EEEC10008(515169) S23: Object-Oriented Programming

Constructors, Destructors, Accessor and Mutator

What you will learn from Lab 3

In this laboratory, you will learn how to use constructor and copy constructor to create an object and use destructor to delete it. You will also learn how to write accessor and mutator functions to set and get the private variables of a class.

TASK3-1 CONSTRUCTOR

✓ Please try to compile and execute the program `lab3-1-1`, and observe the results.

```
// lab3-1-1.cpp
#include <iostream>

class Point2D
{
private:
    int x;
    int y;
    double value;

public:
    void assignPoint2D(int n1, int n2, double v);
    void displayPoint2D();
};

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

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

int main()
{
    Point2D ptArray[10];
    for (int i=0;i<10;i++)
       ptArray[i].displayPoint2D();
    return 0;
}
```

✓ We add ***constructor*** to class Point2D and make program `lab3-1-2` work as expect.

```
// lab3-1-2.cpp
…
class Point2D
{
private:
    int x;
    int y;
    double value;

public:
    Point2D();          // default constructor
    void assignPoint2D(int n1, int n2, double v);
    void displayPoint2D();
};

Point2D::Point2D()
{
    x = 0;
    y = 0;
    value = 0.0;
}
…
```

➢ You can also use a parenthesized expression list to build your default constructor. In the above example, you can replace the declaration and definition of default constructor as `Point2D():x(0), y(0), value(0.0) {}`.

✓ Please modify your class `Point2D` to make the `lab3-1-3` work.

```
// lab3-1-3.cpp
…
int main()
{
    Point2D pt1;
    Point2D pt2(1,2);
    Point2D pt3(3,2,1.9);

    pt1.displayPoint2D();
    pt2.displayPoint2D();
    pt3.displayPoint2D();
    pt3.assignPoint2D(2,1,0.0);
    pt3.displayPoint2D();

    return 0;
}
```

➢ Both `Point2D pt3(3,2,1.9)` and `pt3.assignPoint2D(2,1,0.0)` can assign the value to pt3's private member. Can you explain their difference?

TASK3-2 DESTRUCTOR

- ✓ In class `Point2D`, we do not specific the destructor `~Point2D()` since the compiler will generate one. However, if you use new or delete memory in the object, you need constructor to allocate memory and destructor to release it.
- ✓ Please modify the class `Point2D` as `PointND`, which is used to record the N-dimensional coordinate using an integer array.

```
// lab3-2.cpp
#include <iostream>
#include <assert.h>

const int num = 10;

class PointND
{
private:
    int *coord;
    double value;

public:
    PointND();
    ~PointND();
    void assignValue(double v);
    void assignCoord(int *vec, int len);
    void displayPointND();
};

PointND::PointND()
{
    value = 0.0;
    coord = new int [num];
    for (int i=0;i<num;i++) coord[i] = 0;
}

PointND::~PointND()
{
    delete []coord;
}

void PointND::assignValue(double v)
{
    value = v;
}

void PointND::assignCoord(int *vec, int len)
{
    assert(len <= num);          // make sure len <= num
    for (int i=0;i<len;i++)
        coord[i] = vec[i];
}
```

```
void PointND::displayPointND()
{
    std::cout << "(";
    for (int i=0;i<num;i++)
    {
        std::cout << coord[i];
        if (i!=num-1)
           std::cout << ", ";
    }
    std::cout << ") = " << value << std::endl;
}

int main()
{
    PointND pt1;
    pt1.displayPointND();

    PointND pt2;
    pt2.assignValue(1.0);
    pt2.displayPointND();

    int *vec = new int [num];
    for (int i=0;i<num;i++) vec[i] = i;

    PointND pt3;
    pt3.assignValue(4.3);
    pt3.assignCoord(vec,num);
    pt3.displayPointND();

    delete []vec;
    return 0;
}
```

TASK3-3 COPY CONSTRUCTOR

✓ Copy constructor is a constructor used to create a new object as a copy of an existing object.

```
// lab3-3.cpp
#include <iostream>
#include <assert.h>

/* 1. class PointND
   2. add the definition of copy constructor to the class*/

PointND::PointND(const PointND &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(pt1);
   pt2.displayPointND();

   delete []vec;
   return 0;
}
```

TASK3-4 ACCESSOR AND MUTATOR FUNCTIONS

✓ `lab3-4.cpp` is a simple example of acccessor functions and mutator functions.

```
// lab3-4.cpp
#include <iostream>
using namespace std;

class Point{
   public:
      Point(){
         x = 0;
         y = 0;
      }

      int get_x(){ //Accessor
         return x;
```

```
        }

        int get_y(){ //Accessor
            return y;
        }

        void set_x(int px) { //Mutator
            x = px;
        }

        void set_y(int py) { //Mutator
            y = py;
        }

    private:
        int x;
        int y;
};

int main()
{
    Point p;
    p.set_x(1);
    p.set_y(2);
    cout << "Point: x = " << p.get_x() << ",y = " << p.get_y() << endl;

    return 0;
}
```

EXERCISE 3-1

- ✓ In this exercise, you need to define two structs, Point and Line, and one class, LineGp.
- ✓ The `struct` Point has following variables:

  x(double), **y**(double)
- ✓ The `struct` Line has following variables:

  p1(Point), **p2**(Point)
- ✓ The `class` LineGp has following variables and functions:
  - Private Variables:
    - **lines** (Line [3])
    - **intersectP** (vector<Point>)
  - Constructors:
    - LineGp()
  - Private Functions:
    - void **set_L**(int, Line): set line[i], being called in set_lines().
    - void **intersect**(Line, Line): find the intersection of 2 lines, being called in intersect_lines().
    - void **intersect_lines**(): find the intersection between each line.
  - Public Functions:
    - void **set_lines**(): set lines.
    - void **printLines**(): print line info.
    - void **computeArea**(): compute the area enclosed by three lines.

- ✓ Execution Result

```
$./ex3-1
L0:
Point 1: 2 5
Point 2: 3 7
L1:
Point 1: 4 6
Point 2: 2 4
L2:
Point 1: 1 9
```

```
Point 2: 3 3
////LINES////
L0: (2,5) (3,7)
L1: (2,4) (4,6)
L2: (1,9) (3,3)
////LINES////

L0, L1
2x - 1y = -1
2x - 2y = -4
Intersect point isn't in range!
L0, L2
2x - 1y = -1
-6x - 2y = -24
Intersect at (2.2,5.4)
L1, L2
2x - 2y = -4
-6x - 2y = -24
Intersect at (2.5,4.5)

No Triangle

L0:
Point 1: -3 0
Point 2: 3 7
L1:
Point 1: 4 0
Point 2: 0 5
L2:
Point 1: 4 6
Point 2: -1 3
////LINES////
L0: (-3,0) (3,7)
L1: (0,5) (4,0)
L2: (-1,3) (4,6)
////LINES////

L0, L1
```

```
7x - 6y = -21
-5x - 4y = -20
Intersect at (0.62069,4.22414)
L0, L2
7x - 6y = -21
3x - 5y = -18
Intersect at (0.176471,3.70588)
L1, L2
-5x - 4y = -20
3x - 5y = -18
Intersect at (0.756757,4.05405)

Area is: 0.073036

L0:
Point 1: 2 6
Point 2: -2 4
L1:
Point 1: 0 5
Point 2: 3 9
L2:
Point 1: 7 11
Point 2: 1 3
////LINES////
L0: (-2,4) (2,6)
L1: (0,5) (3,9)
L2: (1,3) (7,11)
////LINES////

L0, L1
2x - 4y = -20
4x - 3y = -15
Intersect at (0,5)
L0, L2
2x - 4y = -20
8x - 6y = -10
Intersect point isn't in range!
L1, L2
```

```
4x - 3y = -15
8x - 6y = -10
Two lines are parallel!

No Triangle
```

✓ Hint

To find the points of intersection, consider line as following equations:

```
X(t) = x1 + (x2 - x1) * t
Y(t) = y1 + (y2 - y1) * t
```

Reference: https://en.wikipedia.org/wiki/Heron%27s_formula

EXERCISE 3-2

✓ In this exercise, we will provide ex3-2.cpp, which is under **/home/share/lab3/**, and you need to complete the header file (Class.h) and source file (Class.cpp) by yourself. Besides, you also have to use struct "Student" to store student information.

✓ The `struct` Student has following variables:

name (string), **student_id** (string), **score** (int)

✓ The `class` Class has following variables and functions:

- Private Variables:
  - **class_size** (int)
  - **passing_score** (int)
  - **students** (Student *)
- Constructors:
  - Class()
  - Copy constructor: please define the function by yourself.
- Private Functions:
  - void **set_class_size**(int size): set class size, being called in initialize().
  - void **set_students**(Student *): set students array, being called in initialize().
  - int **find_student**(string): find the student by **name**, and return the index.
  - double **find_average**(): return average score, being called in score_info().

- double **find_median**(): return median score, being called in score_info()
- int **find_fail**(): return how many students fail to pass the passing score, being called in score_info()

- Public Functions:
  - void **initialize**(): called at first to set class size and student members.
  - void **students_info**(): print students' info, in the order of **student_id**.
  - void **score_info**(): print score info of this class.
  - void **set_passing_score**(int): set passing score.
  - void **set_score**(string, int): set student's score.
  - void **add_student**(Student): add student into the class.
  - void **remove_student**(string): remove student by name.

✓ Execution Result

```
$./ex3-2
Math
Number of students: 7
Set Passing Score: 60
Student Name: Amy
Student ID: A2051
Score: 80
Student Name: Zack
Student ID: A3955
Score: 38
Student Name: Johnny
Student ID: B2042
Score: 52
Student Name: Lowry
Student ID: B7273
Score: 93
Student Name: Nicholas
Student ID: B2651
Score: 73
Student Name: Nancy
Student ID: A6745
Score: 88
Student Name: Brian
```

```
Student ID: B1825
Score: 78
Amy    A2051   80
Zack   A3955   38
Nancy   A6745   88
Brian   B1825   78
Johnny  B2042   52
Nicholas       B2651   73
Lowry   B7273   93

Jimmy   A1703   92
Amy    A2051   80
Zack   A3955   38
Nancy   A6745   88
Brian   B1825   78
Johnny  B2042   52
Nicholas       B2651   73
Lowry   B7273   93

Average Score: 74.25
Median Score: 79
Fail Student Number: 2

Chinese
student doesn't exist!
Jimmy   A1703   92
Amy    A2051   92
Nancy   A6745   88
Brian   B1825   78
Johnny  B2042   52
Nicholas       B2651   73
Lowry   B7273   93

Average Score: 81.1429
Median Score: 88
Fail Student Number: 0

Chemistry
```

```
student doesn't exist!
Jessica A0222   77
Jimmy   A1703   92
Amy     A2051   92
Brian   B1825   64
Johnny  B2042   52
Nicholas        B2651   73
Bob     B4320   54
Lowry   B7273   93

Average Score: 74.625
Median Score: 75
Fail Student Number: 4
```