Introduction to C++

- What is C++
- Applications of C++
- Features of C++
- OOPs concepts
- Code conventions
- Classes & Objects

C++

C++ is an extension to C language and was developed by Bjarne Stroustrup at bell labs in 1979.Originally named C with classes but later it was renamed C++ in 1983.C++ adds many features to the C language.

- C++ is an intermediate(middle) level language, as it comprises a confirmation of both high level and low level language features.
- C++ is a compiled general-purpose language.
- C++ is procedural and Object Oriented Programming language but is not purely Object Oriented.
- C++ is a superset of C, and that virtually any legal C program is a legal C++ program.

Applications of C++:

- Design Operating system
- Design Language Compiler
- Design Database
- Application Software
- To write device drivers etc.

Features of C++

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

- Simple
- Platform dependent
- Portability
- Object oriented Programming language
- Powerful
- Compiler based
- Syntax based language

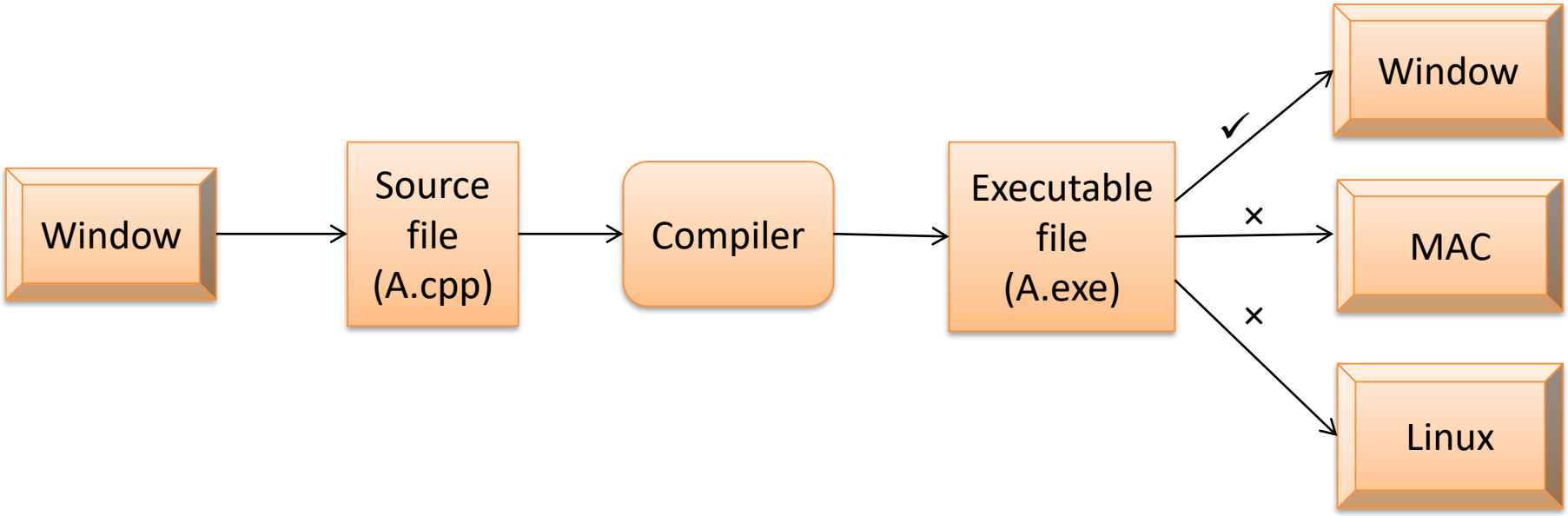
Features of C++

1. Simple:

Every C++ program can be written in simple English language so that it is very easy to understand and developed by programmer.

2. Platform dependent:

A language is said to be platform dependent whenever the program is execute in the same operating system where that was developed and compiled but not run and execute on other operating system. C++ is platform dependent language.



Features of C++

3. Portability:

It is the concept of carrying the instruction from one system to another system. In C++ Language .cpp file contain source code, we can edit also this code. .exe file contain application, only we can execute this file. When we write and compile any C++ program on window operating system that program easily run on other window based system.

4. Object oriented programming language:

This main advantage of C++ is, it is object oriented programming language. It follow concept of oops like polymorphism, inheritance, encapsulation, abstraction.

5. Powerful:

C++ is a very powerful programming language, it have a wide verity of data types, functions, control statements, decision making statements, etc.

6. Compiler based:

C++ is a compiler based programming language that means without compilation no C++ program can be executed. First we need compiler to compile our program and then execute.

7. Syntax based language:

C++ is a strongly tight syntax based programming language. If any language follow rules and regulation very strictly known as strongly tight syntax based language.

OOPs Concept

OOPs (Object Oriented Programming System)

Object-Oriented Programming is a methodology or paradigm to design a program using classes and objects. It simplifies the software development and maintenance by providing some concepts:

1. Object
2. Class
3. Encapsulation
4. Abstraction
5. Inheritance
6. Polymorphism

OOPs Concept

1) Object:
Any entity that has state and behavior is known as an object. For example: chair, pen, table, keyboard, bike etc.

Characteristics of an object:
state: represents data (value) of an object.
behavior: represents the behavior (functionality) of an object

2) Class:
Collection of objects is called class. It is a logical entity.It is a template or blueprint from which objects are created.

3) Encapsulation:
Binding (or wrapping) code and data together into a single unit is known as encapsulation. It keeps the data and the code safe from external interference.
A class is the example of encapsulation.

4) Abstraction:
Hiding internal details and showing functionality is known as abstraction.
For example: phone call, we don't know the internal processing.
In C++, we use abstract class achieve abstraction.

OOPs Concept

5) Polymorphism:
When one task is performed by different ways i.e. known as polymorphism.
For example: to convenes the customer differently, to draw something e.g. shape or rectangle etc.

6) Inheritance:
When one object acquires all the properties and behaviors of parent object i.e. known as inheritance. It provides code reusability. It is used to achieve runtime polymorphism.

C++

Syntax and Structure of C++ program:

Here we will discuss one simple and basic C++ program to print "Hello World" and its structure.

First C++ program:

```
#include <iostream.h>
 using namespace std;
 int main()
 {
 cout << "Hello World";
 }
```

Header files are included at the beginning just like in C program.Header files contained predeclared function libraries, which can be used by users for their ease.

Using namespace std, tells the compiler to use standard namespace. Namespace collects identifiers used for class, object and variables.

main(), is the function which holds the executing part of program its return type is int.

cout <<, is used to print anything on screen, same as printf in C language.

C++

Code Conventions:

■ **Classes**

The first letter should be capitalized, and if several words are linked together to form the name, the first letter of the inner words should be uppercase (a format that's sometimes called "camelCase"). For classes, the names should typically be nouns. For example:

Dog
Account
PrintWriter

Methods

The first letter should be lowercase, and then normal camelCase rules should be used. In addition, the names should typically be verb-noun pairs. For example:

getBalance
doCalculation
setCustomerName

Classes & Objects

Classes:
The classes are the most important feature of C++ that leads to Object Oriented programming. Class is a user defined data type, which holds its own data members and member functions, which can be accessed and used by creating instance of that class.
A class is a group of objects that has common properties. It is a template or blueprint from which objects are created.

A class can contain:
- data member
- Member functions
- constructor

Syntax to declare a class:

```
class <class_name>
{
   data_member;
   member functions;
};
```

Classes & Objects

Object:

Class is mere a blueprint or a template. No storage is assigned when we define a class. Objects are instances of class, which holds the data variables declared in class and the member functions work on these class objects.

An Object in C++ has three characteristics:

- State:
  Represents data (value) of an object.
- Behavior:
  Represents the behavior (functionality) of an object such as deposit, withdraw etc.
- Identity:
  Object identity is typically implemented via a unique ID. The value of the ID is not visible to the external user.

Syntax:

class_name object_reference;

Example:

Employee e;

Classes & Objects

Accessing data members & member functions:

The data members & member functions of a class can be accessed using the direct member access operator (.)

Syntax to access data members of a class:

obj_name.datamember_name;

Syntax to access member functions of a class:

obj_name.func_name(<list of arguments>);

Classes & Objects

Simple program using classes & objects:

```
#include<iostream.h>
#include<conio.h>
using namespace std;
class Employee
{
public:
int id;    // data member
void getId()    //member function
{
cout<<"Enter id: ";
cin>>id;
cout<<"Id is : "<<id;
}};
int main()
{
Employee e;  //creating an object of Employee
e.getId();
getch();
}
```

Classes & Objects

O/P:

Enter id: 4500

Id is : 4500

In this example, we have created a Employee class that have one data member id & one member function getId(). We are creating the object of the Employee class and printing the objects value using getId() member function.

Classes & Objects

Scope resolution operator in class:

Member functions can be defined within the class definition or separately(outside the class) using scope resolution operator(::)

Example:

```
#include<iostream>
#include<conio.h>
using namespace std;
class Employee
{
public:
int id;    // data member
void getId();    //member function declaration
};
void Employee::getId()    //member function
{
cout<<"Enter id: ";
cin>>id;
cout<<"Id is : "<<id;
}
```

Classes & Objects

```
int main()
{
Employee e;  //creating an object of Employee
e.getId();
getch();
}
```

Discussions