C++

An object oriented language

Programming Languages – A history

- Machine Language
  - Everything in 0s and 1s
  - Very tedious and impossible to write
- Assembly language
  - Written using mnemonics
  - Complex and non-portable (different for different processors)
- High Level languages - e.g BASIC, COBOL, FORTRAN
  - English like
  - Program is set of sequential instructions with goto statements
  - Spaghetti code
- Structured languages - e.g. C, Pascal
  - Program divided into functions
  - stand-alone subroutines, local variables, rich control constructs, and lack of reliance upon the GOTO.
  - Difficulty in passing data between functions for large projects

Object Oriented Languages

- Centered around **data** unlike procedure oriented languages
- Functions are encapsulated along with data they manipulate

Some OOP languages

- C++, Java, C#, Eiffel, Simula, Smalltalk

Important Features of OOPs

- Abstraction:
  - Creation of a model considering only relevant data
- Encapsulation
  - Packing together function with data in objects
- Polymorphism
  - Capability to behave differently at different contexts.
- Inheritance
  - Creating new data types by enhancing existing data types

OOP principles

- Everything is an object
- Object has a state and behavior
- Each object has a data type
- It occupies memory (similar to variable)
- Objects receive messages to do certain actions
- Object's interface is separated from implementation and implementation is hidden from client programmers

Features of C++

- C++ is a superset of C.
- With added OOPs features
- Developed by Bjarne (j silent) Stroustrup in 1979 – called it C with classes
- ISO standard in 1999

Features of C++

- Abstraction
- Polymorphism
- Inheritance
- Exception handling
- Templates
- Multiple Inheritance
- Reference parameters

Difference b/w C and C++

1. C follows the procedural programming paradigm while C++ is a multi-paradigm language(procedural as well as object oriented).
2. In case of C, the data is not secured while the data is secured(hidden) in C++.
3. C uses the top-down approach while C++ uses the bottom-up approach.
4. C is function-driven while C++ is object-driven.
5. C++ supports function overloading while C does not.
6. We can use functions inside structures in C++ but not in C.
7. The NAMESPACE feature in C++ is absent in case of C.
8. The standard input & output functions differ in the two languages.
9. C uses scanf & printf while C++ uses cin>> & cout<< as their respective input & output functions.
10. C++ allows the use of reference variables while C does not.
11. C++ supports Exception Handling while C does not.

Structures in c++

- Struct in c++ can have data members as well as function members
- Function members are called using struct variable name, dot operator and function name
- E.g.

```
Student s1;
S1.printDetails();
```

Classes and objects

- Class is description
- Defines the data and function members
- Memory is not allocated for class
- Class is a user defined data type

- Object is one instance of a class (like variable)

Class and struct

- Class in c++ is similar to struct. With the only difference that in struct all members are public
- In class members by default are private

Sample program

No .h

```
#include<iostream>
using namespace std;
int main()
{
  cout<<“Hello world\n”;
   //displays Hello world
  return 0;
}
```

function type is a must

Namespace

- Huge projects global names create name conflict
- Divide the global namespace into manageable pieces
- Similar to struct, union and class, a namespace puts the names in a specified scope.
- Namespace definition is enclosed in a block.
- No ; at end
- Namespaces can be nested

```
namespace n1
{
   void fun(char*s)
   {
        cout<<s<<"\n";
   }
}
```

Namespace – contd..

- To access a name within a namespace **use :: operator**

```
n1::fun("Hello");
std::cout<<"hello
   world";
```

- Or use **namespace declaration** at the beginning

```
using std::cout;
int main(){---
----
cout<<"Hello";----}
```

- Or use namespace directive

```
using namespace std;
```

Namespace

- Namespace is a scope
- All the library names are in the **namespace std**

Input and Output

- C++ uses **cin** object with extraction operator(>>) for input
- And **cout** object with insertion operator(<<) for output
- Both are overloaded for all built in data types and char*
- No format specifiers

```
C
++++++++++++
```

C++ features

- Variables can be declared anywhere.
- Function prototypes and return types are mandatory

C++ data types

- int, char, float, double

And

- bool
  - Can take values true and false
  - Treated as int with true =1 and false = 0
  - Can use ints also

Functions

- Overloaded functions
  - Multiple functions with same name
  - But different number and/or type of parameter
  - Proper function will be called by compiler depending on type and number of arguments

```
int largest(int a,int b)
{
   return a>b?a:b;
}

int largest(int a,int b,int c)
{
   if(a>b)
        return a>c?a:c;
   else
        return b>c?b:c;
}

float largest(float a,float b)
{
   return a>b?a:b;
}
```

```
int  main()
{
    int x=15,y=45,z=77;
   cout<<largest(x,y);

   cout<<largest(x,y,z);

   float a=77.8,b=99.1;
   cout<<largest(a,b);



}
```