Constructors and destructors

Topic 3 term2

Constructors

A constructor is a member function that is used for creating objects and initializing the states of these objects.

Constructor is a member function of the class having the same name as the class. It is mainly used for initializing the members of the class. By default constructors are public.

A constructor has the following characteristics:

1. Its has the same name as the name of the class within which it has been declared
2. It is called automatically
3. It does not have return type
4. It is always declared within the public section of a class
5. It is not inherited
6. A constructor can not be made virtual
7. The address of the memory location occupied by a constructor can not be referenced

Cont..

- Since A constructor is a member function of a class which initializes objects of a class. In C++ ,the constructor is automatically called when the object is created. It is a special member function of the class
- A constructor is different from normal functions in the following ways

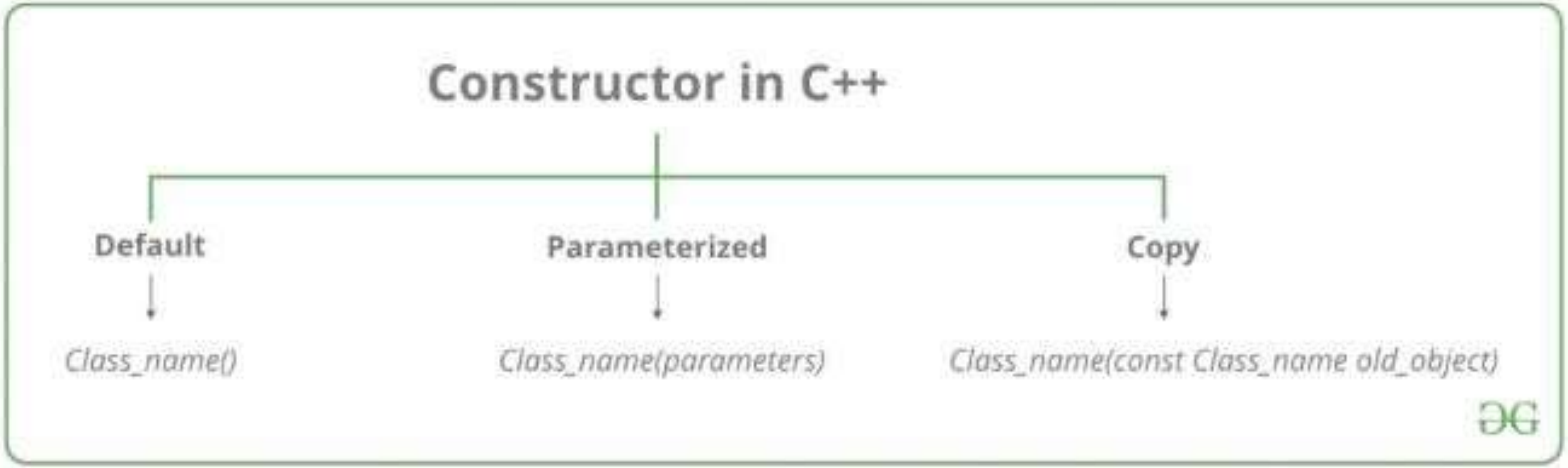
1. Constructor has the same name as the class itself
2. Constructors don’t have a return type
3. A constructor is automatically called when an object is created
4. If one doesn't specify a constructor, the C++ compiler generates a default constructor

Types of Constructors

There are three types of constructors:

1. Default constructors
2. Copy constructors
3. Parameterized constructors

Cont..

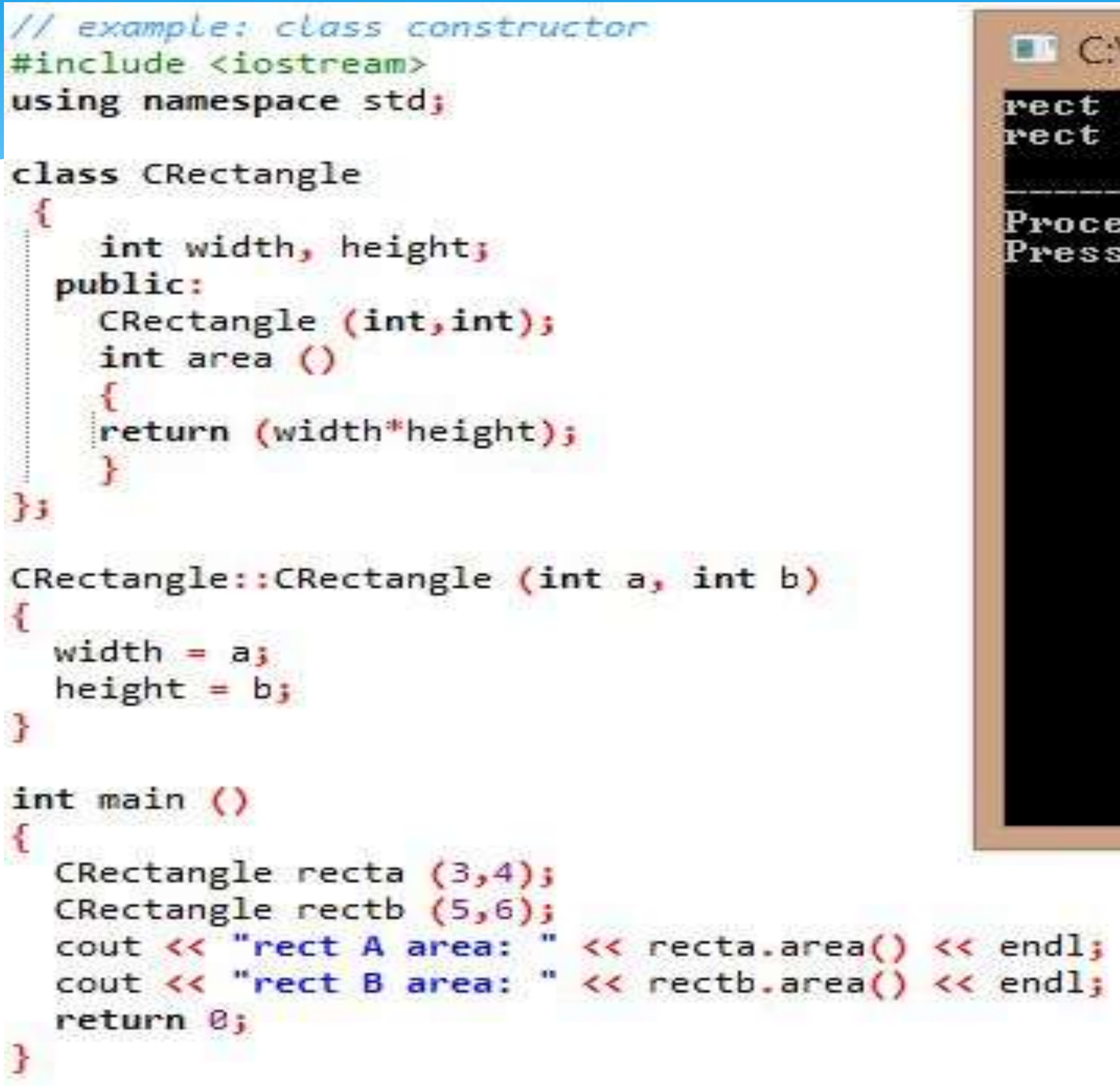

Constructor in C++
Default
Parameterized
Copy
Class_name()
Class_name(parameters)
Class_name(const Class_name old_object)

Declaration of Constructor

A constructor is declared and defined as follows:

```
class Rectangle
{
private:
int length;
int width;
. . .
public:
Rectangle(int l, int w); //declaration of constructor
. . .
};
Rectangle::Rectangle(int l, int w) //definition of constructor
{
length=l;
width=w;
}
```

Example 1

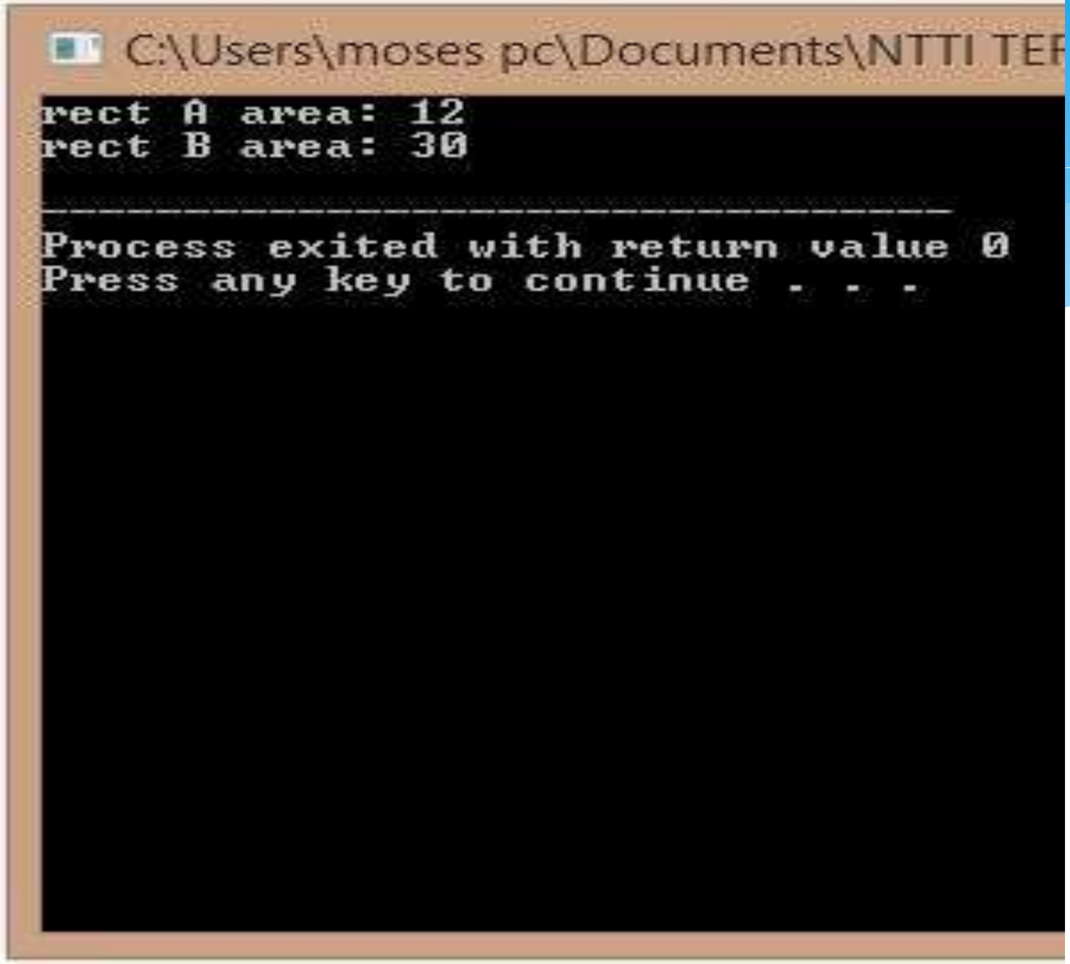


```
// example: class constructor
#include <iostream>
using namespace std;

class CRectangle
 {
    int width, height;
  public:
    CRectangle (int,int);
    int area ()
    {
    return (width*height);
    }
};

CRectangle::CRectangle (int a, int b)
{
  width = a;
  height = b;
}

int main ()
{
  CRectangle recta (3,4);
  CRectangle rectb (5,6);
  cout << "rect A area: " << recta.area() << endl;
  cout << "rect B area: " << rectb.area() << endl;
  return 0;
}
```



C:\Users\moses pc\Documents\NTTI TEF

```
rect A area: 12
rect B area: 30

--------------------------------
Process exited with return value 0
Press any key to continue . . .
```

QUESTION (knec 2021 nov)

Write a C++ program that uses a constructor to create two instances of a class named triangle. The instances should initialize as inst1 (20,10) and inst2(16,12), where the 1st and 2nd values represent the base and height respectively. The class should also contain a member function for determining and outputting the area of the triangle (8 Marks)

QUESTION (knec 2021 nov)

- Write C++ program that uses an overloaded function named calculate to determine the area or volume of rectangular objects based on the number of parameters provided. The program should determine and output area appropriately for object1 (12,8) and object2(6,5,4) . Use function prototypes (7 Marks)

Question 1

Using constructors write a C++ program that calculates the cost of hiring 2 equipments for 6 months

- Offset printer at 16,000 per month
- T-shirt printer at 11,000 per month (6 Marks)

Calling Constructors

Since a constructor is always declared within the public section of a class, it is accessible from external functions(these are functions that are not members of the class that has the class).
For example the main method can call the constructor in the Rectangle class above as shown below.

- Rectangle rect1(6, 5);

This statement is a call to Rectangle constructor in a class called Rectangle. The constructor is also passed two integer values, i.e. 6 and 5.

The constructor will construct the object and 6 and 5 values will be used as the initial state of the object. The object that will be created will be referenced by rect1 variable.

If we now want to pass a message to the object that has been created in Rectangle class, we have to use rect1 to refer to the object as shown below

- rect1.find_area();
- rect1 is our reference to the object created and find_area() is the message that we are passing to our object.
- rect1 refers to the object created.

1. Default Constructor

Default constructor is a constructor that either has no arguments or if there are any, then all of them are default arguments.
A default constructor is a constructor that is not passed any arguments when called or lacks parameters.
A default constructor can be created automatically by the compiler. For example, the default constructor for Rectangle class above is :

```
class Rectangle
{
private:
 . . .
public:
Rectangle();
. . .
};
Rectangle::Rectangle() //definition of default constructor
{
..
 }
```

Cont..

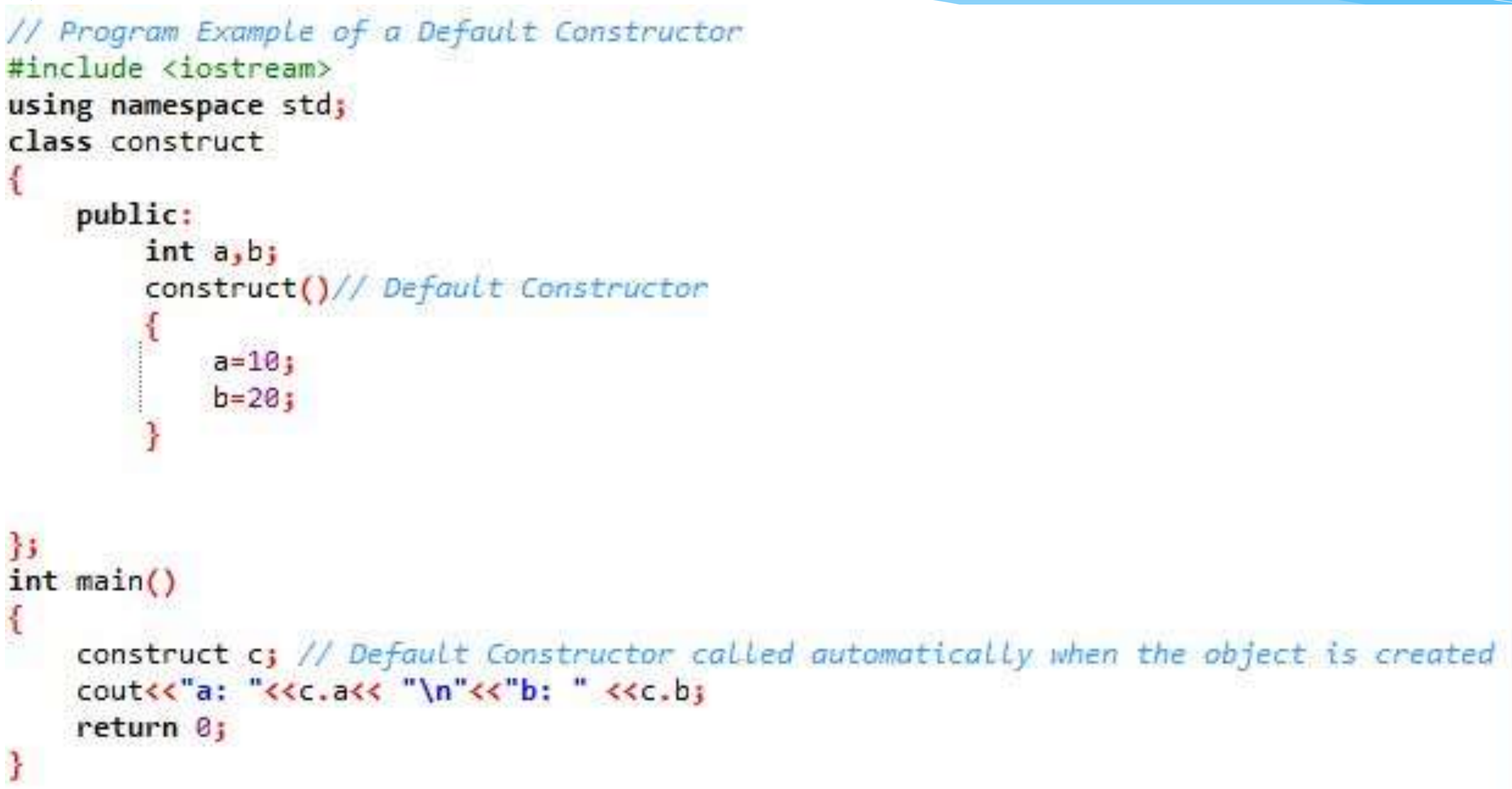
Note that the body of this constructor is empty. Now if we have a statement shown below within the main method

```
int main(void)
{
Rectangle rect1;
. . .
return 0;
}
```

Statement Rectangle rect1; in the above main method is a call to the default. In that case if the Rectangle class do not have the default constructor the compiler will create one automatically.

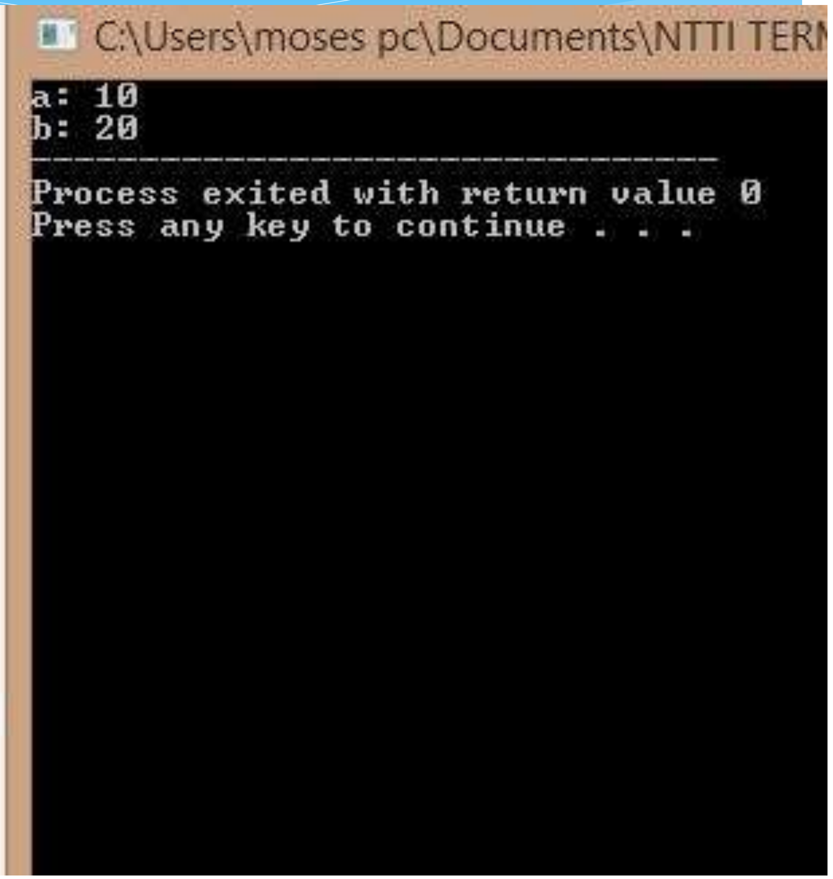
Note : even if we don't define any constructor explicitly, the compiler will automatically provide a default constructor implicitly

Example: default



```
// Program Example of a Default Constructor
#include <iostream>
using namespace std;
class construct
{
    public:
        int a,b;
        construct()// Default Constructor
        {
            a=10;
            b=20;
        }


};
int main()
{
    construct c; // Default Constructor called automatically when the object is created
    cout<<"a: "<<c.a<< "\n"<<"b: " <<c.b;
    return 0;
}
```



C:\Users\moses pc\Documents\NTTI TERN

```
a: 10
b: 20
--------------------------------
Process exited with return value 0
Press any key to continue . . .
```

Question 2

Using default constructor pin , allocate integral value 2556 to pin_1 and 4919 to pin_2 (4 marks)

2.Parameterized constructors

Since it is possible to pass arguments to constructors, typically these arguments help initialize an object when created. To create parameterized constructor, simply add parameters the way you would add to any function, when you define the constructors body , use parameters to initialize the object

Parameterized constructors are constructors that are passed arguments when they are called and must be defined.

Features of parametrized constructors include

1. They are declared in the public section
2. Invoked automatically
3. Do not have a return type
4. Cannot be virtual
5. Cannot be inherited
6. Has the same name as the class

Cont.. example

```
class Rectangle
{
private:
int length;
int width;
. . .
public:
Rectangle(int l, int w);
. . .
};
Rectangle::Rectangle(int l, int w) //definition of a parameterized constructor
 {
length=l; width=w;
}
```

Cont..

There are two ways of passing arguments to parameteized constructors when they are called: by calling the constructor explicitly and by calling the constructor implicitly.

For example

Rectangle rect1= Rectangle(6, 5); //**explicit call**

Rectangle rect1(6, 5); //**implicit call** Implicit call is the one that is normally used.

Difference between implicit and explicit call

Implicit Call :

- Constructors are implicitly called by the compiler when an object of the class is created. This creates an object on a Stack.
- One doesn't use the equal sign(=)

Explicit Call

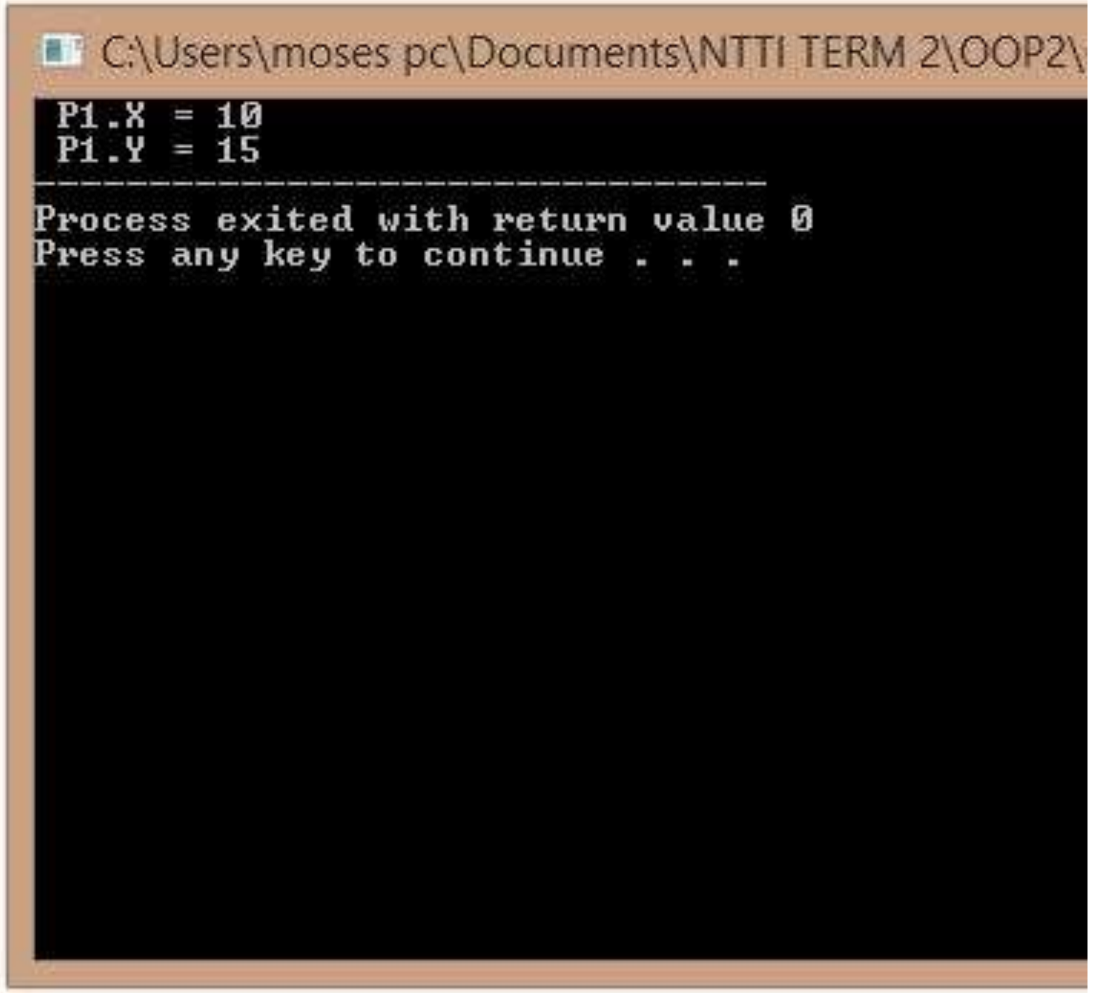
- When the object of a class is created using new, constructors are called explicitly. This usually creates an object on a Stack
- One uses the equal sign(=)

Uses of parameterized constructors

1. Used to initialize various data elements of different objects with different values when they are created
2. Used to overload constructors(having more than one constructor in a class)

Example : parameterized

```cpp
// A program example to illustrate parameterized constructors
#include <iostream>
using namespace std;
class point
{
    private:
        int x, y;
    public:
        point(int x1,int y1)// parameterized constructor
        {
            x=x1;
            y=y1;
        }
        int getx()
        {
            return x;
        }
        int gety()
        {
            return y;
        }
};
int main()
{
    point p1(10,15);// Constructor is called
    cout<<" P1.X = "<<p1.getx()<<"\n P1.Y = "<<p1.gety();// Access values assigned by the constructor
    return 0;
}
```



C:\Users\moses pc\Documents\NTTI TERM 2\OOP2\

```
 P1.X = 10
 P1.Y = 15
--------------------------------
Process exited with return value 0
Press any key to continue . . .
```

Question 3

Using parameterized constructor mark_calc , allocate 40 to cat and 60 to end_term (4 marks)

3. Copy constructors

-Copy constructors are constructors that are used to create and initialize an object using another object of the same class. A new object is created and then assigned a value.. It is used to initialize a newly declared variable from an existing one . A copy constructor is when there is no need to delete an existing value

-A copy constructor is a constructor that accepts an object of the same class as its parameter and copies its data members to the object on the left part of the assignment. It is useful when we need to construct a new object of the same class. E.g.

```
Class-name(cont class-name &obj)
{
//body of construtor
}
```

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

For example

```
Rectangle rect1(6, 5); //a call to a parameterized constructor
Rectangle rect2(rect1); // a call to a copy constructor to create and initialized rect2 object //using rect1 object
```

Other terms in Constructors

- **Conversion Constructor-** This It is a constructor that accepts one argument of a different type. Conversion constructors are mainly used for converting from one type to another.
- **Explicit Constructor-** A conversion constructor is declared with the explicit keyword. The compiler does not use an explicit constructor to implement an implied conversion of types. Its purpose is reserved explicitly for construction.
- **The role of the Static keyword for a class member variable is:** To shares a common memory across all the objects created for the respective class. We need not refer to the static member variable using an object. However, it can be accessed using the class name itself.
- **Static Member Function –** One that can access only the static member variable of the class. Same as the static member variables, a static member function can also be accessed using the class name.

DESTRUCTORS

Destructors

A destructor is a member function that deletes an object. A destructor is used to free a location that is held by an object for use by another object, that object is deleted when the flow of control leaves the scope within which it has been created.

A destructor has the following properties
 Its name is the same as the name of the class

1. Its name should be preceded by a tilde (~) character
2. A destructor is used to delete the memory space
3. It should be declared in the public section of a class
4. It is not passed arguments and it does not have return type not even void
5. It is called implicitly and has no return type
6. A class can have only one destructor
7. A destructor can not be overloaded
8. Does not have any argument or return anything
9. It can be virtual

When is the destructor called

1. When the object goes out of scope
2. When the function ends
3. When the program ends
4. When a block containing local variables ends
5. When a delete operator is called

Cont..

For example

```
Rectangle::~Rectangle()
{
cout<<"An object has been destroyed "<<endl;
}
```

Note that just like other member functions, constructors can have default arguments and they can be overloaded.