Basic Programming

Starting with C++

Classes and Objects

What are Classes?

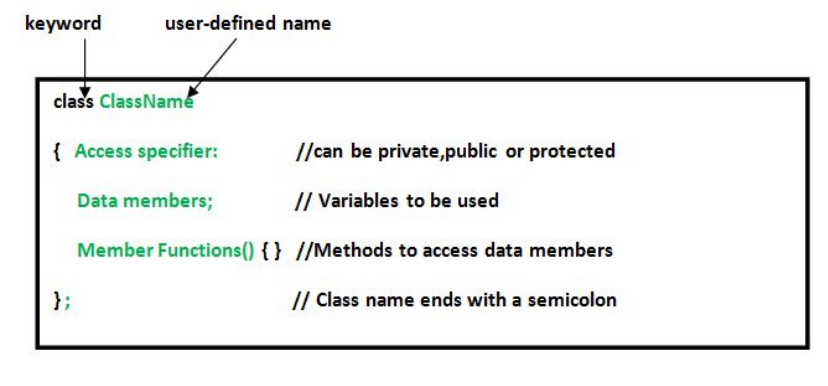
Class: The building block of C++ that leads to Object Oriented programming is a **Class**. It is a user defined data type, which holds its own data members and member functions, which can be accessed and used by creating an instance of that class. A class is like a blueprint for an object.

For Example: Consider the Class of **Cars**. There may be many cars with different names and brand but all of them will share some common properties like all of them will have *4 wheels*, *Speed Limit*, *Mileage range* etc. So here, Car is the class and wheels, speed limits, mileage are their properties.

- A Class is a user defined data-type which has data members and member functions.
- Data members are the data variables and member functions are the functions used to manipulate these variables and together these data members and member functions defines the properties and behavior of the objects in a Class.
- In the above example of class *Car*, the data member will be *speed limit*, *mileage* etc and member functions can be *apply brakes*, *increase speed* etc.
- An **Object** is an instance of a Class. When a class is defined, no memory is allocated but when it is instantiated (i.e. an object is created) memory is allocated.

Defining Class and Declaring Objects

A class is defined in C++ using keyword class followed by the name of class. The body of class is defined inside the curly brackets and terminated by a semicolon at the end.



Declaring Objects: When a class is defined, only the specification for the object is defined; no memory or storage is allocated. To use the data and access functions defined in the class, you need to create objects.

Syntax:

```
ClassName ObjectName;
```

Accessing Data members

```cpp
// C++ program to demonstrate
// accessing of data members

#include <bits/stdc++.h>
using namespace std;
class Geeks
{
    // Access specifier
    public:

    // Data Members
    string geekname;

    // Member Functions()
    void printname()
    {
       cout << "Geekname is: " << geekname;
    }
};

int main() {

    // Declare an object of class geeks
    Geeks obj1;

    // accessing data member
    obj1.geekname = "Abhi";

    // accessing member function
    obj1.printname();
    return 0;
}
```

Output:

```
Geekname is: Abhi
```

Accessing member function

```cpp
// C++ program to demonstrate function
// declaration outside class

#include <bits/stdc++.h>
using namespace std;
class Geeks
{
    public:
    string geekname;
    int id;

    // printname is not defined inside class defination
    void printname();

    // printid is defined inside class defination
    void printid()
    {
        cout << "Geek id is: " << id;
    }
};

// Definition of printname using scope resolution operator ::
void Geeks::printname()
{
    cout << "Geekname is: " << geekname;
}
int main() {

    Geeks obj1;
    obj1.geekname = "xyz";
    obj1.id=15;

    // call printname()
    obj1.printname();
    cout << endl;

    // call printid()
    obj1.printid();
    return 0;
}
```

Output:

```
Geekname is: xyz
Geek id is: 15
```

Constructors

Constructors are special class members which are called by the compiler every time an object of that class is instantiated. Constructors have the same name as the class and may be defined inside or outside the class definition.

There are 3 types of constructors:

- Default constructors
- Parameterized constructors
- Copy constructors

Making Constructors

```cpp
// C++ program to demonstrate constructors

#include <bits/stdc++.h>
using namespace std;
class Geeks
{
    public:
    int id;

    //Default Constructor
    Geeks()
    {
        cout << "Default Constructor called" << endl;
        id=-1;
    }

    //Parametrized Constructor
    Geeks(int x)
    {
        cout << "Parametrized Constructor called" << endl;
        id=x;
    }
};
int main() {

    // obj1 will call Default Constructor
    Geeks obj1;
    cout << "Geek id is: " <<obj1.id << endl;

    // obj1 will call Parametrized Constructor
    Geeks obj2(21);
    cout << "Geek id is: " <<obj2.id << endl;
    return 0;
}
```

Output:

```
Default Constructor called
Geek id is: -1
Parametrized Constructor called
Geek id is: 21
```

Destructor

Destructor is another special member function that is called by the compiler when the scope of the object ends.

```
// C++ program to explain destructors

#include <bits/stdc++.h>
using namespace std;
class Geeks
{
    public:
    int id;

    //Definition for Destructor
    ~Geeks()
    {
        cout << "Destructor called for id: " << id <<endl;
    }
};

int main()
  {
    Geeks obj1;
    obj1.id=7;
    int i = 0;
    while ( i < 5 )
    {
        Geeks obj2;
        obj2.id=i;
        i++;
    } // Scope for obj2 ends here

    return 0;
  } // Scope for obj1 ends here
```

Output:

```
Destructor called for id: 0
Destructor called for id: 1
Destructor called for id: 2
Destructor called for id: 3
Destructor called for id: 4
Destructor called for id: 7
```