function has complete access to the global variables, the new programmer can corrupt the data accidentally by creating function. Similarly, if new data is to be added, all the function needed to be modified to access the data.
- It is difficult to create new datatypes. The ability to create the new datatypes of its own is called extensibility. Structured programming language are not extensible.



# Object Oriented Programming (OOP)

- Object-oriented programming (OOP) is a programming paradigm based on the concept of "objects", which may contain data, in the form of fields, often known as attributes; and code, in the form of procedures, often known as methods.
- Structure of OOP



# Object Oriented Programming (OOP)

- Focus is on data rather than procedure.
- Large program is divided into smaller component is known as object.
- Data movement around the system is not possible.
- It follows bottom up approach.



# Difference between POP and OOP

Sr. No	On the basis of	Procedure Oriented Programming	Object Oriented Programming
1	Definition	It is a programming language that is derived from structure programming and based upon the concept of calling procedures. It follows a step-by-step approach in order to break down a task into a set of variables and routines via a sequence of instructions.	Object-oriented programming is a computer programming design philosophy or methodology that organizes/ models software design around data or objects rather than functions and logic.
2	Security	It is less secure than OOPs.	Data hiding is possible in object-oriented programming due to abstraction. So, it is more secure than procedural programming.
3	Approach	It follows a top-down approach.	It follows a bottom-up approach.
4	Data movement	In procedural programming, data moves freely within the system from one function to another.	In OOP, objects can move and communicate with each other via member functions.
5	Access	There are no access modifiers in	The access modifiers in OOP are named



# Difference between POP and OOP

Sr. No	On the basis of	Procedure Oriented Programming	Object Oriented Programming
6	<b>Inheritance</b>	Procedural programming does not have the concept of inheritance.	There is a feature of inheritance in object-oriented programming.
7	<b>Code reusability</b>	There is no code reusability present in procedural programming.	It offers code reusability by using the feature of inheritance.
8	<b>Importance</b>	It gives importance to functions over data.	It gives importance to data over functions.
9	<b>Data hiding</b>	There is not any proper way for data hiding.	There is a possibility of data hiding.
10	<b>Examples</b>	Examples of Procedural programming include C, Fortran, Pascal, and VB.	The examples of object-oriented programming are - NET, Perl, C#, Python, Java, VB.NET, and C++.



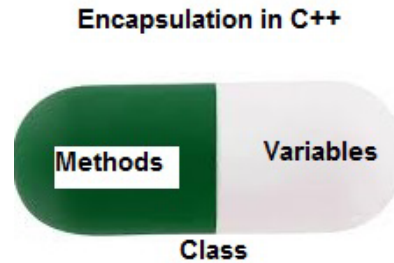
# Features of OOP

- Objects: Objects are the basic runtime entities of an object oriented system.
- They may represent a person, a place or any item that the program must handle.



# Encapsulation

- In normal terms, Encapsulation is defined as wrapping up of data and information under a single unit. In Object-Oriented Programming, Encapsulation is defined as binding together the data and the functions that manipulate them.



- Encapsulation also leads to *data abstraction or hiding*. As using encapsulation also hides the data. In the above example, the data of any of the section like sales, finance or accounts are hidden from any other section.



# Data abstraction

- Abstraction means displaying only essential information and hiding the details.
- Data abstraction refers to providing only essential information about the data to the outside world,
- Consider a real-life example of a man driving a car. The man only knows that pressing the accelerators will increase the speed of the car or applying brakes will stop the car but he does not know about how on pressing accelerator the speed is actually increasing, he does not know about the inner mechanism of the car or the implementation of accelerator, brakes etc in the car. This is what abstraction is. hiding the background details or implementation.

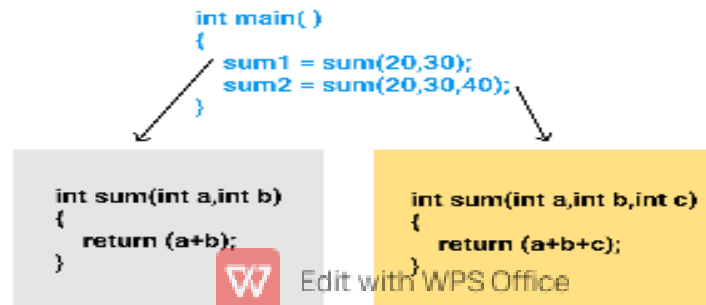


# Polymorphism

- The word polymorphism means having many forms. In simple words, we can define polymorphism as the ability of a message to be displayed in more than one form.
- A person at the same time can have different characteristic. Like a man at the same time is a father, a husband, an employee. So the same person possesses different behaviour in different situations. This is called polymorphism.
- An operation may exhibit different behaviours in different instances. The behaviour depends upon the types of data used in the operation.



- C++ supports operator overloading and function overloading.
- *Operator Overloading*: The process of making an operator to exhibit different behaviours in different instances is known as operator overloading.
- *Function Overloading*: Function overloading is using a single function name to perform different types of tasks.  
Polymorphism is extensively used in implementing inheritance.
- **Example**: Suppose we have to write a function to add some integers, some times there are 2 integers, some times there are 3 integers. We can write the Addition Method with the same name having different parameters, the concerned method will be called according to parameters.

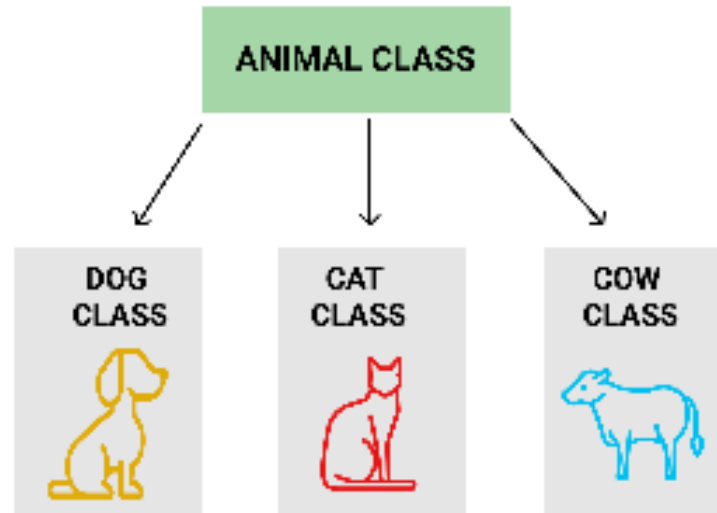


# Inheritance

- The capability of a class to derive properties and characteristics from another class is called Inheritance. Inheritance is one of the most important features of Object-Oriented Programming.
  - **Sub Class:** The class that inherits properties from another class is called Sub class or Derived Class.
  - **Super Class:** The class whose properties are inherited by sub class is called Base Class or Super class.
- **Reusability:** Inheritance supports the concept of “reusability”, i.e. when we want to create a new class and there is already a class that includes some of the code that we want, we can derive our new class from the existing class. By doing this, we are reusing the fields and methods of the existing class.



- **Example:** Dog, Cat, Cow can be Derived Class of Animal Base Class.





# Dynamic Binding

- **Dynamic Binding:** In dynamic binding, the code to be executed in response to function call is decided at runtime.
- It is a process of linking procedure call to specific sequence code (method) run time.
- It is also known as late binding or run-time binding.
- C++ has virtual functions to support this.



# Message Passing

- Objects communicate with one another by sending and receiving information to each other.
- A message for an object is a request for execution of a procedure and therefore will invoke a function in the receiving object that generates the desired results.
- Following are the steps in message passing:
  1. Creating classes that define objects and its behavior
  2. Creating objects from class definitions
  3. Establishing communication among objects



# Applications of OOP

- Real time systems
- Simulation and modeling
- Object Oriented Databases
- Hypertext, hypermedia and expert text
- AI and expert system
- Neural networks and parallel programming
- Decision support and office automation



# The limitations of C programming languages are as follows:

- Difficult to debug.
- C allows a lot of freedom in writing code, and that is why you can put an empty line or white space anywhere in the program. And because there is no fixed place to start or end the line, so it isn't easy to read and understand the program.
- C compilers can only identify errors and are incapable of handling exceptions (run-time errors).
- C provides no data protection.
- It also doesn't feature the reusability of source code extensively.



# Structure of C++

- The structure of the program written in C++ language is as follows:

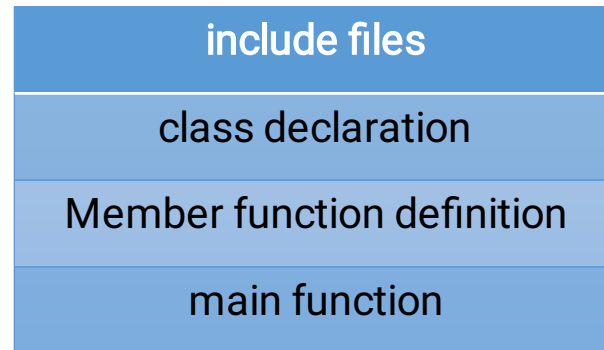


Fig. Structure of C++



# Structure of C++

- Consider the Hello World! program-

1. `#include<iostream>`
2. `using namespace std;`
3. `int main()`
4. `{`
5. `cout<<"Hello World!"<<endl;   //Displays HelloWorld!`
6. `return 0;`
7. `}`



# Structure of C++

- **Line 1: `#include<iostream>`** : It is a preprocessor directive which tells compiler that includes the next file before compiling program.
- The header file `iostream` should be included at the beginning of all program that uses input/output statements.
- **Line 2: `using namespace std;`** tells the compiler to use the `std` (standard) namespace. In this program `std` is the namespace where ANSI C++ standard class libraries are defined.
- A namespace permits grouping of various entities like classes, objects, functions, and various C++ tokens, etc. under a single name.
- Any user can create separate namespaces of its own and can use them in any other program.
- In the program namespace `std` contains declarations for `cout`, `cin`, `endl`, etc. statements.
- Namespaces can be accessed in multiple ways:
  - `using namespace std;`
  - `using std :: cout;`



# Structure of C++

- **Line 3: `int main()`** ; Every C++ program must have a `main()`. Like C, in C++ also program execution begins at `main()`.
- **Line 5: `cout<<"HelloWorld!" <<endl;`** is output statement.
- “cout” refers to the standard output(or console)output.
- The symbol `<<` is called the stream **insertion operator( put-to-operator)**, which is used to put the string “Hello World!” to the console.
- “endl” denotes the end-of-line or newline, which is put to the console to bring the cursor to the beginning of the next line.
- “cin” refers to the standard input.
- C++ also uses `>>` extraction operator( get-from-operator) to read data from a standard input device.





# Key concepts

- **Tokens:** Tokens are the smallest individual units in the program.
- C++ has following tokens:
  - o **Keywords:** The keywords are the reserved words by the language to understand instruction specifically.
  - o **Constants:** Constants are refer to fixed values that do not change during execution of a program
  - o **Special Symbol**
  - o **Identifiers :** Identifiers refers as the names of variables, user defined functions, arrays, classes etc..
    - Following are the rules:
      1. The first character of any identifier must be a letter or an underscore(\_).
      2. C++ is a case sensitive language.
      3. An identifier should not be a keyword.
  - o **Operators**



# Variables in C++

- A named memory location use to store data and whose content may change during the execution of program is called as variable.
- A variable in C++ is an identifier that refers to the data item stored at a particular memory location. This data item can be accessed in the program simply by using the variable name.
- The rules for defining variable names:-
  1. The name of variable should be unique and a valid C++ identifier.
  2. Variable name can be as a single letter and first 32 characters can be considered.
  3. Do not use keywords as variable names.
  4. Variable names must begin with a letter of the alphabet or an underscore.
  5. No blank space is allowed in a variable name.

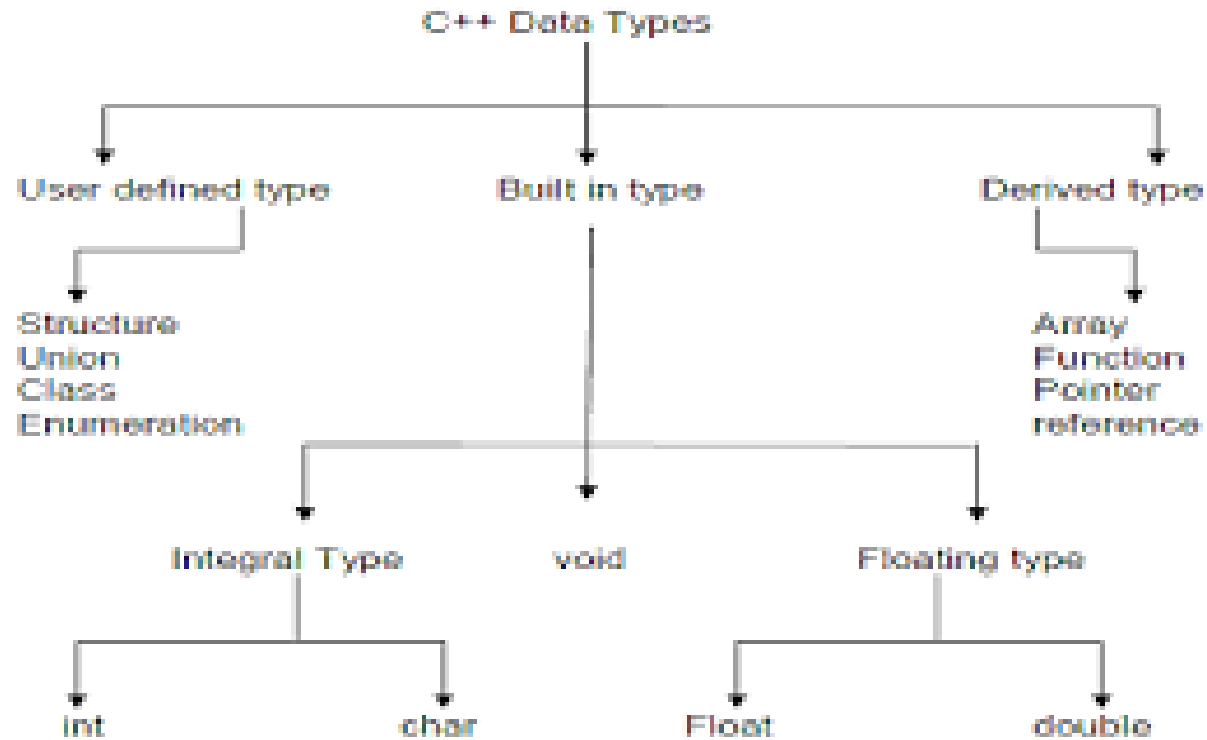


# Variables in C++

- Declaration of Variable:
- Syntax: `data_type variable_name;`
- Initialization of variables:
- Syntax: `data_type variable_name=initial_value;`



# Data Types



# References in C++

- When a variable is declared as a reference, it becomes an alternative name for an existing variable.
- A variable can be declared as a reference by putting '&' in the declaration.
- `int x=10;`

`//ref is a reference to x`

`int & ref=x;`

`//Value of x is now changed ref=20;`

`cout<<"x= "<<x<<endl;`

`x=30;`

`cout<<"ref= "<<ref<<endl;`

Output:  
x=20  
ref=30



# Scope Resolution Operator

- C++ supports a mechanism to access a global variable from a function in which a local variable is defined with the same name as a global variable. It is achieved using a scope resolution operator.
- The scope resolution operator is denoted by pair of colon (::).

- The syntax is `::variable_name;`

```
#include <iostream>
using namespace std;
int a=10;
int main(){
    int a=15;
    cout<<"\n Local a="<<a<<"Global a="<<::a;
    ::a=20;
    cout<<"\n Local a="<<a<<"Global a="<<::a;
    return 0;
}
```

Output:

Local a=15 Global a=10

Local a=15 Global a=20



# Memory Management Operators

- In C, malloc() and calloc() functions for dynamic memory allocation and to release memory free() function is used.
- For same task in C++, new and delete operators used for allocating and deallocating the memory.



# new Operator

- new operator is used to create objects of any type.
- Syntax: `ptr_var =new data_type;`
- The new operator allocates sufficient memory to hold data object of type data type and returns the address of the object
- For example, `int *p; float *q;`  
`p=new int;`  
`q=new float`
- Here p is a pointer of type int and q is a pointer of type float.
- We can also initialize value using new operator.
- e.g. , `int * p=new int(20)`  
`int *q=new float(4.5);`
- We can also allocate memory to the array using new operator.
- E.g. `int *p=new int [20];`





# delete Operator

- When the need of any data object is ended and we don't want that any more at that time we can use delete operator to release memory space allocated by that object.
- Syntax: `delete ptr_var;`
- The `ptr_var` is the pointer to a data object which is created by `new`.
- E.g. `delete p;`    `delete q;`
- If the array is allocated by `new`, then to free it we use the following syntax: `delete [size] ptr_var;`
- E.g. `delete [5] a;`



# Type casting

- Type casting means conversion of one data type to another data type.
- There are two types of conversions in C++-
  1. Implicit typecasting
  2. Explicit typecasting
- **Implicit Typecasting:** In this, conversion is done automatically by compiler.
- In this type of conversion, a smaller data type is converted into a larger data type.



# Type casting

- **Explicit Typecasting:** It refers to the type conversion that is performed explicitly using type cast operator.
- Consider example, `int sum=489;`  
`float avg;`  
`avg=sum/5;`
- In the above example, the avg value is expected 97.80 but it will return the value 97.00 because sum is an int and 5 is also integer. So division of integer must be integer.
- So for correct result, type casting is done as given below:  
`avg=(float) sum/5;`
- Syntax: `data_type(expression)` OR  
`(data_type) expression` where, data\_type is data type(also known as cast operator) to which expression is to be converted.



# Manipulators in C++

- Manipulators are helping functions that can modify the input/output stream.
  - It does not change the value of variable, it only modifies the I/O stream using insertion(<<) and extraction(>>) operators.
1. endl Manipulator: endl is line feed operator in C++.
  2. setw Manipulator: This manipulator sets the minimum field width on output. Syntax: setw(x)
  3. setfill Manipulator: If a value does not entirely fill a field, then the character specified in the setfill argument of the manipulator is used for filling the fields.

