Constructors:

Default Constructor :

Either have no parameters or the parameters have default values

Ex: // the parameters have default values

rectangle(int Width = 5, int Length = 5):width(6), length(7)

{

}

//have no parameters

If you don’t define a default constructor (without any arguments) and define another non default constructor . Therefore you have to call the non default constructor when defining an object :  
Ex:-  
  
Class myClass

{

Private:

Int len

Int Wid

Public:  
 myClass (intx, int y)

}

Constructor initialize based on the class member’s order not based on the initialzerlist order

For ex:  
  
Class MyClass

{

Private:

Int a;

Int b;

Public:

MyClass(int Avalue, int Bvalue): b(Bvalue), a(Avalue)

{

}

};

In the above example ‘a’ member will be initialized with Avalue first then ‘b’ will be initilized even though that b comes first in the initialzer list and the opposite is true for the destructor ( destructors will be called based on the reverse order)

// Note: If a member object is not initialized through a member initializer(initializer list), the member object’s default constructor will be called "implicitly".

// Avoid duplicate initlaization! (initialze inside non default constructor without initializer list, and initialze inside default constructor)

destructor

Are mainly used to deallocate memory to avoid memory leaks

Memory leak

1- forget to delete dynamically allocated memory in a destructor

2- forget to use [] when deleting a dynamically allocated array

Dynamic memory allocation

create:

Datatype ptr = new datatype[length]

Ex: int \* arr = new int [10]

Delete:

Delete[] arr;

Notice that if you forget [] when deleting an array >>> memory leak

This keyword

This is a pointer to the class it is used in