**CHAPTER 2**

CONSTRUCTOR AND DESTRUCTOR

**Class Constructor**

##A class constructor is a special member function of a class that is executed whenever we create new objects of that class.

## A constructor will have exact same name as the class and it does not have any return type at all, not even void.

## It must be declared as public. Constructors can be very useful for setting initial values for certain member variables.

## It may /may not have any arguments. A constructor that does not take any parameter is known as default constructor.

## If any constructor is not written then the compiler automatically takes a default constructor. If any non-default constructor is written then the default constructor is not added by the compiler.

**Parameterized Constructor**

## In case of parameterized constructor, you can use following syntax to initialize the fields:

Line::Line( double len): length(len)

{

cout <<"Object is being created, length ="<< len << endl;

}

## Above syntax is equal to the following syntax:

Line::Line( double len)

{

cout << "Object is being created,length = “<< len << endl;

length = len;

} ***[ Note by -Jannatul Ferdous Umama(Bristy)***

## If for a class C, you have multiple fields X, Y, Z, etc., to be initialized, then you can use same syntax and separate the fields by comma as follows:

C::C( double a, double b, double c): X(a), Y(b), Z(c) (…………}

**Class Destructor**

\*\* A destructor is used to destroy the object. It will be invoked automatically as soon as the life of the object is finished. A destructor is a special member function of a class that is executed whenever an object of it's class goes out of scope or whenever the delete expression is applied to a pointer to the object of that class.

\*\*A destructor will have exact same name as the class prefixed with a tilde (~) and it can neither return a value nor can it take any parameters.

\*\* Destructor can be very useful for releasing resources before coming out of the program like closing files, releasing memories etc.

\*\* As like constructor, a default destructor is always written if not explicitly mentioned by the programmer.

\*\*Destructors are called exactly in the opposite order of the constructor. Usually destructors are written only when memory is dynamically allocated in constructors.

**Copy Constructor**

\*\* The copy constructor is a constructor which creates an object by initializing it with an object of the same class, which has been created previously. It is used to:

♣ Initialize one object from another of the same type.

♣ Copy an object to pass it as an argument to a function.

♣ Copy an object to return it from a function.

\*\* If a copy constructor is not defined in a class, the compiler itself defines one. If the class has pointer variables and has some dynamic memory allocations, then it is a must to have a copy constructor. The most common form of copy constructor is shown here:

Classname(const classname &obj)

{

//body of constructor

}

Here, obj is a reference to an object that is being used to initialize another object

***[ Note by -Jannatul Ferdous Umama(Bristy)***