

UEE1303(1070) S12: Object-Oriented Programming

Advanced Topics of Class



What you will learn from Lab 6

In this laboratory, you will learn the advance topics of object-oriented programming using class.

TASK 6-1 STATIC MEMBER

- ✓ static member can be taken as a global member for this class and all objects own the same copy (or value) of the member.

```
// lab6-1.cpp
#include <iostream>

class Point2D
{
private:
    int x;
    int y;
    static const double limit = 10.0;
    static double value; // indicates that all object's value are the same

public:
    Point2D();
    void assignPoint2D(int x, int y);
    void displayPoint2D();
    static void setValue(double v);
    // only static member function can access static member
};

Point2D::Point2D()
{
    x = 0;
    y = 0;
}

void Point2D::assignPoint2D(int n1, int n2)
{
    x = n1;
    y = n2;
}

void Point2D::displayPoint2D()
{
    std::cout << "(" << x << "," << y << ") = ";
    std::cout << value << std::endl;
}
```

```
void Point2D::setValue(double v)
{
    if (v < limit)
        value = v;
    else
        value = limit;
}

double Point2D::value = 0.0;  // It needs to initialize static member

int main()
{
    Point2D ptArray[10];
    ptArray[0].setValue(1.1);
                        // modify the static member by static member fuction

    for (int i=0;i<10;i++)
    {
        ptArray[i].assignPoint2D(i,i+2);
        ptArray[i].displayPoint2D();
    }

    return 0;
}
```

- Remark the line `double Point2D::value = 0.0;` and compile the program again. Try to explain the error message.
- Remove static in `static const double limit = 10.0;` and compile the program again. Remove const in `static const double limit = 10.0;` and compile the program again.
- Try to modify `ptArray[0].setValue(1.1);` as `ptArray[0].setValue(30.1);` and execute the program again.

TASK 6-2 CONST AND MUTABLE MEMBERS

- ✓ `const` member functions are not supposed to modify objects of a class. However, if a data member is declared to be mutable, then it can be changed by any member function even in a `const` member function. Please identify which member function should be `const` to make the program work successfully.

```
// lab6-2-1.cpp

/* class Point2D declares and defines in lab6-1*/

int main()
{
    const Point2D pt1;
    Point2D pt2;
```

```
    pt1.displayPoint2D();  
    pt2.displayPoint2D();  
  
    return 0;  
}
```

```
// lab6-2-2.cpp  
  
/* class Point2D declares and defines in lab6-2-1*/  
/* add mutable (int) member named color to class Point2D */  
  
void Point2D::displayPoint2D() const  
{  
    x = 5; y = 4;  
    color = 10;  
    std::cout << "(" << x << "," << y << ") = ";  
    std::cout << value << std::endl;  
}  
  
int main()  
{  
    const Point2D pt1;  
    Point2D pt2;  
  
    pt1.displayPoint2D();  
    pt2.displayPoint2D();  
  
    return 0;  
}
```

- Please identify the difference between mutable data member and non-mutable data member.

TASK 6-3 THIS POINTER

- ✓ this pointer is an implicit private member to store the address of the object for a class.

```
// original version in lab5-2.cpp  
PointND::PointND()  
{  
    value = 0.0;  
    coord = new int [num];  
    for (int i=0;i<num;i++) coord[i] = 0;  
}  
  
// modify version in lab6-3-1.cpp  
PointND::PointND()  
{
```

```
this->value = 0.0;  
this->coord = new int [num];  
for (int i=0;i<num;i++) this->coord[i] = 0;  
}
```

- ✓ this pointer includes the address of the object, so it can be used to compare the addresses between different objects.

```
// lab6-3-2.cpp  
#include <iostream>  
  
/* class PointND declares and defines in lab 5-2 with copy constructor*/  
/* add declaration of member function: copyPoint2D() to class PointND */  
  
void PointND::copyPointND(const PointND &pt)  
{  
    if (this != &pt)  
    {  
        value = pt.value;  
        coord = new int [num];  
        for (int i=0;i<num;i++) coord[i] = pt.coord[i];  
    }  
}  
  
int main()  
{  
    int *vec = new int [num];  
    for (int i=0;i<num;i++) vec[i] = i;  
  
    PointND pt1;  
    pt1.assignValue(4.3);  
    pt1.assignCoord(vec,num);  
    pt1.displayPointND();  
  
    PointND pt2;  
    pt2.copyPointND(pt1);  
    pt2.displayPointND();  
  
    PointND pt3;  
    pt3.copyPointND(pt3);  
    pt3.displayPointND();  
  
    delete []vec;  
  
    return 0;  
}
```

TASK6-4 NESTED CLASS

- ✓ A class can be defined in another class, so called nested class. *Nested class* can be taken as a (public, private, or protected) member in the *enclosing class*. The name of nested class can be resolved in enclosing class scope, but it cannot be access in other class scope or other namespace.

```
// lab6-4.cpp
#include <iostream>
#include <assert.h>

class Vec
{
public:
    Vec(){len =0;}
    Vec(int n);
    ~Vec();

    void setValue(int idx, int v);
    void printVec() const;

private:
    class Items                // nested class Items for Vec
    {                          // all members in Items are private
        friend class Vec;     // make Vec can access member in Items
        Items(){value = 0;}
        Items(int v){value = v;}
        int value;
    };

    int len;
    Items *vec;
};

Vec::Vec(int n)
{
    len = n;
    vec = new Items [len];
}

Vec::~~Vec()
{
    if (len > 0)
        delete []vec;
}

void Vec::setValue(int idx, int v)
{
    assert(idx < len);
    vec[idx].value = v;
}
```

```
}

void Vec::printVec() const
{
    for (int i=0;i<len;i++)
        std::cout << vec[i].value << " ";
    std::cout << std::endl;
}

int main()
{
    Vec vector(5);
    vector.printVec();

    for (int i=0;i<5;i++)
        vector.setValue(i,i);
    vector.printVec();

    Items n;

    return 0;
}
```

- There is a compiler error in this example because the nested class cannot be used in global scope. Try to modify the program and make Items be accessed in global scope.

TASK 6-5 EXERCISE

1. *QUADRANGLE

- ✓ Define a class for a quadrangle given four vertices (define a point called vertices), which is represented by an object of a *nested class* for two-dimensional points. Using member initializer lists, define two constructors for it, one with four points as arguments for the four vertices and another with two points (lower-left corner and upper-right corner) to represent a rectangle. A data member decides if the quadrangle is rectangle or not and a static data member defines the origin as (0,0). Also define a destructor to destroy the object. The member functions move() and draw() moves a quadrangle to a new location, and print out the coordinates of the quadrangle and the distance from the origin (use the lower-left corner to calculate it), respectively. If the quadrangle is also a rectangle, print out the area of it. Write a program include several quadrangle object to test all of your member functions work properly.
- ✓ The main function is defined as follows,

```
// Ex6-1.cpp
int main()
{
```

```
    quadrangle q1(quadrangle::vertex(0,0),
                  quadrangle::vertex(3,2),
                  quadrangle::vertex(10,7),
                  quadrangle::vertex(8,10));
    quadrangle q2(quadrangle::vertex(3,6), quadrangle::vertex(10,9));
    cout << "q1 information" << endl;
    q1.draw();
    cout << endl;
    cout << "q2 information" << endl;
    q2.draw();
    cout << endl;
    cout << "q1 move to (5,5) " << endl;
    q1.move(quadrangle::vertex(5,5));
    cout << "q1 information" << endl;
    q1.draw();
    cout << endl;
    quadrangle::origin = quadrangle::vertex(-5,3);
    cout << "q2 move to (-1,2) " << endl;
    q2.move(quadrangle::vertex(-1,2));
    cout << "q2 information" << endl;
    q2.draw();
    cout << endl;
    return 0;
} // end main
```

✓ The sample output is

```
q1 information
v1: (0,0)      v2: (3,2)      v3: (10,7)      v4: (8,10)
q2 information
v1: (3,6)      v2: (10,6)     v3: (10,9)     v4: (3,9)      area: 21
q1 move to (5,5)
Distance: 7.07107
q1 information
v1: (5,5)      v2: (8,7)      v3: (15,12)    v4: (13,15)
q2 move to (-1,2)
Distance: 4.12311
q2 information
v1: (-1,2)     v2: (6,2)     v3: (6,5)     v4: (-1,5)     area: 21
```