Constructors & Destructor

PRESENTED BY –ANJALI SONA

DEPARTMENT OF COMPUTER SCIENCE

INTRODUCTION

- Constructors are special class functions which performs initialization of every object. The Compiler calls the Constructor whenever an object is created. Constructors initialize values to object members after storage is allocated to the object.
- Whereas, Destructor on the other hand is used to destroy the class object.
- Before moving forward with Constructors and Destructors in C++ language, check these topics out to understand the concept better:

1. Function in C++
2. Class and Objects in C++
3. Data Members

- Let's start with Constructors first, following is the syntax of defining a constructor function in a class:

```
class A
 {
public: int x;
 // constructor A()
 {
// object initialization
 }
 };
```

While defining a contructor you must remeber that the **name of constructor** will be same as the **name of the class**, and contructors will never have a return type.

Constructors can be defined either inside the class definition or outside class definition using class name and scope resolution :: operator.

```
class A
{
   public:
 int i;
  A(); // constructor declared
};


// constructor definition
A::A()
{
   i = 1;
}
```

Types of Constructors in C++

Constructors are of three types:

1. Default Constructor
2. Parametrized Constructor
3. Copy COnstructor

Default Constructors

- Default constructor is the constructor which doesn't take any argument. It has no parameter.

SYNTAX-

```
class_name(parameter1, parameter2, ...)
{
    // constructor Definition
}
```

EXAMPLE

```
class Cube
{
   public:
 int side;
   Cube()
  {
     side = 10;
   }
};

int main()
{
   Cube c;
   cout << c.side;
}
```

OUTPUT – 10

Parameterized Constructors

- These are the constructors with parameter. Using this Constructor you can provide different values to data members of different objects, by passing the appropriate values as argument.

EXAMPLE

```
class Cube
{
   public:
   int side;
 Cube(int x)
 {
side=x;
 }
};
int main()
{
  Cube c1(10);
 Cube c2(20);
Cube c3(30);
cout << c1.side;
cout << c2.side;
cout << c3.side;
}
```

OUTPUT

10

20

30

Copy Constructors

- These are special type of Constructors which takes an object as argument, and is used to copy values of data members of one object into other object. We will study copy constructors in detail later.

Constructor Overloading in C++

- Just like other member functions, constructors can also be overloaded. Infact when you have both default and parameterized constructors defined in your class you are having Overloaded Constructors, one with no parameter and other with parameter.
- You can have any number of Constructors in a class that differ in parameter list.

Open filec.c Save

```cpp
// c++ program to illustrate the concept of constructors

#include <iostream>
using namespace std;

class integer
{
public:
  int x, y;

// Default Constructor declared
  integer()
  {
    x = 50;
    y = 20;
  }
};

int main()
{
  // Default constructor called automatically when the object is created
  integer a;
  cout << "x: " << a.x << endl << "y: " << a.y <<endl;
  return 0;
}
```

What is a Destructor in C++?

- Destructor is a member function that is instantaneously called whenever an object is destroyed. The destructor is called automatically by the compiler when the object goes out of scope i.e. when a function ends the local objects created within it also gets destroyed with it. The destructor has the same name as the class name, but the name is preceded by a tilde
- **Syntax of Destructor:**

```
class scaler {
 public:
scaler(); //constructor
~scaler(); //destructor
};
```

Characteristics of a Destructor in C++

- A destructor deallocates memory occupied by the object when it's deleted.
- A destructor cannot be overloaded. In function overloading, functions are declared with the same name in the same scope, except that each function has a different number of arguments and different definitions. But in a class, there is always a single destructor and it does not accept any parameters, hence, a destructor cannot be overloaded.
- A destructor is always called in the reverse order of the constructor. In C++, variables and objects are allocated on the Stack. The Stack follows LIFO (Last-In-First-Out) pattern. So, the deallocation of memory and destruction is always carried out in the reverse order of allocation and construction. This can be seen in code below.
- A destructor can be written anywhere in the class definition. But to bring an amount of order to the code, a destructor is always defined at the end of the class definition.

Implementation of Constructors and Destructors in C++

```
#include <iostream>

using namespace std;
class Department {
 public:
   Department() {
 //constructor is defined
  cout << "Constructor Invoked for Department class" << endl;
  }

   ~Department() {
     //destructor is defined
     cout << "Destructor Invoked for Department class" << endl;
   }
};
class Employee {
 public:
  Employee() {
    //constructor is defined
    cout << "Constructor Invoked for Employee class" << endl;
   }

   ~Employee() {
     //destructor is defined
     cout << "Destructor Invoked for Employee class" << endl;
   }
};
int main(void) {
  Department d1; //creating an object of Department
  Employee e2; //creating an object of Employee
  return 0;
}
```

OUTPUT

Constructor Invoked for Department class

Constructor Invoked for Employee class

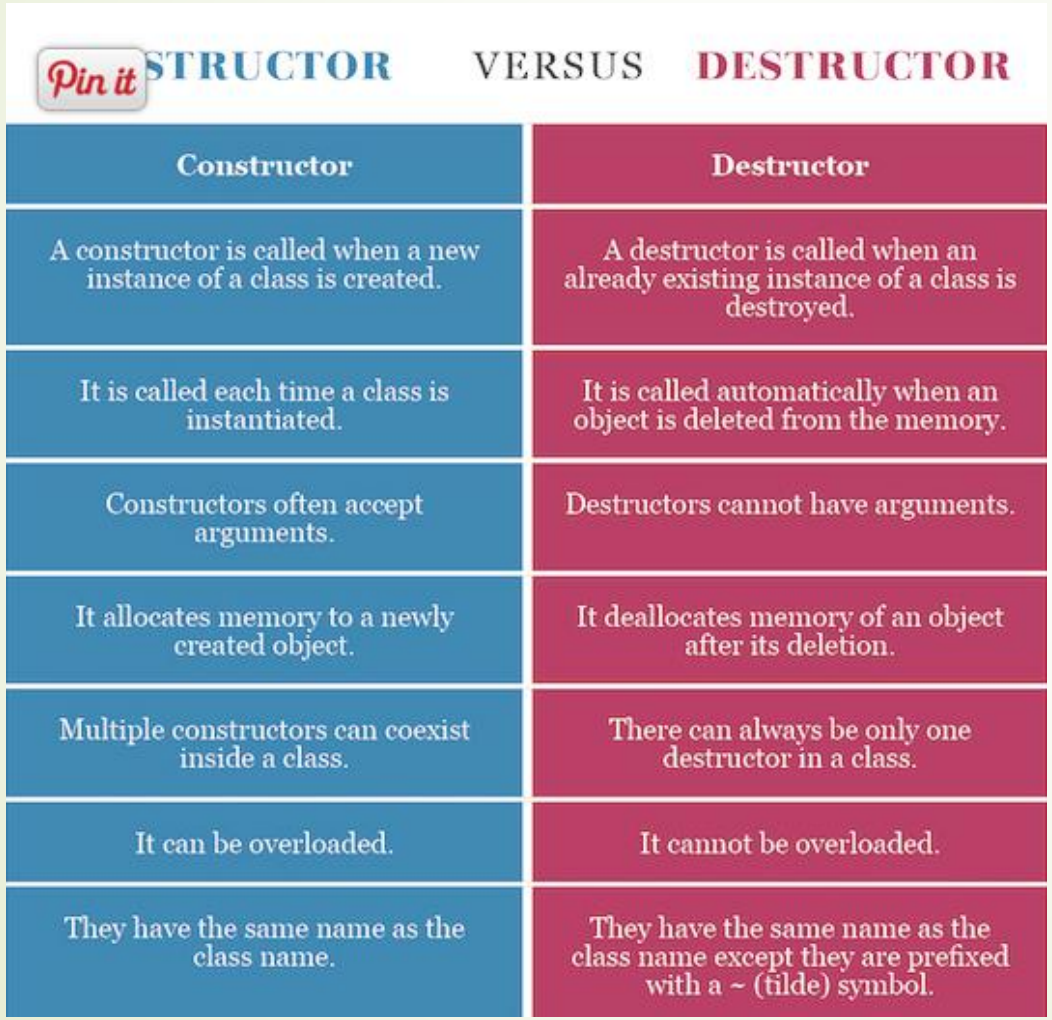
Destructor Invoked for Employee class

Destructor Invoked for Department class

Explanation:

- When an object named d1 is created in the first line of main() i.e (Department d1), it's constructor is automatically invoked during the creation of the object. As a result, the first line of output "Constructor Invoked for Department class" is printed. Similarly, when the e2 object of Employee class is created in the second line of main() i.e (Employee e2), the constructor corresponding to e2 is invoked automatically by the compiler and "Constructor Invoked for Employee class" is printed.
- A destructor is always called in the reverse order as that of a constructor. When the scope of the main function ends, the destructor corresponding to object e2 is invoked first. This leads to printing "Destructor Invoked for Employee class". Lastly, the destructor corresponding to object d1 is called and "Destructor Invoked for Department class" is printed.

Difference Between Constructors and Destructors



Constructor	Destructor
A constructor is called when a new instance of a class is created.	A destructor is called when an already existing instance of a class is destroyed.
It is called each time a class is instantiated.	It is called automatically when an object is deleted from the memory.
Constructors often accept arguments.	Destructors cannot have arguments.
It allocates memory to a newly created object.	It deallocates memory of an object after its deletion.
Multiple constructors can coexist inside a class.	There can always be only one destructor in a class.
It can be overloaded.	It cannot be overloaded.
They have the same name as the class name.	They have the same name as the class name except they are prefixed with a ~ (tilde) symbol.

Conclusion

- The constructor may be defined as the special feature of the programming languages which is used to make the program effective and efficient. It can also be considered as a special type of method that has the same name as that of the class and can be invoked whenever the object of that class is created. Based on the requirement of the constructor once can choose between the default and the parameterized constructor. It has to be understood that it can only be used in the case when there is something that has to be called immediately right after the instance of the class has been created.

THANK YOU