Chapter 4 : Constructor and Destructor

Reference : E. Balaguruswamy Object Oriented Programming With C++; chapter-6.

Topics:

- **Topic 1: Basic of Constructor & Destructor**
- **Topic 2: Overloaded Constructor**
- **Topic 3: Constructor with default argument**
- **Topic 4: Copy constructor**

Questions:

Topic 1: Basic of Constructor & Destructor

1. What are constructor and destructor? When are they executed? Write down the properties of them. Or, What are constructor and destructor functions? Write down the characteristics of them.
2. Write down the difference between constructor and destructor?
3. Is it mandatory to use constructor in a class?

Topic 2: Overloaded Constructor

4. How is the constructor overloaded? Give me two reasons why you need to overload a class's constructor?

Topic 3: Constructor with default argument

5. What do you mean by default constructor and default argument constructor(or default value constructor)? Why do we need default argument constructor(or default value constructor)?Distinguish between default constructor and default argument constructor(or default value constructor)

Topic 4: Copy constructor

6. What is a copy constructor? How can a copy constructor copy the value of an object? Show the use of a copy constructor with an example program.

Q:1:: What are constructor and destructor? When are they executed? Write down the properties of them. Or, What are constructor and destructor functions? Write down the characteristics of them.

Answer:

Constructor: A constructor is a special member function which enables an object to initialize itself when it is created. That's why constructor known as automatic initialization of objects. It is special because its name is the same as the class name. The constructor is invoked whenever an object of its associated class is created. It is called constructor because it constructs the values of data members of the class.

```
1   #include<iostream>
2   using namespace std;
3   class add
4   {
5       private:
6           int a=10,b=20;
7       public:
8           add()
9           {
10              cout<<"Addition = "<<a+b<<endl;
11          }
12  };
13  int main()
14  {
15      add ad1;
16      return 0;
17  }
```

```
Addition = 30

Process returned 0 (0x0)   execution time : 0.077 s
Press any key to continue.
```

Characteristics of the constructor : The constructor functions have some special characteristics. These are :

• They should be declared in the public section,

• They are invoked automatically when the objects are created.

• They do not have return types, not even void and so they cannot return values.

• They cannot be inherited, though a derived class can call the base class constructor.

• Like other C++ functions, they can have default arguments.

• Constructors cannot be virtual.

• We cannot refer to their addresses.
• An object with a constructor (or destructor) cannot be used as a member of a union.
• They make “implicit calls” to the operators new and delete when memory allocation is required.
• When a constructor is declared for a class, initialization of the class objects becomes mandatory.

Destructor: A destructor, as the name implies is used to destroy the object that have been created by a constructor. Like a constructor, the destructor is a member function whose name is the same as the class name but is preceded by a tilde(~). For example, the destructor for the class add can be defined as shown below:

```
~add()
{
    cout<<"From destructor: object becoming destroy....."<<endl;
}
```

An example program with constructor and destructor:

```
1  #include<iostream>
2  using namespace std;
3  class add
4  {
5      private:
6          int a=10,b=20;
7      public:
8          add()
9          {
10             cout<<"From constructor: object initialized!!"<<endl;
11             cout<<"Addition = "<<a+b<<endl<<endl;
12         }
13         ~add()
14         {
15             cout<<"From destructor: object becoming destroy....."<<endl;
16         }
17 };
18 int main()
19 {
20     add ad1;
21     return 0;
22 }
```

```
From constructor: object initialized!!
Addition = 30
From destructor: object becoming destroy.....
```

Characteristics of the destructor :

- A destructor is a member function whose name is the same as the class name but is preceded by a tilde(~).
- A destructor never takes any argument nor does it return any value.
- It will be invoked implicitly by the compiler upon exit from the program (or block or function as the case may be) to clean up storage that is no longer accessible.
- It is a good practice to declare destructors in a program since it releases memory space for future use.

When constructor and destructor are executed :

Constructor and destructor both are member functions of the class.When object of the specific class is created then implicitly constructor and destructor are called by the object i.e. constructor and destructor both are executed then if they are explicitly created before within the specific class.

Q:2:: Differentiate between constructor and destructor.

Answer:

Constructor	Destructor
Constructors help allocate memory to an object.	Destructors deallocate the memory of an object.
Constructors can take arguments.	Destructors don't take any arguments.
Constructors are called automatically when an object is created.	Destructors are called automatically when the block is exited or when the program terminates.
Constructors allow an object to initialize a value before it is used.	They allow objects to execute code when it is being destroyed.
They are called in the successive order of their creation.	They are called in the reverse order of their creation.
There can be multiple constructors in a single class.	There is a single destructor in a class.
Constructors can be overloaded.	Destructors cannot be overloaded.
The concept of copy constructor allows an object to get initialized from another object.	There is no such concept in the case of destructors.

Q:3:: Is it mandatory to use constructor in a class?

Answer:

No, it is not mandatory to use constructor in a class.We use constructor to automatically initialize the object at the time of its declaration and use destructor to deallocate memory or destroy the created object which is not required. But these things are not mandatory for object oriented programming.So, without using constructor in a class we can perform programming with simple class,object and member functions.

Q:4:: How is the constructor overloaded? Give me two reasons why you need to overload a class's constructor?

Answer:

When more than one constructor function is defined in a class we say that the constructor is overloaded.The criteria to overload a constructor is to differ the number of arguments or the type of arguments.The compiler differentiate which constructor to be called depending upon the number of arguments and their sequence of data types.

Necessity of constructor overloading:

1. To bring flexibility of creating various types of objects in a class we can use the constructor overloaded concept.
2. For solving complex problems we can use the constructor overloaded concept since it is easier to remember if several constructors have the same name.

There is an example program which demonstrates the constructor overloaded concept.

Code link 1

Code link 2

```cpp
#include<iostream>
#define pi 3.1416
using namespace std;
class area
{
    public:
        area()
        {
            cout<<"Area calculation"<<endl;
        }
        area(int r)
        {
            cout<<"Area of the circle = "<<pi*r*r<<endl;
        }
        area(double r)
        {
            cout<<"Area of the square = "<<r*r<<endl;
        }
        area(double h,double w)
        {
            cout<<"Area of the rectangle = "<<h*w<<endl;
        }
        area(int h,int w)
        {
            cout<<"Area of the triangle = "<<0.5*h*w<<endl;
        }
};
int main()
{
    area aa;
    area circle(3);
    area square(3.0);
    area rectangle(5.0,6.0);
    area triangle(5,6);
    return 0;
}
```

```
Area calculation
Area of the circle = 28.2744
Area of the square = 9
Area of the rectangle = 30
Area of the triangle = 15
```

Q:5:: What do you mean by default constructor and default argument constructor(or default value constructor)? Why do we need default argument constructor(or default value constructor)?Distinguish between default constructor and default argument constructor(or default value constructor).

Answer:

Default constructor : A constructor without any arguments(parameters) or with the default value for every parameter i.e the constructor which does not require any arguments is said to be the default constructor. Default constructor also known as zero argument constructor. Syntax :

```
class class_name
{
    public:
        class_name(){};
};
```

or,

```
class class_name
{
    public:
        class_name()
        {
            statements;
        };
};
```

or,

```
class class_name
{
    public:
        class_name(default_parameter_list)
        {
            statements;
        };
};
```

Some features of default constructor:

- Default constructor does not require any arguments(parameters).
- If the default constructor is not defined in the program by the programmer, then the compiler defines the default constructor implicitly during compilation.
- If the default constructor is defined explicitly in the program by the programmer, then the compiler will not define the constructor implicitly, but it calls the constructor(defined by the programmer) implicitly.

Default argument constructor(or default value constructor): A constructor with the default value for at least one parameter(argument) is said to be the default argument constructor or default value constructor. In default argument constructor if every parameter has default value then it is called default constructor, since then the constructor does not require any arguments.

If the programmer passes values i.e. mention arguments for the default parameters(arguments) then the default parameters(arguments) values will be overridden by the passed value while the constructor will be invoked by the compiler implicitly.

Syntax:

```
class class_name
{
    public:
        class_name(default_parameter_list)
        {
            statements;
        };
};
```

Need of default argument constructor(or default value constructor):

Default argument constructor is useful in situations where some arguments always have the same value. For instance, bank interest may remain the same for all customers for a particular period of deposit. It also provides a greater flexibility to the programmer. A constructor can be written with more parameters than are required for its most common application. Using default arguments, a programmer can use only those arguments that are meaningful to a particular situation. Default arguments can be used to combine similar constructor functions into one.

Difference between default constructor and default argument constructor:

	Default constructor		Default argument constructor
1.	A constructor without any arguments (parameters) or with the default value for every parameter is said to be the default constructor.	1.	A constructor with the default value for at least one parameter(argument) is said to be the default argument constructor.
2.	It is also known as the zero argument constructor.	2.	It is also known as the default value constructor.
3.	Default constructor does not require any arguments(parameters).	3.	Default argument constructor may require one or more arguments.
4.	Default constructor can be invoked by the compiler implicitly.	4.	Default argument constructor is always invoked by the compiler explicitly if all parameters have not been initialized with default values.
5.	The compiler defines the default constructor implicitly during compilation when the default constructor is not defined explicitly.	5.	The compiler does not define the default argument constructor implicitly during compilation when the default argument constructor is not defined explicitly.
6.	Default constructor can not be a default argument constructor.	6.	Default argument constructor can be a default constructor when every parameter has default values.

7.	Example:
	`class add`
	`{`
	`  int sum;`
	`  public:`
	`  // add(){}; //default constructor`
	`  add()  //default constructor`
	`  {`
	`  sum = 0;`
	`  }`
	`};`
		7.	Example:
			`class add`
			`{`
			`  int sum;`
			`  public:`
			`  add(int a,int b,int c = 10)`
			`  {`
			`  sum = a + b + c;`
			`  cout<<"Sum = "<<sum<<endl;`
			`  }`
			`};`

Here is an example program to demonstrate the use of default constructor and default argument constructor:

```cpp
#include<iostream>
using namespace std;
class add
{
    int sum;
    public:
        // add(){}; //default constructor
        add()         //default constructor
        {
            sum = 0;
        }
        add(int a,int b,int c = 10) //default argument constructor
        {
            sum = a + b + c;
            cout<<"Sum = "<<sum<<endl;
        }
};
class subtract
{
    int sub = 0;
    public:
        subtract(int a = 50,int b = 10)    //default constructor
        {
            sub = a - b;
            cout<<"Subtract = "<<sub<<endl;
        }
};
int main()
{
    add a1;
    add a2(30,20);
    add a3(30,20,40);

    subtract s1;
    return 0;
}
```

```
Sum = 60
Sum = 90
Subtract = 40

Process returned 0 (0x0)   execution time : 0.113 s
Press any key to continue.
```

[Code link]

Q:6::What is a copy constructor? How can a copy constructor copy the value of an object? Show the use of a copy constructor with an example program.

Answer:

Copy constructor : A copy constructor is a constructor function which takes a reference of an object of its own class as a parameter(argument). A copy constructor is used to declare and initialize an object by another object of the same class.Example:

```
code(code &x)
{
    id = x.id;
}
```

Mechanism : Copy constructor initializes the members(data/function) of a newly created object by copying the members(data/function) of an already existing object of the same class.The reference of an object must be used as a parameter(argument) in a copy constructor function because it only accepts pass by reference not the pass by value of the object.

Characteristics of copy constructor:

1. The copy constructor is used to initialize the members of a newly created object by copying the members of an already existing object.
2. Copy constructor takes a reference to an object of the same class as an argument.
3. The process of initializing members of an object through a copy constructor is known as copy initialization.
4. It is also called member-wise initialization because the copy constructor initializes one object with the existing object, both belonging to the same class on a member-by-member copy basis.
5. The copy constructor can be defined explicitly by the programmer. If the programmer does not define the copy constructor, the compiler does it for us.

Here is an example program to demonstrate the use of copy constructor :

Code link 1

```cpp
#include<iostream>
using namespace std;
class code
{
    int id;
    public:
        code(){};        //default constructor
        code(int a)      //constructor
        {
            id = a;
        }
        code(code &x)    //copy constructor
        {
            id = x.id;
        }
        int display()
        {
            return id;
        }
};
int main()
{
    code obj1(100);  //constructor called
    cout<<"id of obj1 : "<<obj1.display()<<endl;

    code obj2(obj1); //copy constructor called
    cout<<"id of obj2 : "<<obj2.display()<<endl;

    code obj3 = obj1; //copy constructor called
    cout<<"id of obj3 : "<<obj3.display()<<endl;

    code obj4;
    obj4 = obj1; //copy constructor called
    cout<<"id of obj4 : "<<obj4.display()<<endl;
    return 0;
}
```

```
id of obj1 : 100
id of obj2 : 100
id of obj3 : 100
id of obj4 : 100

Process returned 0 (0x0)   execution time : 0.092 s
Press any key to continue.
```