Page 1

30 October 2020 19:13

Preprocessor Directives

The preprocessors are the directives, which give instructions to the compiler to preprocess the information before actual compilation starts.
All preprocessor directives begin with #

Types of Preprocessor Directives -

a. Macros – starts with "#define"
b. File Inclusion – starts with #include<filename> or #include "filename" (For custom files).
c. Conditional Compilation – It tells complier to execute or skip a piece of code based on some condition.

Virtual Function

A virtual function is a member function which is declared within a base class and is re-defined(Overriden) by a derived class. Functions are declared with a **virtual** keyword in base class. The resolving of futypnction call is done at Run-time.
We use virtual function to let compiler know that the call to this function needs to be resolved at runtime (also known as **late binding** and dynamic linking) so that the object type is determined and the correct version of the function is called.

Runtime Errors

Runtime errors occurs during program execution.

A. Integer Overflow
B. Divide by Zero
C. Segmentation Fault

OOPS CPP

Class - A class is like a blueprint of data member and functions and object is an instance of class.

```
MyClass myclasso;       // if you only need to call the default constructor
MyClass myclasso(12);  // if you need to call constructor with parameters*

MyClass *myclassp = new MyClass();        // if you only need to call the default constructor
MyClass *myclassp = new MyClass(12);    // if you need to call constructor with parameters

myclasso.member;
myclassp->member;
```

Abstraction - Abstraction means displaying only essential information and hiding the details.
An abstract class at least one pure virtual function. A pure virtual function is one which is assigned to 0.

```
virtual func() = 0;
```

Inheritance - It allows child to acquire the properties of a parent class.

Types of Inheritance

A. **Single** - In Single inheritance one class inherits one class exactly.
B inherits A

B. **Multilevel** - In this type of inheritance one class inherits another child class.
C inherits B and B inherits A

C. **Multiple** - In multiple inheritance, a class can inherit more than one class.
C inherits A and B both

D. **Hierarchical** - In this type of inheritance, one parent class has more than one child class
Class B and C inherits class A

Polymorphism - It allows objects to behave differently under different conditions.

Compile time - This is also as static (or early) binding.(Function & opeartor Overloading)

Runtime Polymorphism – This is also known as dynamic (or late) binding. (Function Overriding).

Function overloading

```
#include <iostream>
using namespace std;
class Add {
public:
 int sum(int num1, int num2){
   return num1+num2;
 }
 int sum(int num1, int num2, int num3){
   return num1+num2+num3;
 }
};
int main() {
 Add obj;
 //This will call the first function
 cout<<"Output: "<<obj.sum(10, 20)<<endl;
 //This will call the second function
 cout<<"Output: "<<obj.sum(11, 22, 33);
 return 0;
}
```

Function Overriding

```
#include <iostream>
using namespace std;
class A {
public:
 virtual void disp(){
   cout<<"Super Class Function"<<endl;
 }
};
class B: public A{
public:
 void disp(){
   cout<<"Sub Class Function";
 }
};
int main() {
 //Parent class object
 A obj;
 obj.disp();
 //Child class object
 B obj2;
 obj2.disp();
 return 0;
}
```

Dynamic Memory Allocation - IN CPP the dynamic memory allocation is done using **new** keyword. Allocates memory for the object from the Free storage.
The new requires us to explicitly delete object later or it will cause memory leak. The deletion is done using "delete" variable.

```
int *a = new int[10];
delete[] a;
```

Malloc Vs New -

1. Malloc doesn't call the constructor of class and is not type-safe. New call the constructor of class and is type-safe.
2. New is slower than the malloc.
3. New doesn't have a way to reallocate memory.

Member Function of Class - The member function are functions which have declaration inside class definition. The definition of member functions can be inside or outside the definition of class.

Member function inside class

```
class Cube
{
   public:
   int side;
   int getVolume()
   {
      return side*side*side;     //returns volume of cube
   }
};
```

Member function outside class

```
class Cube
{
   public:
   int side;
   int getVolume();
}
// member function defined outside class definition
int Cube :: getVolume()
{
   return side*side*side;
}
```

Friend class and function in C++ - A friend class can access private and protected members of other class in which it is declared as friend.
A friend function can access the private and protected member.

```
class Node {
private:
   int key;
   Node* next;
   // Now class  LinkedList can
   // access private members of Node
   friend class LinkedList;
};
```