Constructor

Constructor:

A constructor is a special type of member function of a class which initializes objects of a class. In C++, Constructor is automatically called when object(instance of class) create. It is special member function of the class because it does not have any return type.

A constructor resembles an instance method, but it differs from a method in that it has no explicit return type, it is not implicitly inherited and it usually has different rules for scope modifiers. Constructors often have the same name as the declaring class. They have the task of initializing the object's data members and of establishing the invariant of the class, failing if the invariant is invalid.

Most languages allow overloading the constructor in that there can be more than one constructor for a class, with differing parameters. Some languages take consideration of some special types of constructors. Constructors, which concretely use a single class to create objects and return a new instance of the class, are abstracted by factories, which also create objects but can do so in various ways, using multiple classes or different allocation schemes such as an object pool.

Example:

```
#include<bits/stdc++.h>
using namespace std;
class cons{
    public:
       int x;
       int y;

    public:
        cons(){

            cout<<"EMPTY"<<endl;
  }


};
int main(){
  cons val;
}
```

Output:

EMPTY

Types of Constructors:

* Default constructor
* Copy constructor
* Parameterized constructor

1)Default constructor:

If the programmer does not supply a constructor for an instantiable class, compiler inserts a default constructor into your code on your behalf. This constructor is known as default constructor. You would not find it in your source code as it would be inserted into the code during compilation and exists in .class file. The behavior of the default constructor is language dependent. It may initialize data members to zero or other same values, or it may do nothing at all.

Example:

```
#include<bits/stdc++.h>
using namespace std;
class cons{
    public:
       int x;
       int y;

    public:
        cons(){
           int x = 30;
           int y = 20;
           cout<<"Value of a : "<< x
<<endl<<"Value of b : "<< y <<endl;
```

```
  }


};
int main(){
    cons val;
}
```

Output:

```
Value of a : 30
Value of b : 20
```

2) Copy constructor:

A copy constructor is a member function which initializes an object using another object of the same class. Detailed article on Copy Constructor.Whenever we define one or more non-default constructors (with parameters) for a class, a default constructor(without parameters) should also be explicitly defined as the compiler will not provide a default constructor in this case. However, it is not necessary but

it’s considered to be the best practice to always define a default constructor.

Example:

```
#include<bits/stdc++.h>
using namespace std;
class cons{
    public:
       int x;
       int y;

    public:
```

```
        cons(int x,int y){
           int a = x;
           int b = y;
           cout<<"Value of a :
"<<a<<endl<<"Value of b : "<<b<<endl;
        }

        cons(cons &val){
          int *b,*a;
          b = &x;
          a = &y;
          std::cout<<"Value of a : "<< *a
<<endl<<"Value of b : "<< *b <<std::endl;
        }
};

int main(){
  cons val1(20,30);
  cons val2 = val1;
}
```

Output:

Value of a : 20
Value of b : 30
Value of a : 32765
Value of b : 594481184

3)Parameterized constructor:

It is possible to pass arguments to constructors. Typically, these arguments help initialize an object when it is created. To create a parameterized constructor, simply add parameters to it the way you

would to any other function. When you define the constructor’s body, use the parameters to initialize the object.

Example:

```
#include<bits/stdc++.h>
using namespace std;
class cons{
    public:
       int x;
```

```
    int y;

  public:
     cons(int x,int y){
        int a = x;
        int b = y;
        cout<<"Value of a :
"<<a<<endl<<"Value of b : "<<b<<endl;
  }


};
int main(){
  cons val(20,30);
}
```

Output:

```
Value of a : 20
Value of b : 30
```