Constructors and Destructors

Constructors in C++

What is constructor?

- The main use of constructors is to initialize objects.
- A constructor is a **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.

How constructors are different from a normal member function?

A constructor is different from normal functions in following ways:

- Constructor has same name as the class itself
- Constructors don't have return type
- A constructor is automatically called when an object is created.
- If we do not specify a constructor, C++ compiler generates a default constructor for us (expects no parameters and has an empty body).

General Syntax of Constructor

- Constructor is a special member function that

  takes the same name as the class name.

- The syntax generally is as given below:

  <class name> { arguments};
- The default constructor for a class X has the form

  X::X()

Types of Constructors

- ✔ Default Constructor
- ✔ Parameterized Constructors
- ✔ Copy constructor

Default Constructor:

- This constructor has no arguments in it.
- Default Constructor is also called as *no argument constructor*.

Example:

```
class creature
{
    private:
    int yearofBirth;
    public:
```

Cont.....

```
creature()
        {
                cout<<“Contructor     called";
                }
        };
int main()
{
   creature obj;
    getch();
    return 0;
   }
```

Parameterized Constructors:

- A parameterized constructor is just one that has parameters specified in it.
- We can pass the arguments to constructor function when object are created.
- A constructor that can take arguments are called

parameterized constructors.

Example:

```
class Creature {
private:
   int yearOfBirth;
public:
  // …
  Creature(int year) {                    //Parameterized Constructor
      yearOfBirth = year;
      }
};
```

Copy Constructor:

- Copy Constructor is used to declare and initialize an object from another object.
- For example the statement:

  abc c2(c1);

  would define the object c2 and at the same time initialize it to the value of c1.
- The processof initializing through a copy constructor is known as *copy initialization.*

Example:

```
class abc
{
      int a, b;
      public:
        abc(int x, int y)
          {
                a = x;
                b =
                y;
                }
                abc::abc(abc &p)
                {
                          a = p.a;
                          b = p.b;
                }
```

Cont……

```
 void showdata()
        {
            cout << a << " " << b << endl;
        }
};

int main()
{
    abc c1(10, 20);
    abc c2(c1);
    c1.showdata();
    c2.showdata();
    getch();
}
```

- **Uses of Parameterized constructor:**
  - It is used to initialize the various data elements of different objects with different values when they are created.
  - It is used to overload constructors and support polymorphism.

- **What is destructor?**
  Destructor is a member function which destructs or deletes an object.
- **When is destructor called?**
  A destructor function is called automatically when the object goes out of scope:
  (1) the function ends
  (2) the program ends
  (3) a block containing local variables ends
  (4) a delete operator is called

How destructors are different from a normal member function?

- Destructors have same name as the class preceded by a tilde (~)
  Destructors don't take any argument and don't return anything

Example:

```
class creature
{
    private:
    int yearofBirth;
    public:
        creature()
        {
            yearofBirth=1970;
            cout<<"constructure called"<<endl;
            }
        ~creature()
        {
             cout<<"destructure called"<<endl;
             }
};
```

Cont……

```
int main()
   {
        cout<<"main start"<<endl;

             creature obj;

        cout<<"main end"<<endl;
        getch();
        return 0;
        }
```

Some important points about destructors and constructor:

- Take the same name as class name.
- Both are Defined in the public.
- Constructor can be overloaded but Destructors cannot be overloaded.
- No return type is specified.

Viva

- **Can there be more than one destructor in a class?**
- **Can we have more than one constructors in a class?**
- **How c++ allows you to create objects?**
- **What happens When we don’t to write a user-defined constructor and destructor?**

Thank You