NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCE

CL 1004 - Object Oriented Programming Lab

Semester: Spring 2022

Lab 05

Instructor: Muhammad Sudais
Email: muhammad.sudais.v@nu.edu.pk

Outline

- this Pointer
- Constant Key word
- Static Key Word
- Examples
- Exercise

this POINTER

- 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.
- 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(->).

CONSTANT DATA MEMBERS IN CLASSES

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

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-const object.
- 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.

CONSTANTOBJECTS

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.

STATIC VARIABLE IN FUNCTIONS

A variable declared static in a function retains its state between calls to that function.

STATIC CLASS MEMBERS

- There is an important exception to the rule that each object of a class has its own copy of all the data members of the class. In certain cases, only one copy of a variable should be shared by all objects of a class. A static data member is used for these and other reasons.
- A static member variable cannot be initialized inside the class declaration. That's because the declaration is a description of how memory is to be allocated, but it doesn't allocate memory.

- Static members exist as members of the class rather than as an instance in each object of the class. So, this keyword is not available in a static member function.
- A non-static member function can be called only after instantiating the class as an object. This is not the case with static member functions. A static member function can be called, even when a class is not instantiated.
- Static functions may access only static data members.

THIS HOLDS THE ADDRESS OF CURRENT OBJECT

```
#include <iostream> using
namespace std; class example
{
private: int
x; public:

/* If function argument and data member is same then use this pointer to identify the object's field */
void set(int x)
{
(*this).x = x;
}

int get()
{
return x;
}

void printAddressAndValue()
{
cout<<"The adrress 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
```

CHAINED FUNCTION CALLS

```
#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;
        }
        Test& setX(int a)
        {
```

```
                x = a;
                return *this;
                }
        Test& setY(int b)
        { y = b;
                  return *this;
        }
        void print()
        { cout<< "x = " << x << " y = " << y <<endl;
        }
};

int main()
{
    Test obj1(5, 5);
// Chained function calls.
All calls modify the same object // as the same object is
    returned by reference
    obj1.setX(10).setY(20).print();
 }
```

```
x = 10 y = 20

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

CONSTANT DATA MEMBERS IN CLASSES

```
#include <iostream>  using
namespace std;  class Students
{
private:
string name;
const int rollno;                              // need to be initialized once member is created
float cgpa;
 public:
Students(int rno):rollno(rno){}              // using member initializer list 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 MEMBERFUNCTIONS

```
#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
                {
                        cout<<"Constant Function is called"<<endl;
                //      this->a=a+10;
                return a;
        }
};
main()
{
        test t;
        cout<<t.nonconstFucntion(10)<<endl;
        cout<<t.constFucntion(20);
return 0;
}
```

CONSTANT OBJECTS

```
#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
                {
```

```
                cout<<"Constant Function is called"<<endl;
                // this->a=a+10; error
        return a;
                }
};
int main()
{
const test t;
//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 VARIABLES IN FUNCTIONS

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

void showstat( int curr )
{
        static int nStatic;          // Value of nStatic is retained
        // between each function call
nStatic += curr;
        cout<< "nStatic is " <<nStatic<<endl;
}

int main()
{
        for ( int i = 0; i< 5; i++ )
                showstat( i );
        return 0;
}
```

STATIC VARIABLES IN FUNCTIONS

```
#include <iostream> using namespace
std;
class Car
{
        int year;          int mileage
= 34289;
        // Warning: not-static data members initializers only available with c++11 or gnu++11 */

        static int vin = 12345678;
        // error: non-constant data member
        // only static const integral data members
        // can be initialized within a class
```

```
        static const  string model = "Sonata";  // error: not-integral type
        // cannot have in-class initializer          static const int engine = 6;        //
allowed: static const integral type public:          static void f(int);
};

int Car::year = 2013;      // error: non-static data members  //
cannot be defined out-of-class

void Car::f(int z)
{
        mileage=z;
        }
/* error: f(), a static function,  is trying to
access non-static member x. */

intmain()
{
return 0;
}
```

STATIC CLASS VARIABLE

```
#include<stdio.h>
#include<iostream> using namespace
std;
class ATM
{
        public:
        static int balance;

};
int ATM::balance = 250;
main()
{
        ATM ob1,ob2,ob3;
 cout<<ob1.balance++<<endl;
 cout<<ob2.balance++<<endl;
        cout<<ob3.balance;

}
```

EXERCISE

Question # 1

- Take any class from your course project. *"Each member should have different class from their project"*.
- Make one of its int data member constant.
- Create 4 functions to see the use of Constant keyword. Each function should return the modified integer **(not void)**.

Function Name	Data to be modified	Function Type
Int NonConstant(int x)	Non-Constant	Non-Constant
Int ConstData(int x)	Constant	Non-Constant
Int ConstFunction(int x)	Non-Constant	Constant
Int ConstBoth(int x)	Constant	Constant

Question # 2

- Take the same class from your project again.
- Take an integer data member (assume it is int x) from this class.
- Create a function **chain()** in this class that returns the object of your class (returns **this** basically) after decrementing the data member x by 1.
- In the driver function (main), create an object with x as 4. Call the function chain() in a single line as many times as it requires to change the value of x to 0.

Hint: Chained Function Calls and use of this pointer.

Question # 3

- Create a class 'Student' having two data members 'StudentName'(string) and 'EnrollmentId' (int).
- Keep both data members private.
- Create three initialized objects 'Student1', 'Student2' and 'Student3' of type 'Employee' in such a way that the employee's 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 and mutators for appropriate data members. The result must be displayed by calling the accessors.
- All of the accessors must not have the ability to modify the data.

Question # 4

In the above class 'Student', add one static data member that stores the number of student in a class. Create a static function that returns the number of Students.

Question # 5

Define a class to represent a **Bank account**. Include the following members.

Data members: -

1. Name of the depositor
2. Account number
3. Type of account.
4. Balance amount in the account.
5. Rate of interest

Provide a default constructor, a parameterized constructor to this class.

Also provide Member Functions: -

1. To deposit amount.
2. To withdraw amount after checking for minimum balance.
3. To display all the details of an accountholder.
4. Display rate of interest (a static function)

Illustrate all the constructors as well as all the methods by defining objects.