National University of Computer & Emerging Sciences, Karachi
Computer Science Department
Spring 2024, Lab Manual – 06

Course Code: CL-1004	**Course : Object Oriented Programming Lab**
Instructor(s) :	**Muhammad Ali**

Contents:

- Working with static functions
- Constant
- Constant Functions
- Inline Functions
- Member initializer list
- Has - a relationship

Working with Static Functions:

A class's static member function is a function that has been designated as static and as a result a static member function is unrelated to any class object.

Frequently, shared data between all objects in a class is stored in static members.
For instance, you could use a static data member as a counter to keep track of the number of newly created objects of a particular class type. To track the total number of objects, this static data member can be increased each time an item is created.

The scope resolution operator also allows calling static member functions using the class name. A static member function can call static member functions inside or outside of the class, as well as static data members although the scope of static member functions is restricted to the class and they are unable to access the current object pointer.

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

class Student
{
   private:
   static int ID;
   static int SemesterNumber;
   static float CGPA ;

   public:

   static void DisplayDetails()
   {
      cout << "The ID of the student is: " << ID << endl;
     cout << "The current semester of the student is: " << SemesterNumber  <<
endl;
      cout << "The CGPA of the student is: " << CGPA << endl;
   }
};

// initialize the static data members

int Student :: ID = 101;
int Student :: SemesterNumber = 3;
float Student :: CGPA = 3.56;


int main()
{

   Student stu;
   Student::DisplayDetails();

cout << "\nStatic member function can be called by object or by
class reference: \n"  << endl;
   return 0;
}
```

Constant:

const Keyword in C++

Constant is something that doesn't change. In C language and C++ we use the keyword const to make program elements constant. **const** keyword can be used in many contexts in a C++ program. It can be used with:

1. Variables
2. Function arguments
3. Class Data members
4. Class Member functions
5. Objects

Constant Variables in C++

```
int main
{
    const int i = 10;
    const int j = i + 10;     // works fine
    i++;    // this leads to Compile time error
}
```

If you make any variable as constant, using const keyword, you cannot change its value. Also, the constant variables must be initialized while they are declared.

In the above code we have made i as constant, hence if we try to change its value, we will get compile time error. Though we can use it for substitution for other variables.

Constant Functions in C++

Functions declared constant are those that are prohibited from altering the data members of their class. The function prototype and the function definition header both receive the keyword "const" to designate a member function as a constant.

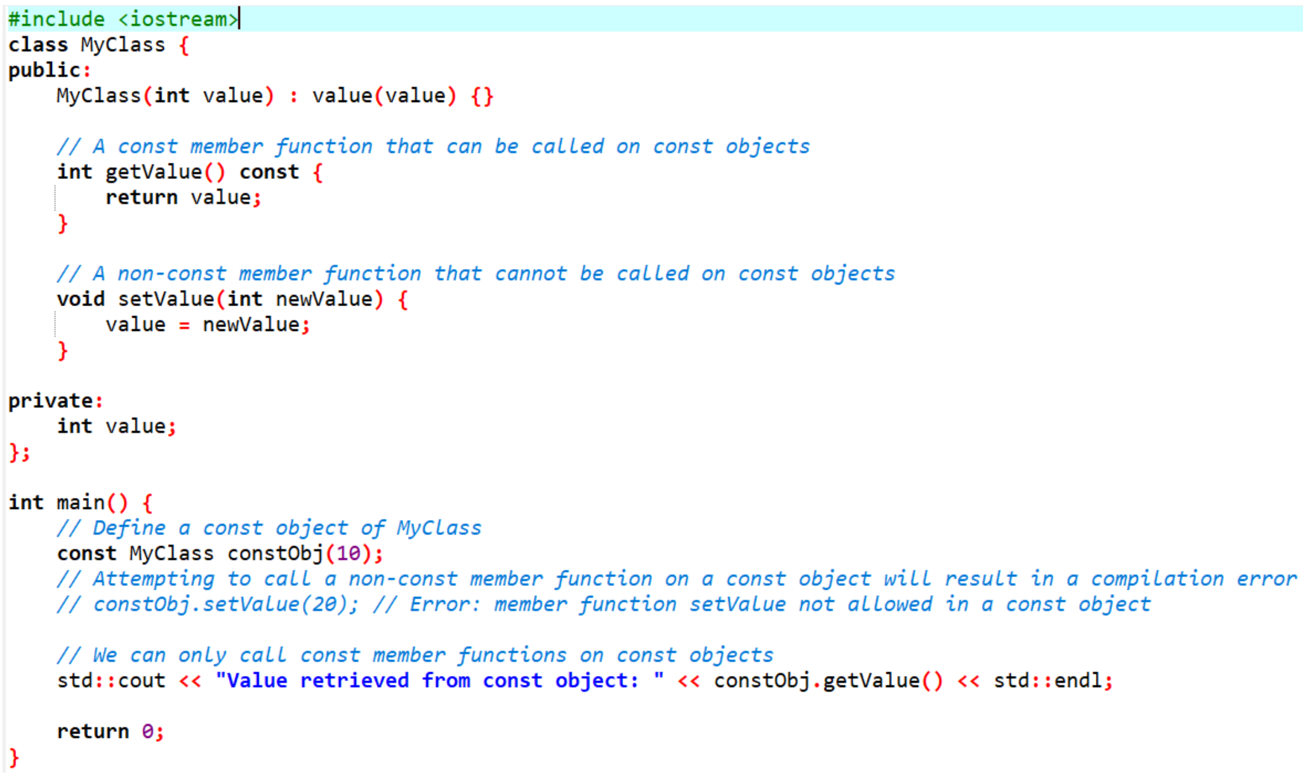
Class objects can also be defined as const, just like member functions and member function arguments. Because a const object cannot be changed, only const member functions may be called on it because they guarantee that the object won't be changed.

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

class ConstTest
{
      double var;
      public:

             void set_var(int var)
             {
                 this->var=var;
             }

              int get_var() const   //The function is now constant
              {
                  ++var;    // Error (cannot change the member now)
                  return var;
              }

         };


main()
{
ConstTest objC;
objC.set_var(80);
cout<<endl<<objC.get_var();

return 0;
}
```

Constant objects in C++

In C++, you can declare objects of a class as const. This means that once initialized, the object cannot be modified through any of its non-const member functions. Only const member functions can be invoked on const objects.

By adding the const keyword before the object definition, a const object can be made. A compile-time error occurs whenever a const object's data member is attempted to be changed.



```
#include <iostream>
class MyClass {
public:
    MyClass(int value) : value(value) {}

    // A const member function that can be called on const objects
    int getValue() const {
        return value;
    }

    // A non-const member function that cannot be called on const objects
    void setValue(int newValue) {
        value = newValue;
    }

private:
    int value;
};

int main() {
    // Define a const object of MyClass
    const MyClass constObj(10);
    // Attempting to call a non-const member function on a const object will result in a compilation error
    // constObj.setValue(20); // Error: member function setValue not allowed in a const object

    // We can only call const member functions on const objects
    std::cout << "Value retrieved from const object: " << constObj.getValue() << std::endl;

    return 0;
}
```

HAS-A RELATIONSHIP

In Object-Oriented Programming (OOP), a has-a relationship is a type of association between classes where one class has a member variable of another class type. This relationship is also known as composition or aggregation.

Association

A type of relationship between classes in C++ called association explains how objects of one class are connected to objects of another class. An instance of association is one in which one class makes use of or interacts in some manner with another class.

In C++, there are primarily two kinds of associations:

Aggregation

Aggregation is a type of association where objects from different classes are referenced by one another but can still live separately.

Due to the fact that it has a pointer to an Engine object, the Car class in this example has an aggregate relationship with the Engine class. Several Car objects may share the same Engine object, and the Engine object may exist separately from the Car object.

```
#include <iostream>
using namespace std;
class Address {
     public:
    string addressLine, city, state;
      Address(string addressLine, string city, string state)
     {
         this->addressLine = addressLine;
         this->city = city;
         this->state = state;
     }
 };
 class Employee
     {
         private:
         Address* address;  //Employee HAS-A Address
         public:
         int id;
         string name;
         Employee(int id, string name, Address* address)
        {
            this->id = id;
            this->name = name;
            this->address = address;
        }
      void display()
        {
            cout<<id <<" "<<name<< " "<<
              address->addressLine<< " "<< address->city<< " "<<address->state<<endl;
        }
    };
 int main(void) {
     Address a1= Address("C-146, Sec-15","Noida","UP");
     Employee e1 = Employee(101,"Nakul",&a1);
             e1.display();
    return 0;
 }
```

Composition

This form of association involves the composition of two classes, in which case the containing object determines the lifetime of the composed object. In other terms, the composed object is also destroyed when the containing object is.

```
            class Wheel {
                // implementation of Wheel class
            };

            class Car {
                Wheel wheels[4]; // array of 4 Wheel objects
            public:
                // constructor
                Car() {
                    // create 4 Wheel objects and store them in the
     wheels array
                    for (int i = 0; i < 4; i++) {
                        wheels[i] = Wheel();
                    }
                }

                // other member functions of Car class
            };
```

In this example, the Car class contains an array of four Wheel objects, each of which is created using the default constructor of the Wheel class. The wheels cannot exist on their own, when the object of a car is destroyed so is the Wheel object.

Examples:

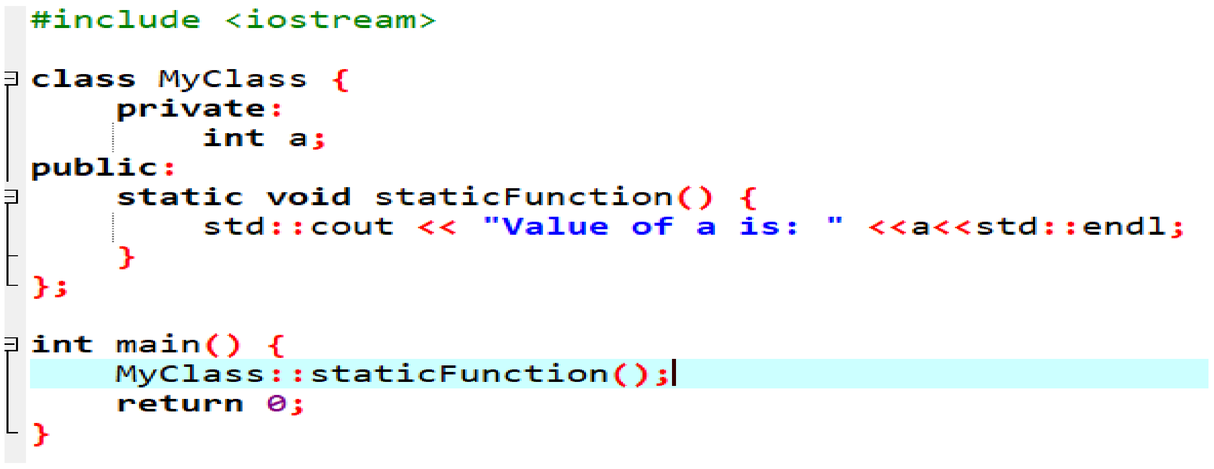
Example-01:

```cpp
#include <iostream>
class Test
{
public:
    void fun();
};
static void Test::fun()
{
    std::cout<<"fun() is static\n";
}
int main()
{
    Test::fun();
    return 0;
}
```

What will be the output of the above code and also provide a reason to your answer in the code comment (if the code is incorrect write a correct version of it as a replacement)

(A) fun() is static
(B) Empty Screen
(C) Compiler Error

Example-02:



```cpp
#include <iostream>

class MyClass {
    private:
        int a;
public:
    static void staticFunction() {
        std::cout << "Value of a is: " <<a<<std::endl;
    }
};

int main() {
    MyClass::staticFunction();
    return 0;
}
```

What will be the output the above code and also provide a reason to your answer? in the code comment (if the code is incorrect write a correct version of it as a replacement)

(A) Value of a is: 0
(B) Empty Screen
(C) Compiler Error

Lab-06 Tasks

Task 1:

Consider a banking system where you have a BankAccount class representing individual accounts. Each BankAccount object contains a pointer to an array of transactions representing the account history.

For that, Create class **BankAccount** which contains the private member variables **int accountId, double balance** to store account information., **int* transactionHistory** is a pointer to an array storing transaction history and **int numTransactions** used to stores the number of transactions.

Also class have following Public Member Functions:

Constructor that initializes account information and allocates memory for the transaction history, copying provided transactions.

Copy constructor that creates a new BankAccount object as a copy of another BankAccount object, including its transaction history.

Destructor that deallocates memory allocated for the transaction history.

display() Function: function to output the account details including the account ID, balance, and transaction history.

updateTransactionHistory() Function:

This function updates the transaction history of the BankAccount object.

It deletes the existing transaction history, allocates memory for the new transaction history, and copies the new transaction data into the allocated memory.

Main Function:

Create a BankAccount object with initial account details and transaction history.

Create a copy of the original account.

Displays the details of both the original and copied accounts using the display() function.

Now , call updateTransactionHistory() Function to update the transaction history of the original account to observe the changes.

Now call the display function to see the changes.

Task-02

Consider that You're working on Images to handle images in your graphics application. Each Image object contains a pointer to image data stored in memory. You are asked to create Image Class: The Image class represents an image with width, height, and image data as private. The data member represents the image data stored in memory. It's a pointer to a block of memory where the actual pixel values of the image are stored.

Create Constructor: Constructor that initializes the width, height, and data pointer. It dynamically allocates memory for the image data and copies the provided data. By copying the provided image data, the Image class creates an independent copy of the data. This means that modifications made to the original image data outside the class won't affect the internal representation of the image within the Image object.

Create Copy constructor that creates a new Image object as a copy of another Image object. It allocates memory for the image data and copies the data from the original image.

Class must have Destructor that deallocates the memory allocated for the image data.

Create void display(): Displays the image data.

Create the updateData function: that takes original image data and updates those values which are 0 are less than 0 with any random value between the range of 1<=255

Main Function:

Create an Image object with dimensions 2x3 and initialize it with sample data.

Creates a copy of the original image.

Displays the data of both the original and copied images using the display() function.

After that call the updateData function and then call the display() function again to verify that changes are made in both original and copied data.

Task-03

You have been hired to develop a program for a hair salon that keeps track of appointments made, total earnings, and the average cost per appointment using static members and functions.

Your task is to define a class that represents an appointment at the salon. The class should have private data members that store information about the appointment, and static data members that keep track of the total number of appointments made and the total earnings from all appointments.

Define a constructor for the appointment class that takes arguments for the appointment details and updates the static data members accordingly. Additionally, you should define public member functions that allow access to the appointment details.

Finally, you should define a static function that calculates and returns the average cost of all the appointments made. This function should use the static data members to perform the calculation.

In the main function, you should create several instances of the appointment class with different appointment details, and then call the static function to calculate the average cost of all the appointments made. This will allow you to verify that the program is correctly keeping track of appointments and earnings

Demonstrate the functions using main function

Task-04

Consider a School Management System. You are required to create two classes, Student and Course. Each student can enroll in multiple courses. The Course class should have the following properties: course code (string), course name (string), and course credit hours (integer). The Student class should have the following properties: student ID (string), student name (string), and a list of enrolled courses (an array of Course objects).

The Student class should have the following member functions:

A constructor that initializes the student ID and name.

A function named "enroll" that takes a Course object as input and adds it to the list of enrolled courses for the student.

A function named "drop" that takes a Course object as input and removes it from the list of enrolled courses for the student.

A function named "getTotalCreditHours" that returns the total number of credit hours for all courses the student is currently enrolled in.

A function named "printEnrolledCourses" that prints out the course code, name, and credit hours for all courses the student is currently enrolled in.