Constructors

A Member function with the same name as its classis called Constructor and it is used to initilize the objects of that class type with a legal value.

A Constructor is a special member function of a class that is called automatically when an object of class is created.

Example :

```
class Student
{
int rollno;
float marks;
public:
student( ) //Constructor
{
rollno=0;
marks=0.0;
}
//other public members
};
```

```
// Implicit Call
Student ob1;

// Explicit Call
Student ob1=student();
```

TYPES OF CONSRUCTORS

1. **Default Constructor:** A constructor that accepts no parameter is called the Default Constructor. If you don't declare a constructor or a destructor, the compiler makes one for you. The default constructor and destructor take no arguments and do nothing.

2. **Parameterized Constructors:** A constructor that accepts parameters for its invocation is known as parameterized Constructors ,also called as Regular Constructors

DESTRUCTORS

✓A destructor is also a member function whose name is the same as the class name but is preceded by tilde(“~”).It is automatically by the compiler when an object is destroyed.

✓Destructors are usually used to deallocate memory and do other cleanup for a class object and its class members when the object is destroyed.

✓A destructor is called for a class object when that object passes out of scope or is explicitly deleted.

DESTRUCTORS

Example :

```
class TEST
{ int Regno,Max,Min,Score;
Public:
TEST( ) // Default Constructor
{ }
TEST (int Pregno,int Pscore) // Parameterized Constructor
{
Regno = Pregno ;Max=100;Max=100;Min=40;Score=Pscore;
}
~ TEST ( ) // Destructor
{ Cout<<”TEST Over”<<endl;}
};
```

The following points apply to constructors and destructors:

✓Constructors and destructors do not have return type, not even void nor can they return values.

✓References and pointers cannot be used on constructors and destructors because their addresses cannot be taken.

✓Constructors cannot be declared with the keyword virtual

✓Constructors and destructors cannot be declared static, const, or volatile.

✓ Unions cannot contain class objects that have constructors or destructors.

The following points apply to constructors and destructors:

✓The compiler automatically calls constructors when defining class objects and calls destructors when class objects go out of scope

✓Derived classes do not inherit constructors or destructors from their base classes, but they do call the constructor and destructor of base classes.

✓The default destructor calls the destructors of the base class and members of the derived class.

The following points apply to Constructors and Destructors:

✓The destructors of base classes and members are called in the reverse order of the completion of their constructor:

✓The destructor for a class object is called before destructors for members and bases are called.
✓You can have only one destructor for a class.
✓Destructors can’t be overloaded.
✓They can not be inherited.
✓No argument can be provided to destructors.

Copy Constructor

A copy constructor is a special constructor in the C++ programming language used to create a new object as a copy of an existing object.

✓A copy constructor is a constructor of the form classname (classname &).The compiler will use the copy constructors whenever you initialize an instance using values of another instance of the same type.

✓Copying of objects is achieved by the use of a copy constructor and a assignment operator.

Copy Constructor Example :

```
class Sample{ int i, j;
public:
Sample(int a, int b) // constructor
{ i=a;j=b;}
Sample (Sample & s) //copy constructor
{ j=s.j ; i=s.j;
cout <<"\n Copy constructor working \n";
}
void print (void)
{cout <<i<< j<< "\n";}
};
```

Note : The argument to a copy constructor is passed by reference, the reason being that when an argument is passed by value, a copy of it is constructed. But the copy constructor is creating a copy of the object for itself, thus ,it calls itself. Again the called copy constructor requires another copy so again it is called. In fact it calls itself again and again until the compiler runs out of the memory .so, in the copy constructor, the argument must be passed by reference.

CALLING COPY CONSTRUCTOR

Sample s2(s1); // s1 is copied to s2. Copy constructor is called.

Sample s3=s1; // s1 is copied to s3. copy constructor called again

When Copy Constructor is called?

- **When an object is passed by value to a function:**

The pass by value method requires a copy of the passed argument to be created for the function to operate upon .Thus to create the copy of the passed object, copy constructor is Invoked If a function with the following prototype :

void cpyfunc(Sample); // Sample is a class then for the following function call

cpyfunc(obj1); // obj1 is an object of Sample type the copy constructor would be invoked to create a copy of the obj1 object for use by cpyfunc().

When Copy Constructor is called?

✓**When a function returns an object :**

When an object is returned by a function the copy constructor is invoked

Sample cpyfunc(); // Sample is a class and it is return type of cpyfunc()

If func cpyfunc() is called by the following statement

obj2 = cpyfunc();

Then the copy constructor would be invoked to create a copy of the value returned by cpyfunc() and its value would be assigned to obj2. The copy constructor creates a temporary object to hold the return value of a function returning an object.

CBSE QUESTION PATTERN

RELATED
TO
CONSTRUCTORS
AND
DESTRUCTORS
2 MARKS
QNO 2 (b)

2 (b) Answer the questions (i) and (ii) after going through the following class : Delhi 2006

```
class Interview
{
int month;
public:
Interview(int y) {month=y;} //Constructor 1
Interview(Interview&t); //Constructor 2
};
```

(i) Create an object, such that it invokes Constructor 1 1

(ii) Write complete definition for Constructor 2 1

(b) Interview Ob1(5);

OR

int N=5; Interview Ob1(N);

(1 mark for proper declaration of Object)

```
Interview(Interview &t)
{
month = t.month;
}
```

(1 mark for writing proper statements inside definition of Constructor 2)

OR

(1 mark for writing the conceptual definition of the copy constructor)

OR

(Only ½ mark for mentioning the term: copy constructor)

Note: Any valid statement in C++ working with t as an object of Interview must be accepted while defining the constructor.

(b) Answer the questions (i) and (ii) after going through the following class : Outside Delhi 2006

```
class Exam
{
int year;
public:
Exam(int y) { year=y;} //Constructor 1
Exam(Exam & t); //Constructor 2
};
```

(i) Create an object, such that it invokes Constructor 1. 1

(ii) Write complete definition for Constructor 2. 1

(i) Exam Ob1(2015);
(ii) Exam (Exam &t)
{year = t.year;}
OR
Copy constructor: It is an overloaded constructor, in which object of the same class is passed as parameter.
(1 mark for writing proper statements inside definition of Constructor 2)
OR
(1 mark for any valid statement in C++ working with t as an object of Exam)
OR
(1 mark for writing the definition/explanation of the concept of copy constructor)
OR
(1/2 mark for mentioning only the term copy constructor)