C++ Constructor

In C++, constructor is a special method which is invoked automatically at the time of object creation. It is used to initialize the data members of new object generally. The constructor in C++ has the same name as class or structure.

There can be two types of constructors in C++.

- Default constructor
- Parameterized constructor

C++ Default Constructor

A constructor which has no argument is known as default constructor. It is invoked at the time of creating object.

Let's see the simple example of C++ default Constructor.

```
1. #include <iostream>
2. using namespace std;
3. class Employee
4.  {
5.    public:
6.        Employee()
7.        {
8.            cout<<"Default Constructor Invoked"<<endl;
9.        }
10.};
11.int main(void)
12.{
13.   Employee e1; //creating an object of Employee
14.   Employee e2;
15.   return 0;
16.}
```

Output:

```
Default Constructor Invoked
Default Constructor Invoked
```

C++ Parameterized Constructor

A constructor which has parameters is called parameterized constructor. It is used to provide different values to distinct objects.

Let's see the simple example of C++ Parameterized Constructor.

```
#include <iostream>
#include <string>
using namespace std;
class Student
{
	string name;
	public:
		Student( string n )
		{
			name = n;
		}
		Student()
		{
			name = "unknown";
		}
		void printName()
		{
			cout << name << endl;
		}
};
int main()
{
	Student a( "xyz" );
	Student b;
	a.printName();
	b.printName();
	return 0;
}
```

Destructor

A destructor works opposite to constructor; it destructs the objects of classes. It can be defined only once in a class. Like constructors, it is invoked automatically.

A destructor is defined like constructor. It must have same name as class. But it is prefixed with a tilde sign (~).

```
1. #include <iostream>
2. using namespace std;
3. class Employee
4.  {
5.    public:
6.        Employee()
7.        {
8.            cout<<"Constructor Invoked"<<endl;
9.        }
10.       ~Employee()
11.       {
12.           cout<<"Destructor Invoked"<<endl;
13.       }
14.};
15.int main(void)
16.{
17.   Employee e1; //creating an object of Employee
18.   Employee e2; //creating an object of Employee
19.   return 0;
20.}
```

Output:

```
Constructor Invoked
Constructor Invoked
Destructor Invoked
Destructor Invoked
```

Function Overloading

```
#include<iostream>
#include<string.h>
using namespace std;
class first
{
	int a,b;
	float x,y;
	public:
			void functionoverload(int a,int b)
			{
				int c;
				c=a+b;
				cout<<"the Int Addition:- "<<c;
				cout<<endl;
				//cout<<"first function call";
			}
			void functionoverload(float x,float y)
			{
				float z;
				z=x+y;
				cout<<"the Float Addition:- "<<z;
				cout<<endl;
			}
			void functionoverload(string a,string b)
			{
```

```
                a="Hello";
                b="Good Evening";
                cout<<a<<endl<<b;
                cout<<endl;
            }
};

int main()
{
    int a,b;
    float x,y;
    first f;
    cout<<"Enter the first number:- "<<endl;
    cin>>a;
    cout<<"enter the second number:- "<<endl;
    cin>>b;
    f.functionoverload(a,b);
    cout<<"Enter the first number:- "<<endl;
    cin>>x;
    cout<<"enter the second number:- "<<endl;
    cin>>y;
    f.functionoverload(x,y);
    //f.functionoverload(a,b);
    return 0;
}
```

Function Overriding

```
#include<iostream>
#include<string.h>
using namespace std;
class override
{

        public:
                        void car()
                        {

                                cout<<"This is The Hundai Company Car"<<endl;

                        }

};

class cars : public override
{
        public:
                        void car()
                        {
                                cout<<"This is the Maruti Suzuki Company Car"<<endl;
                        }
};
```

```
int main()
{
	override od;
	od.car();
	cars cr;
	cr.car();
	return 0;
}
```