

Constructors

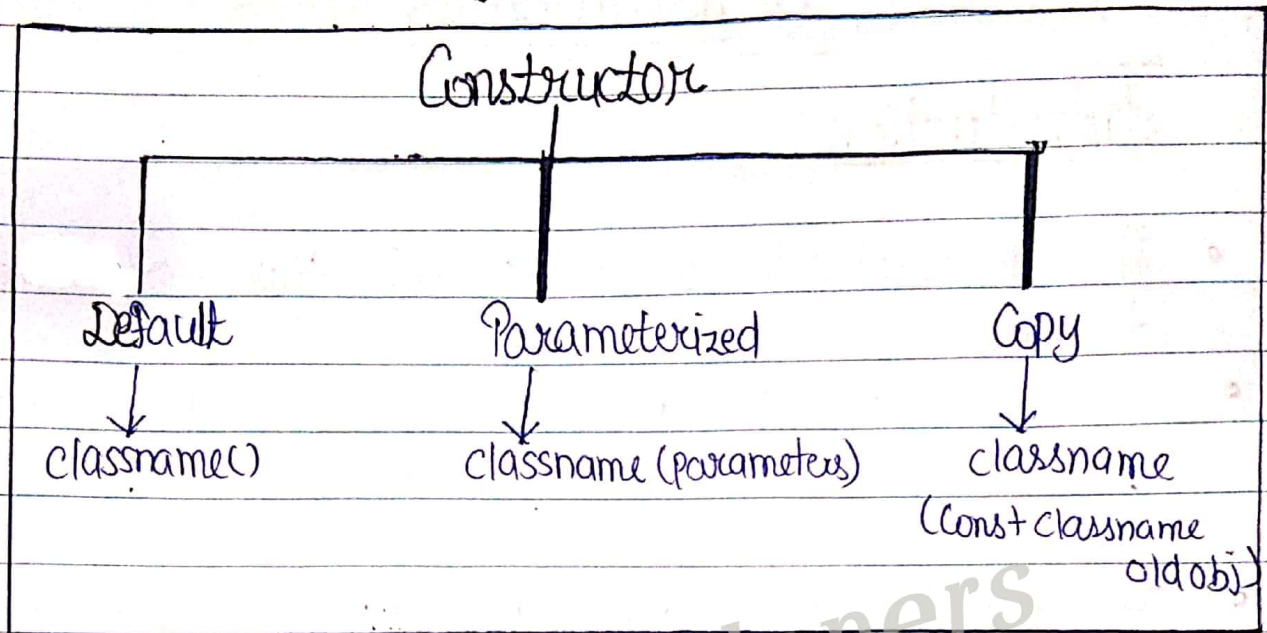
Introduction :-

- A Constructor is a special type of member function of class.
- It is called automatically when an object is created of that class.
- It has same name as that of class name.
- It does not have a return type.
- To create a constructor, use the same name as the class, followed by parentheses ().

For Example :-

```
class abc
{
    public :
    abc()
    {
        cout << "hello";
    }
};
```

Types of Constructor



1 Default Constructors :-

- A Constructor which has no argument is known as default constructor. (Parameters)
- When the class does not contain a constructor, the compiler automatically supplies a default constructor with no arguments.

Program for Default Constructors

```
#include <iostream.h>
#include <conio.h>
```

```

class student
{
    public :
    student()
    {
        cout << "Default Constructor";
    }
};

void main()
{
    clrscr();
    student s;
    getch();
}

```

Output :- Default Constructor

2. Parameterized Constructor

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

Notes by :- jpwebdevelopers.in

WAP to Show Parameterized Constructor:-

```
#include <iostream.h>
#include <conio.h>
class student
{
public:
    int id;
    int rn;
    student (int i, int r)
    {
        id = i;
        rn = r;
    }
    void show()
    {
        cout << id << " " << rn;
    }
};

void main()
{
    clrscr();
    student s = student (1, 101);
    s.show();
    getch();
}
```

Output :- 1 101

3° Copy Constructor

The copy constructor in C++ is used to copy data of one object to another.

It is used to copy the value of one object into another object is called Copy Constructor.

Syntax:-
`classname(classname &ref)`
`{`
`// code;`
`}`

→ A Copy Constructor takes a ref to an object of the same class as an argument.

for e.g.:-
`student s1;` // default constr.
`student s2 = s1;` // Copy constructor is called

Program for Copy Constructor

```
#include <iostream.h>
#include <conio.h>
class student
{
```

```

int id;
int rollno;
public
{
    student (int i, int r)
    {
        id = i;
        rollno = r;
    }
    student (student stu)
    {
        id = stu.id;
        rollno = stu.rollno;
    }
    void show()
    {
        cout << id << " " << rollno;
    }
};

```

```

void main()
{
    student s1 (1, 101);
    student s2 = s1;
    s1.show();
    getch();
}

```

Output:- 1 101

* Multiple Constructors in class or Constructor Overloading in C++

- In C++, we can have more than one constructor in a class with same name, each has a different list of arguments.
- This concept is also known as Constructor Overloading.
- It is quite similar to function overloading.
- Overloaded constructors have the same name (name of the class) but the different number of arguments.
- A constructor is called depending upon the number and type of arguments is passed.

Syntax:-

```
class classname  
{  
    classname()  
}
```

```

    }
    classname (arg1, arg2)
    {
    }
}

```

(Program)
 # WAP to show the concept of Constructor Overloading or multiple Constructor in class.

```

#include <iostream.h>
#include <conio.h>

```

```

class add
{
public:
    int c;
    add () // Constructor with no arguments
    {
        c = 0;
    }
}

```

```

// Constructor with two arguments
add (int a, int b)
{

```

```

        c = a + b;
    }
    void show()
    {
        cout << c << endl;
    }
};

```

```

void main()
{
    clrscr();
    add obj; // Constructor Overloading
    add obj2(10, 20); // with two different
    obj.show(); // constructors of class name

    obj2.show();
    getch();
}

```

Output :- 0 30

Notes by :- jpwebdevelopers.in

* Dynamic Initialization of Objects

- It means Initial value may be provided during runtime.
- Dynamic Initialization of object refers to initializing the objects at run time, the initial value of an object is provided during run time.

Examples: `student (int i, int m)`

```
{  
    id = i;  
    marks = m;  
}  
main() {  
    student(x,y);  
}
```

dynamic initialization
with values x,y.

Program :-

```
#include <iostream.h>  
#include <conio.h>  
class student  
{
```

```

int id;
float marks;
public:
    student (int i, float m)
    {
        id = i;
        marks = m;
    }

    void show()
    {
        cout << id << endl;
        cout << marks;
    }
};

void main()
{
    int a;
    float b;
    cout << "Enter student Id ";
    cin >> a;
    cout << "Enter marks ";
    cin >> b;
    student s (a, b);
    s.show();
    getch();
}

```

Output:-

Enter Id	1
Enter marks	200
	1 200

* Destructors :-

- Destructor is used to destroy the objects that have been created by Constructor.
- A destructor work opposite to constructor. It destructs the objects of classes.
- It can be defined only once in a class.
- It is defined like constructor. It must have same name as class.
But It is prefixed with a tilde sign (~).

Note :- C++ destructor cannot have modifiers.

Syntax

```
class classname  
{
```

```
public :
```

```
~classname()
```

```
{
```

```
}
```

```
};
```

Program of Destructor

```
#include <iostream.h>
#include <conio.h>
class student
{
public:
    student()
    {
        cout << " Constructor called " << endl;
    }
    ~student()
    {
        cout << " Destructor called " << endl;
    }
};

void main()
{
    student s;
    getch();
}
```

Download PDF of Notes
jpwebdevelopers.in