NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES

CL 103 - COMPUTER PROGRAMMING LAB

Instructors: Mr. Basit Ali, Ms.Mahrukh Khan, Ms. Ammara Yaseen, Ms. Tooba Ali, Ms. Maham Mobin, Mr. Muhammad Irfan Ayub

Email: basit.jasani@nu.edu.pk , mahrukh.khan@nu.edu.pk, ammara.yaseen@nu.edu.pk, tooba.ali@nu.edu.pk, maham.mobin@nu.edu.pk, muhammad.irfan@nu.edu.pk

Lab # 04

Outline

- Default Constructor
- Parameterized Constructor
- Copy Constructor
- Destructor

CONSTRUCTORS

A constructor is a special function that is a member of a class.

- C++ requires a constructor call for each object that's created, which helps ensure that each object is initialized properly before it's used in a program. The constructor call occurs implicitly when the object is created. This ensures that objects will always have valid data to work on.
- Normally, constructors are declared public.
- Constructors can be overloaded, just like other functions.

DIFFERENCE BETWEEN CONSTRUCTORS AND OTHER FUNCTIONS

CONSTRUCTOR	FUNCTION
Can NOT return any value	Can return value
Constructor name is always class name	Function name can NOT be class name

DEFAULT CONSTRUCTOR

- A constructor without parameters is referred to as a default constructor.
- If a class does not contain a constructor definition, the compiler will create a minimal version of the default constructor as a public member. However, this constructor will not perform initialization. An uninitialized variable typically contains a "garbage" value.

By contrast, if a class contains at least one constructor, a default constructor must be defined explicitly.

See following example:

```
#include<iostream>
#include<string>
using namespace std;

class Employee
{
 int salary;
 string name;

public:
Employee()
   {
      salary = 60000;
            name = "ABCD";
   }
void display()
{
            cout<<"name  :"<<name<<"  "<<"salary  :"<<salary<<endl;
}

};
 int main()
{
   Employee e1;
   e1.display();
    return 0;
   }
```

Parameterized Constructor:

The following example demonstrates how we can pass argument in a constructor.

Example 1:

```
#include<iostream>
#include<string>
using namespace std;
class Employee
{
 double salary;
public:
Employee(double annualSalary)
   {
      salary = annualSalary;
   }

void display()
{
                cout<<"salary  :"<<salary<<endl;
}

};
 int main()
{
   Employee e1(19876);
   e1.display();
   return 0;
   }
```

In the following example, we have used constructor with 2 arguments to initialize class object.

Example 2:

```
#include<iostream>
using namespace std;
class CoOrds
{
public:
 int x, y;
// constructor with two arguments:
public:
CoOrds(int x, int y)
   {
(*this).x = x;
(*this).y = y;
   }
 void display()
   {
cout<<"x = {0}, y = {1} : "<<x<<"  "<<y<<endl;
   }
};

int main()
{
```

```
    CoOrds p1(5, 3);
            p1.display();

}
```

MULTIPLE CONSTRUCTORS

When a class have two or more constructors, the concept is said to be is said to be overloaded. Any class member function may be overloaded, including the constructor. For example, one constructor might take an integer argument, while another constructor takes a double.

Example:

```
#include<iostream>
using namespace std;
class CoOrds
{
public:
 int x, y;
// constructor with two arguments:
public:
CoOrds(int x, int y)
   {
(*this).x = x;
(*this).y = y;
   }
   // constructor with one arguments:
CoOrds(int cor)
   {
      x = cor;
      y = cor;
   }

 void display()
   {
cout<<"x = {0}, y = {1} : "<<x<<"  "<<y<<endl;
   }
};

int main()
{           CoOrds p(15);
            p.display();
            CoOrds p1(5, 3);
            p1.display();

}
```

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. The copy constructor 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.

Syntax:

```
classname (const classname  &obj)
{
   // body of constructor
}
```

If a copy constructor is not defined in a class, the compiler itself defines one.

Example 1:

```
#include <iostream>

using namespace std;

class oneD {
             private:
     int i;
   public:
     int getLength(  )
      {
        return i;
       }
           // simple constructor
     oneD ( int len ){
   cout << "Normal constructor allocating i" << endl;

   i = len;
            }
             // copy constructor
     oneD (oneD &obj)
     {
   cout << "Copy constructor allocating i" << endl;

   i = obj.i; // copy the value
      }
     // destructor

~ oneD ()
  {
   cout << "Freeing memory!" << endl;

}


};

 void display(oneD obj) {
   cout << "Length of line : " << obj.getLength() <<endl;
}

int main() {

   oneD a(10);

  oneD b=a;
  display(a);// This also calls copy constructor
```

```
  display(b);

  return 0;
}
```

If the class has pointer variables and has some dynamic memory allocations, then it is a must to have a copy constructor.

Example 2:

```
#include <iostream>

using namespace std;

class oneD {
          private:
          int *ptr;
         public:

oneD (int len) {
  cout << "Normal constructor allocating ptr" << endl;

  // allocate memory for the pointer;
  ptr = new int;
  *ptr = len;
}


oneD (const oneD &obj) {
  cout << "Copy constructor allocating ptr." << endl;
  ptr = new int;
  *ptr = *obj.ptr; // copy the value
}

~ oneD () {
  cout << "Freeing memory!" << endl;
  delete ptr;
}

int getLength( ) {
  return *ptr;
}
};
void display(oneD obj) {
  cout << "Length of line : " << obj.getLength() <<endl;
}


int main() {
  oneD a(10);
  oneD b=a;
  display(a);
  display(b);

  return 0;
}
```

DESTRUCTOR

A destructor is a special function that is a member of a class.

- A destructor is a special member function that is called when the lifetime of an object ends.
- The purpose of the destructor is to free the resources that the object may have acquired during its lifetime.
- Normally, destructors are declared public.
- Destructor have the same name as class name followed by a ~.

Called when an object is destroyed.

Example:

```
class Example
 {
        public:
        Example()
{
cout<< “Welcome”;
}

        ~Example()
        {

                cout<<”Good Bye”;
        }
};
```

EXERCISE

Question # 1

Write a program using class to process shopping list for a departmental store. The list includes details such as code no, price, qty and total and perform operations like adding and deleting items from the list. The program will also print the total value of an order. You are supposed to add a constructor, destructor and other necessary functions to perform the said tasks.

Question # 2

A phone number, such as (021) 38768214, can be thought of as having three parts: the area code (021) the exchange (3876) and the number (8214). Write a program that uses a class Phone to store these three parts of a phone number in specific attributes. Add a constructor that accept a number and separate these elements from that number. Write a display function that display the details of the number.
Sample Program output:

```
Please enter Your No: 02134567893
Your Area code is: 021
Your Exchange Code is: 3456
Your Consumer No is: 7893
```

QUESTION # 3

Create a class distance that stores distance in feet and inches. Add a constructor that initializes the object with default values. There must be a function that ask user to enter distance in meters and stores accordingly. Add two functions to display the distance in meters and in feet. Add a destructor that will notify the user when an object is killed.

QUESTION # 4

Create a class Sales with 3 private variables SaleID of type integer, ItemName of type string , and Quantity of type integer.
Part (a) Use a default constructor to initialize all variables with any values.
Part (b) Use a constructor to take user input in all variables to display data.
Part (c) Use a parameterized constructor to initialize the variables with values of your choice.
Part (d) Use copy constructor to copy the quantity of previously created object to current one.

QUESTION# 5

Write a program in which a class named student has member variables name, roll_no, semester and section. Use a parameterized constructor to initialize the variables with your name, roll no, semester and section. Print all data calling some public method.