National University of Computer & Emerging Sciences, Karachi Computer Science Department

Spring 2023, Lab Manual - 04

Course Code: CL-1004	Course : Object Oriented Programming Lab

Contents

Class:

A class is a user-defined data type. It holds its own data members and member functions, which can be accessed and used by creating an instance of that class. A class is like a blueprint for an object.

Types of Classes:

- Concrete Classes
- Generalized classes
- Specialized classes

Concrete Classes:

A concrete class is a class that has an implementation for all of its methods. They cannot have any unimplemented methods.

Example:

```
class Concrete {
private:
   string info;
public:
   Concrete(string s) : info(s) { }
   void printContent() {
      cout << "Concrete Object Information\n" << info << endl;
   }};
```

```
int main()
{
string s;
s = "Object Creation";
   Concrete c(s);
   c.printContent();
}
```

Generalized Classes:

A class which tells the main features but not the specific details. The classes situated at the top of the inheritance hierarchy can be said as General.

Example:

"Car" can be considered generalized class.

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

class Car {
   public:
      int price;
      int year;
      string make;
      string model;

      Car(int p, int y, string m, string mo) {
         price = p;
         year = y;
         make = m;
         model = mo;
      }

      void displayInformation() {
         cout << "Price: " << price << endl;
         cout << "Year: " << year << endl;
         cout << "Make: " << make << endl;
         cout << "Model: " << model << endl;
      }
};
```

Specialized Classes:

A class which is very particular and states the specific details. The classes situated at the bottom of the inheritance hierarchy can be said as Specific.

Example:

In the code provided, the class "ToyotaCars" is an example of a specialized class. This class represents a specific type of car, namely Toyota cars, and provides specific information about them, such as the model, year, and color. The class is defined with private variables to store this information, and public methods to display it.

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

class ToyotaCars {
   private:
      string model;
      int year;
      string color;
   public:
      ToyotaCars(string model, int year, string color)
      {
         this->model = model;
         this->year = year;
         this->color = color;
      }
      void displayInfo()
      {
         cout<<"Model: "<<model<<endl;
         cout<<"Year: "<<year<<endl;
         cout<<"Color: "<<color<<endl;
      }
};
```

Introduction to Constructor

- **Constructor** is the special type of member function in C++ classes. It is automatically invoked when an object is being created. It is special because its name is same as the class name.
- **To initialize data member of class:** In the constructor member function (which the programmer will declare), we can initialize the default vales to the data members and they can be used further for processing.
- **To allocate memory for data member:** Constructor is also used to declare run time memory (dynamic memory for the data members).
- Constructor has the same name as the class name. It is case sensitive.
- Constructor does not have return type.
- We can overload constructor; it means we can create more than one constructor of class.
- It must be public type.

Types of Constructors

- **Default Constructors:** Default constructor is the constructor, which does not take any argument. It has no parameters.
- **Null constructors:** Null constructors in C++ are a special type of constructor that does nothing. The compiler knows that there is no code to execute, so it will not generate any executable code for the constructor.
- **Parameterized Constructors:** It is possible to pass arguments to constructors. Typically, these arguments help initialize an object when it is created. To create a parameterized constructor, simply add parameters to it the way you would to any other function. When you define the constructor's body, use the parameters to initialize the object.
- **Copy Constructor:** A copy constructor is a member function, which initializes an object using another object of the same class. The copy constructor in C++ is used to copy data of one object to another.

Default Constructor Example:

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

class construct
{
public:
    int a, b;
```

```
    construct()
    {
        a = 10;
        b = 20;
    }
};

int main()
{

    construct c;
    cout << "a: " << c.a << endl
        << "b: " << c.b;
    return 1;
}
```

Parameterized Constructor Example:

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

class Point
{
private:
    int x, y;

public:

    Point(int x1, int y1)
    {
        x = x1;
        y = y1;
    }

    int getX()
    {
        return x;
    }
```

```
    int getY()
    {
        return y;
    }
};

int main()
{

    Point p1(10, 15);

    cout << "p1.x = " << p1.getX() << ", p1.y = " << p1.getY();

    return 0;
}
```

Copy Constructor Example:

```
#include<iostream>
#include<conio.h>

using namespace std;

class Example {

    int a, b;
public:

    Example(int x, int y) {

        a = x;
        b = y;
        cout << "\nIm Constructor";
    }
Example(const Example& obj) {

        a = obj.a;
        b = obj.b;
        cout << "\nIm Copy Constructor";
```

```
    }

    void Display() {
        cout << "\nValues :" << a << "\t" << b;
    }
};
int main() {


    Example Object(10, 20);


    Example Object2(Object);


    Example Object3 = Object;

    Object.Display();
    Object2.Display();
    Object3.Display();
    return 0;
}
```

Constructor Overloading:

In C++, We can have more than one constructor in a class with same name, as long as each has a different list of arguments. This concept is known as Constructor Overloading. Overloaded constructors have the same name (name of the class) but the different number of arguments. Depending upon the number and type of arguments passed, the corresponding constructor is called

Example:

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

class Student
{
    private:
        string name;
        int age;
        string address;
        string department;
```

```
public:
    // Default constructor
    Student()
    {
        name = "";
        age = 0;
        address = "";
        department = "";
    }

    // Overloaded constructor with name and age parameters
    Student(string studentName, int studentAge)
    {
        name = studentName;
        age = studentAge;
        address = "";
        department = "";
    }

    // Overloaded constructor with name, age, and address parameters
    Student(string studentName, int studentAge, string studentAddress)
    {
        name = studentName;
        age = studentAge;
        address = studentAddress;
        department = "";
    }

    // Overloaded constructor with all parameters
    Student(string studentName, int studentAge, string studentAddress, string studentDepartment)
    {
        name = studentName;
        age = studentAge;
        address = studentAddress;
        department = studentDepartment;
    }

    // Accessor functions to access private data members
    string getName()
    {
        return name;
    }
```

```
    int getAge()
    {
        return age;
    }
    string getAddress()
    {
        return address;
    }
    string getDepartment()
    {
        return department;
    }
};

int main()
{
    // Creating objects using different constructors
    Student student1;
    Student student2("Ali Hasan", 20);
    Student student3("Junaid Khan", 22, "Gulshan, Khi");
    Student student4("Ayesha Usman", 23, "Saddar, Khi", "Computer Science");

    // Printing details of each student
    cout << "Student 1 Details:" << endl;
    cout << "Name: " << student1.getName() << endl;
    cout << "Age: " << student1.getAge() << endl;
    cout << "Address: " << student1.getAddress() << endl;
    cout << "Department: " << student1.getDepartment() << endl;
    cout << endl;

    cout << "Student 2 Details:" << endl;
    cout << "Name: " << student2.getName() << endl;
    cout << "Age: " << student2.getAge() << endl;
    cout << "Address: " << student2.getAddress() << endl;
    cout << "Department: " << student2.getDepartment() << endl;
    cout << endl;

    cout << "Student 3 Details:" << endl;
    cout << "Name: " << student3.getName() << endl;
    cout << "Age: " << student3.getAge() << endl;
    cout << "Address: " << student3.getAddress() << endl;
    cout << "Department: " << student3.getDepartment() << endl;
    cout << endl;
```

```
    cout << "Student 4 Details:" << endl;

    cout << "Name: " << student4.getName() << endl;
    cout << "Age: " << student4.getAge() << endl;
    cout << "Address: " << student4.getAddress() << endl;
    cout << "Department: " << student4.getDepartment() << endl;
    cout << endl;
}
```

Initializer List:

Initializer List is used in initializing the data members of a class. The list of members to be initialized is indicated with constructor as a comma-separated list followed by a colon.

Syntax:

```
Constructorname(datatype value1, datatype value2) : datamember(value1), datamember(value2)
{
   ...
}
```

Example:

```
class Point {
        private:
                int x;
                int y;
        public:
                Point(int i = 0, int j = 0) :x(i), y(j) {}

                int getX() const { return x; }
                int getY() const { return y; }
        };

        int main() {
                Point t1(10, 15);
                cout << "x = " << t1.getX() << ", ";
                cout << "y = " << t1.getY();
return 0;
}
```

Exercises

Task_01:

Create a class named as “StudentRecord” and ask for the following field values:

1). Name of the student
2). ID of the student (limit to 4 integers)
3). The current semester of the student

You need to begin the program by providing default values to the fields which must be “NaN” for the name, 0 and 0 respectively for the other two. Make another overload which asks for the values of at least 5 students and displays them accordingly.

Task_02:

Create a C++ program that requests a person's NIC number in order to register them in a government housing scheme option. If no NIC number is provided, the message "The person does not want the scheme" should be displayed. If the object is created, however, a String stating "The person on the registered NIC has been included in the government scheme" should be provided. Destroy all the objects that have been created

Task_03:

Create a class called “ShapeCalculator”, the concerned class must have fields which asks for the length and breadth of various shapes (you can select the shapes as per your wish, min 3 shapes), the goal is to print the area for all the shapes, now you must perform theses tasks following the guidelines as given:

1- A default constructor that provides default values.
2- Two accessors and mutators that set and receive the values of length and breadth respectively.
3- An overload that calculates the area.
4- A display function that displays the area of the first shape.
4- A copy constructor that displays the area of the second and third shape.

Task_04:

Create definitions for two iterations of a function that is overloaded. The first iteration of this function, sum(), accepts an int array as an input and returns the sum of all its entries. The second iteration of sum() accepts an int array and a character ('E' or 'O') as two inputs. If the passed character is "E," it returns the array's sum of even items, and if it is "O," it returns the array's total of odd elements. It returns 0 for any other character (zero).

Task_05:

Create the "TollCars" class in C++. A double field is used to keep the total amount of money received, while an unsigned int field is used to store the total number of vehicles. With the use of a default constructor, they are both initialized to 0. Another constructor increases the number of automobiles and raises the total amount of money by 0.50. The third constructor must increase the number of cars while keeping the amount of money the same. A member function is used to display the two sums.
According to the program, a paying car should be counted by pushing one key, and a non-paying car should be counted by hitting a separate key. The application must print out the totals for both the number of cars and the amount of money received.

Task_06:

FAST needs a small program that calculates the testcores of any particular class, your task is to constructor such a class and it should have a pointer member variable to hold test scores. The class should have a constructor that allows the user to specify the number of test scores.

Provide a copy constructor.
Include a destructor.
Include a member function that returns the average of the scores.
Include accessor and mutator functions.

Write a different application to create at least two objects to demonstrate the class. One object should be defined once the program asks the user how many test scores they have. The mutator function should be used by the program to ask the user for the values before saving them in the object. The program should then execute the accessor member function to display the scores stored in the object.

When you instantiate the second object, initialize it to the first object (to test the copy constructor).