Object Oriented Programming (CT-260)
Lab 04

Introduction to Use of Static and Constant Keywords for Data, Member Functions and Objects of a Class

Objectives

The objective of this lab is to familiarize students with use of constant and static keywords in Object-oriented paradigm. By the end of this lab, students will be able to understand the concepts of static and constant data, member functions and objects.

- **Tools Required**

DevC++ IDE / Visual Studio / Visual Code

Course Coordinator –
Course Instructor –
Lab Instructor –
Prepared By Department of Computer Science and Information Technology
NED University of Engineering and Technology

this POINTER

You can access class members using by default, the compiler provides each member function of a class with an implicit parameter that points to the object through which the member function is called. The implicit parameter is this pointer.

```
#include<iostream>
using namespace std;
class example{
private:
        int x;
public:
        void set(int x){
                (*this).x = x;
        }
        int get( ){
                return x;
        }
        void printAddressAndValue( ){
                cout << "The address is"<<this<<"and the value is"<<(*this).x<<endl;
        }
};
```

```
The adrress is 0x23fe40 and the value is 5
The adrress is 0x23fe30 and the value is 6
```

One copy of each member function in a class is stored no matter how many objects exist, and each instance of a class uses the same function code. When you call a member function, it knows which object to use because you use the object's name. The address of the correct object is stored in this pointer and automatically passed to the function.

Within any member function, you can explicitly use this pointer to access the object's data fields. You can use the C++ pointer-to-member operator, which looks like an arrow (->).

```
#include<iostream>
using namespace std;
class Test{
private:
        int x;
        int y;
public:
        Test(int x = 0, int y = 0){
                this->x = x;
                this->y = y;
        }
        void print( ){
                cout<<"x="<<x<<" y="<< y <<endl;
        }
};
int main( ){
        Test obj1(10, 20);
        obj1.print( );
}
```

```
x = 10 y = 20

--------------------------------
Process exited after 0.01273 seconds with return value 0
Press any key to continue . . .
```

Member Initialization 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. Following is an example that uses the initializer list to initialize x and y of Point class.

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

Uses

- For initialization of non-static const data members.
- For initialization of reference members.
- For initialization of member objects which do not have a default constructor.
- For initialization of base class members.
- When the constructor's parameter name is same as the data member.
- For Performance reasons

Constant Keyword

If there is a need to initialize some data members of an object when it is created and cannot be changed afterward, use const keyword with data members.

CONSTANT DATA MEMBERS

Those members of a class are made constant which needs not to be changed after its initialization. The data member needs to be initialized constant when its created.

```
#include <iostream>
using namespace std;
class Students{
private:
    string name;
```

```
    const int rollno;
    float cgpa;
public:
    Students (int rno): rollno(rno){ }
    void set(string sname, float cg){
          name = sname;
          cgpa = cg;
     }
    void print( ){
         cout<<"Name: "<<name<<", Roll #"<<rollno<<", CGPA, "<<cgpa<<endl;
    }
};
int main( ){
       Students s(12);
       s.set("Ahmad",3.67);
      s.print( );
}
```

CONSTANT MEMBER FUNCTIONS

- Constant member function is the function that cannot modify the data members.
- To declare a constant member function, write the const keyword after the closing parenthesis of the parameter list. If there is separate declaration and definition, then the const keyword is required in both the declaration and the definition.
- Constant member functions are used, so that accidental changes to objects can be
- avoided. A constant member function can be applied to a non-constobject.
- keyword const can't be used for constructors and destructors because the purpose of a constructor is to initialize data members, so it must change the object. Same goes for destructors.

```
#include<iostream>
using namespace std;
class test{
private:
    int a;
public:
    int nonconstFucntion(int a){
         cout<<"Non Constant Function is called"<<endl;
         a=a+10;
         return a;
    }
    int constFucntion(int a) const{
         return a;
     }
};
main( ){
     test t;
     cout<<"Constant Function is called"<<endl;
     cout<<t.nonconstFucntion(10)<<endl;
     cout<<t.constFucntion(30);
     return 0;
}
```

Output:

```
Constant Function is called
Non Constant Function is called
20
30
--------------------------------
Process exited after 0.6311 seconds with return value 0
Press any key to continue . . .
```

CONSTANT OBJECTS

As with normal variables we can also make class objects constant so that their value can't change during program execution. Constant objects can only call constant member functions. The reason is that only constant member function will make sure that it will not change value of the object. They are also called as read only objects. To declare constant object just write const keyword before object declaration.

```
#include<iostream>
using namespace std;
class test{
public:
        int a;
        test( ){
            a=8;
        }
        int nonconstFucntion( ){
             cout<<"Non Constant Function is called "<<endl;
            //a=a+10;
           return a;
      }
        int constFucntion(int a) const{
             this->a=a+10; //error
            return a;
      }
};
int main( ){
        const test t;
        cout<<"Constant Function is called"<<endl;
        t.a=10;  // error, can't modify const objects
         cout<<t.nonconstFucntion():://error, can't call non const objects
       cout<<t.constFucntion(10);
       return 0;
}
```

Static Members of a Class

Similar to **static variables** a class can have **static members**. Let us note the following about the static members of a class.

- If a function of a class is static, in the class definition it is declared using the keyword static in its heading.
- If a member variable of a class is static, it is declared using the keyword static.
- A static member, function, or variable of a class can be accessed using the class name and **the scope resolution operator '::'**.

Defining the static data member

It should be defined outside of the class following this syntax:

- data_type class_name :: member_name =value;
- If you are calling a static data member within a member function, member function should be declared as static (i.e. a static member function can access the static data members)

```
#include <iostream>
using namespace std;
class Demo{
private:
        static int x;
public:
        static void fun( ){
        cout<<"value of X: " << x << endl;
        }
};
//defining
int Demo::x=10;
int main( ){
        Demo X;
        X.fun( );
        return 0;
}
```

Accessing static data member without static member function:

```
#include <iostream>
using namespace std;
class Demo{
public:
        static int ABC;
};
//defining
int Demo :: ABC =10;
int main( ){
        cout<<"\nValue of ABC: "<<Demo::ABC;
        return 0;
}
```

Exercise

1. Create a class 'Employee' having two data members **'EmployeeName'(char*)** and 'EmployeeId' (int). Keep both data members private. Create three initialized objects 'Employee1', 'Employee2' and 'Employee3' of type 'Employee' in such a way that the employee name for each employee can be changed when required but the employee Id for each employee must be initialized only once and should remain same always. Use member initializer list, accessors/getters and mutators/setters for appropriate data members. The result must be displayed by calling the accessors.
2. Create a class called **DynamicArray**. The class should contain a pointer to the array, and size of the array as data members. Create the parameterized constructor which take size of the array as input and initializes all the values with 0. Create the member function "push" which takes value as parameter and push it to the end of the array. Create the member function size() which returns the size of the array.
3. Write a program in which a class named **Account** has private member variables named account_no ,account_bal ,security_code. Use a public function to initialize the variables and print all data. Keep track of number of objects using the keyword static.
4. By considering a scenario of your own choice, write a program to demonstrate the concept of constant keyword.
5. "Hotel Mercato" requires a system module that will help the hotel to calculate the rent of the customers. You are required to develop one module of the system according to the following requirements:
    - The hotel wants such a system that should have the feature to change the implementation independently of the interface. This will help when dealing with changing requirements.
    - The hotel charges each customer 1000.85/- per day. This amount is being decided by the hotel committee and cannot be changed fulfilling certain complex formalities.
    - The module then analyses he number of days. If the customer has stayed for more than a week in the hotel, he gets discount on the rent. Otherwise, he is being charged normally.
    - The discounted rent is being calculated after subtracting one day from the total number of days.
    - In the end, the module displays the following details:
        - Customer name
        - Days
        - Rent

    Note that, the function used for displaying purpose must not have the ability to modify any data member.