National Yang Ming Chiao Tung University
Department of Electrical and Computer Engineering
Computer Intelligence on Automation(C.I.A.) Lab

Laboratory Manual 07
April 24, 2023
Prof. Hung-Pin (Charles) Wen

EEEC10008 S23: Object-Oriented Programming
Inheritance

What you will learn from Lab 7

In this laboratory, you will learn the concept of inheritance and its usage.

TASK 7-1 EXAMPLE OF INHERITANCE

✓ Please compile and execute the program `lab7-1`, where `Point4D` is a derived class from the base class `Point2D`.

```
// lab7-1-1.cpp
#include <iostream>
using std::cout;
using std::endl;

class Point2D
{
private:
   int x;
   int y;

public:
   Point2D(int n1 = 0, int n2 = 0):x(n1), y(n2){}
   void display() const;
};

void Point2D::display() const
{
   cout << x << "," << y;
}

class Point4D : public Point2D
{
private:
   int z;
   int t;

public:
   Point4D(int n1=0,int n2=0,int n3=0,int n4=0):Point2D(n1,n2),z(n3), t(n4){}
   void display() const;
};

void Point4D::display() const
{
   Point2D::display();
   cout << "," << z << "," << t;
}
```

```
int main()
{
   Point4D pt(1,2,3,4);
   pt.display(); cout << endl;

   return 0;
}
```

- You can put the constructor of the base class in the initialization list for the derived class.
- Note that member function of a derived class cannot access the private part of a base class. For example, the function `Point4D::display()` cannot be defined as

  ```
  void Point4D::display() const
  {
      cout << x << "," << y;    // x and y are inaccessible
      cout << "," << z << "," << t;
  }
  ```

  The hidden member `x` and `y` of the derived class `Point4D` is accessible through the public member function `Point2D::display();`.
- You can define *accessor* and *mutator* functions in Point2D to access private members.

✓ Please compile and execute the program `lab7-1-2`

```
// lab7-1-2.cpp

/*  The Point2D and Point4D class defined in lab7-1-1  */

int main()
{
   Point2D pt2(3,4);

   Point4D pt4(1,2,3,4);
   pt4.display(); cout << endl;

   pt2 = pt4;                  // OK, every Point2D is a Point4D
   pt2.display(); cout << endl;

   pt4 = pt2;                  // Error, not every Ponint4D is a Point2D
   pt4.display(); cout << endl;

   return 0;
}
```

- You can comment out the incorrect lines to observe the results.
- If you require type conversion from a base class to derived class (e.g., `pt4 = pt2`), you must provide additional member functions of `Point4D` to achieve it.

- ✓ Please compile and execute the program `lab7-1-3`

```
// lab7-1-3.cpp

/*  The Point2D and Point4D class defined in lab7-1-1  */

void f(const Point2D &p1, const Point2D &p2)
{
   p1.display(); cout << endl;
   p2.display(); cout << endl;
}

int main()
{
   Point2D pt2(3,4);
   Point4D pt4(1,2,3,4);

   f(pt2, pt4);

   return 0;
}
```

  - ➢ Note that the prototype of function `f` is
    `void f(const Point2D &, const Point2D &).`

TASK 7-2 CLASS HIERARCHY

- ✓ A derived class can be a base class of another derived class.

```
// lab7-2.cpp

/*  The Point2D and Point4D class defined in lab7-1-1  */

class Car : public Point4D
{
private:
   int color;
   int year;
public:
   Car(int n1=0, int n2=0, int n3=0, int n4=0):Point4D(n1,n2,n3,n4)
   {
      color = 0;
      year = 0;
   }
   Car(const Point4D &p):Point4D(p){color = 0; year = 0;}  // copy constructor

   void display() const;
   void setColor(const int c){color = c;}
   void setYear(const int y){year = y;}
};
```

```
void Car::display() const
{
    cout << "color: " << color << endl;
    cout << "year: " << year << endl;
    Point4D::display();
}

int main()
{
    Point4D pt4(1,2,3,4);

    Car c1(pt4);
    c1.setColor(128);
    c1.setYear(2011);
    c1.display(); cout << endl;

    return 0;
}
```

- Note that, to enable copy constructor of `Car`, you should also provide copy constructor for `Point2D` and `Point4D`.

TASK 7-3 ACCESS TO BASE CLASSES

✓ In the following example, `B` is a public-base class for `X`. Please fix the compiling error here.

```
//lab7-3.cpp
#include <iostream>

class B
{
    private:
        int i;
    protected:
        float f;
    public:
        B() { i =0; f = 0.0; d =0.0; }
        double d;
        void g1(B b){f = b.f;}
};

class X: public B
{
    protected:
        short s;
    public:
        X() {s=0;}
        void g2(X x) {f = x.f;}
        void g3(B b) {f = b.f;}
};
int main()
```

```
{
   B b1;
   X x1;
   x1.g1(b1);

   return 0;
}
```

- Please modify `B` as a `protected` base and compile the program again.
- Guidelines for access control:
  - If B is a ***private*** base, its public and protected members become private members of derived class.
  - If B is a ***protected*** base, its public and protected members become protected members of derived class.
  - If B is a ***public*** base, its public members become public members of derived class and its protected members become protected members of derived class.
- The access control for protected member, like private member, is that only its member and friend can access it. However, the protected member can become private, protected, or public members of derived class but private member cannot. Therefore, protected members of a class are designed for use by derived classes and are not intended for general use.

EXERCISE 7-1

- ✓ Please implement the base class `Stocks` and derived class `Fruits`.
- ✓ **DO NOT MODIFY THE MAIN PROGRAMS**.
- ✓ Please finish the “`Stocks.h`”, “`Fruits.h`”
- ✓ In `Stocks`, there are two private members
  - ■ `type` is a pointer to integer, represent the number of stock types.
  - ■ `itemList` is a dynamic array of string, stores all stock types inside.
- ✓ Implement below two functions and any other functions you need
  - ■ `Initialize()`,called at first to set up `type` and `itemList`.
  - ■ `Display()`, print out all items in `itemList`.

```
// Stocks.h
#ifndef STOCKS_H
#define STOCKS_H
#include <iostream>
#include <string>
using namespace std;
```

```
class Stocks {
  private:
   int* type;
   string* itemList;

  public:
   /* any member functions if necessary */
   void Initialize();
   void Display();
};
#endif
```

✓ In `Fruits`, there are two private members

- `expired_month` is a pointer to integer, represent the expired month.
- `expired_date` is a pointer to integer, represent the expired date.

```
// Stocks.h
#ifndef FRUITS_H
#define FRUITS_H
#include "Stocks.h"

class Fruits : public Stocks {
  private:
   int *expired_month;
   int *expired_date;

  public:
   /* any member functions if necessary */
};

#endif
```

✓ You can get files `Stocks.h`, `Fruits.h` and `main.cpp` in `/home/share/lab7/ex1/`.

✓ To pass the test, your program cannot contain memory leaks. You can use the following command to test for memory leaks.

```
> valgrind <your_executable_file> <arguments_if_needed>
ex: valgrind ./ex1
```

✓ Please finish the remaining part to make the following main function work successfully.

```
#include "Fruits.h"
#include "Stocks.h"
using namespace std;

int main() {
   // Test 1
```

```
    Stocks A(3);  // create a Stocks called A, with *type = 3
    Stocks B(3);

    cout << "Stocks A" << endl;
    A.Initialize();
    A.Display();
    cout << endl;

    cout << "Stocks B" << endl;
    B.Initialize();
    B.Display();
    cout << endl;

    cout << "Stocks B = A" << endl;
    B = A;
    B.Display();
    cout << endl;

    Fruits C(3, 10, 15);  // create a Fruits called C, with *type = 3,
*expired_month = 10, *expired_date = 3

    cout << "Fruits C" << endl;
    C.Initialize();
    C.Display();
    cout << endl;

    cout << "Stocks B = C" << endl;
    B = C;
    B.Display();
    cout << endl;

    cout << "Fruits C = A" << endl;
    C = A;              // set expired_month and expired_month to 0
    C.Display();        // show No Expiration Date
    cout << endl;
    return 0;
}
```

✓ Sample output1:

```
valgrind ./ex1
Stocks A
Initialize
Item 1: Pen
Item 2: Book
Item 3: Erasor
Item List: Pen Book Erasor


Stocks B
```

```
Initialize
Item 1: Violin
Item 2: Piano
Item 3: Drum
Item List: Violin Piano Drum

Stocks B = A
Item List: Pen Book Erasor

Fruits C
Initialize
Item 1: Apple
Item 2: Banana
Item 3: Orange
Item List: Apple Banana Orange
Expired at 10/15

Stocks B = C
Item List: Apple Banana Orange

Fruits C = A
Item List: Pen Book Erasor
No Expiration Date
```

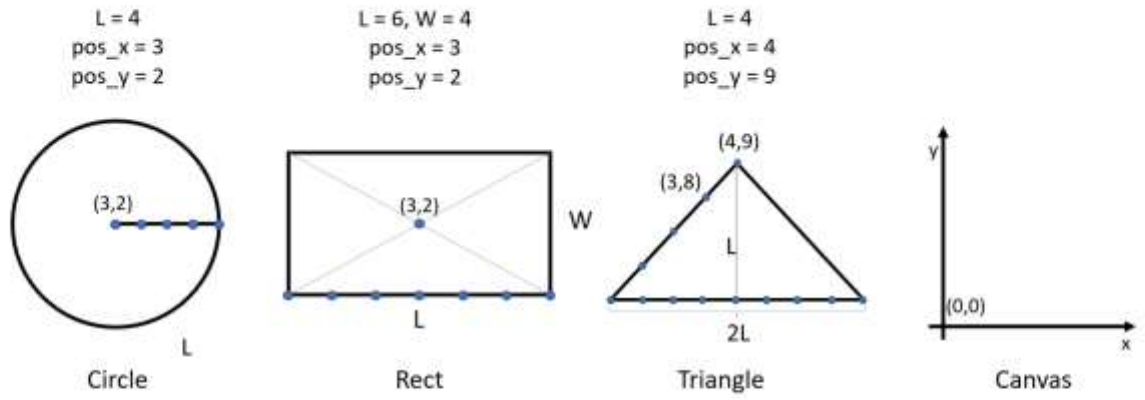
✓ Sample output2:

```
valgrind ./ex1
Stocks A
Initialize
Item 1: Pen
Item 2: Book
Item List: Pen Book

Stocks B
Initialize
Item 1: Table
Item 2: Chair
Item 3: Phone
Item List: Table Chair Phone
```

```

Stocks B = A
Item List: Pen Book

Fruits C
Initialize
Item 1: Melon
Item 2: Cherry
Item 3: Orange
Item 4: Guava
Item List: Melon Cherry Orange Guava
Expired at 10/15

Stocks B = C
Item List: Melon Cherry Orange Guava

Fruits C = A
Item List: Pen Book
No Expiration Date
```

EXERCISE7-2: SHAPE PLOTTING TOOL

- ✓ Shape plotting tool offer several different shape options, each with specific area, perimeter equation and plotting method. Create an **inheritance hierarchy** to represent **various types of shapes**. Use `Shape` as the base class of the hierarchy, and then include classes `Circle`, `Rect` and `Triangle` that derive from `Shape`. Also, include class `Square` derives from `Rect`.
- ✓ Base class `Shape` should include following data members
  - L : shape length
  - pos_x: x origin of shape
  - pos_y: y origin of shape
  - canvas_size: the size of plotting canvas, will be 20 if not specified
  - points: a vector store all plotting points.
- ✓ Example



- ✓ You should define `appendPoints()`,`draw()` for Base class `Shape`.
- ✓ You should define `computeArea()`,`ComputePerimeter()`,`ComputePoints()` for each derived class.
- ✓ **DO NOT MODIFY THE MAIN PROGRAMS.**
- ✓ Please finish the “`Shape.h`”, “`Circle.h`”, “`Rect.h`”, “`Square.h`” and “`Triangle.h`”.

```
// Shape.h
#ifndef SHAPE_H
#define SHAPE_H

#include <algorithm>
#include <cmath>
#include <iomanip>
#include <iostream>
#include <vector>
using namespace std;

struct Point {
    int x;
    int y;
};
/* any functions if necessary */
class Shape {
  private:
    int L;
    int pos_x;
    int pos_y;
    int canvas_size;
    vector<Point> points;  //all plotting points

  public:
    /* any member functions if necessary */
    void appendPoints(int x, int y);
    void draw();
};

#endif
```

```
// Circle.h
#ifndef CIRCLE_H
#define CIRCLE_H

#include "Shape.h"
#define PI 3.14

class Circle : public Shape {
  private:
  public:
   /* any member functions if necessary */
};
#endif
```

```
// Rect.h
#ifndef RECT_H
#define RECT_H

#include "Shape.h"

class Rect : public Shape {
  private:
   int W;
  public:
   /* any member functions if necessary */
};
#endif
```

```
// Square.h
#ifndef SQUARE_H
#define SQUARE_H

#include "Rect.h"
class Square : public Rect {
  private:
  public:
   /* any member functions if necessary */};

#endif
```

```
// Triangle.h
#ifndef TRIANGLE_H
#define TRIANGLE_H

#include "Shape.h"
class Triangle : public Shape {
  private:
  public:
   /* any member functions if necessary */};
};
```

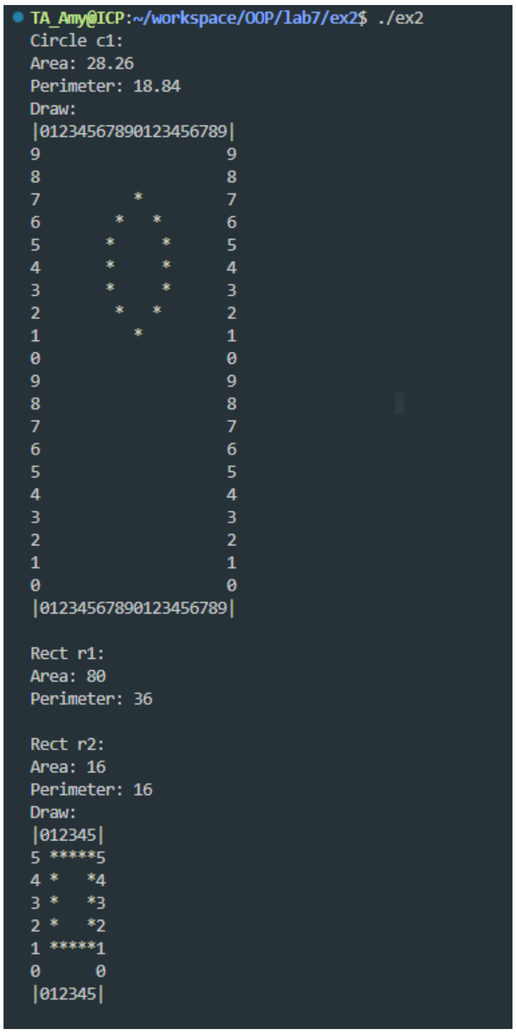
```
#endif
```

✓ Please finish the remaining part to make the following main function work successfully.

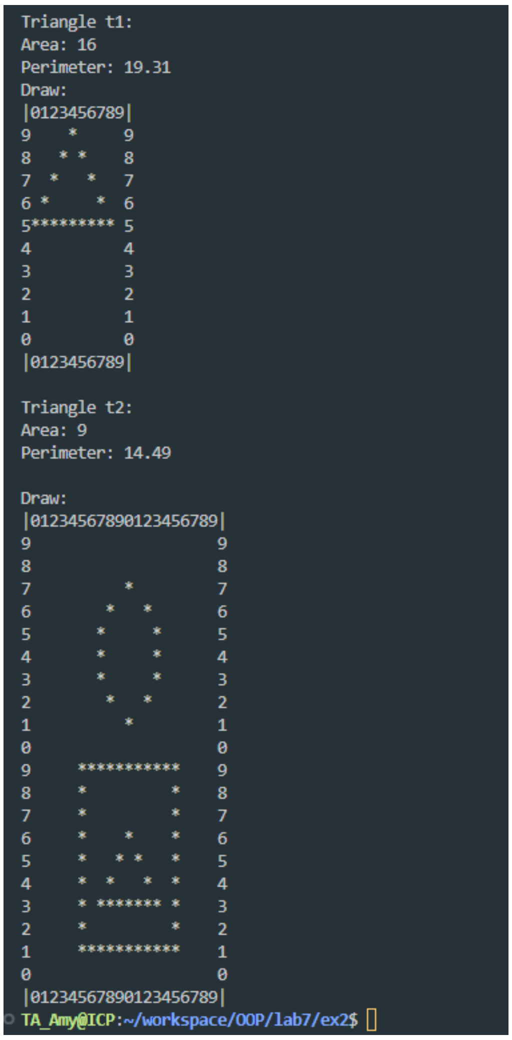
```
#include "Circle.h"
#include "Rect.h"
#include "Shape.h"
#include "Square.h"
#include "Triangle.h"
#define CANVA_LEN 20

int main() {
   cout << "Circle c1: " << endl;
   Circle c1(3, 10, 14);  // Circle(L, pos_x, pos_y, canvas_size = 20)
   c1.ComputeArea();
   c1.ComputePerimeter();
   c1.ComputePoints();
   c1.draw();
   cout << endl;

   cout << "Rect r1: " << endl;
   Rect r1(10, 8, 10, 5);  // Rect(L, W, pos_x, pos_y, canvas_size = 20)
   r1.ComputeArea();
   r1.ComputePerimeter();
   r1.ComputePoints();
   cout << endl;

   cout << "Rect r2: " << endl;
   Square r2(4, 3, 3, 6);  // Square(L, pos_x, pos_y, canvas_size = 20)
   r2.ComputeArea();
   r2.ComputePerimeter();
   r2.ComputePoints();
   r2.draw();
   cout << endl;

   cout << "Triangle t1: " << endl;
   Triangle t1(4, 4, 9, 10);  // Triangle(L, pos_x, pos_y, canvas_size = 20)
   t1.ComputeArea();
   t1.ComputePerimeter();
   t1.ComputePoints();
   t1.draw();
   cout << endl;

   cout << "Triangle t2: " << endl;
   Triangle t2(3, 10, 6);  // Triangle(L, pos_x, pos_y, canvas_size = 20)
   t2.ComputeArea();
   t2.ComputePerimeter();
   t2.ComputePoints();
   cout << endl;

   Shape s1(0, 0, 0);  // Shape(L, pos_x, pos_y, canvas_size = 20)
   for (auto p : c1.getPoints()) {
        s1.appendPoints(p.x, p.y);
    }
    for (auto p : r1.getPoints()) {
        s1.appendPoints(p.x, p.y);
    }
    for (auto p : t2.getPoints()) {
        s1.appendPoints(p.x, p.y);
    }
    s1.draw();
    // cout << endl;

    return 0;
}
```

✓ You can get files `Shape.h`, `Circle.h`, `Rect.h`, `Square.h`, `Triangle.h` and `main.cpp` in `/home/share/lab7/ex2/`.

✓ Sample output:



```
TA_Amy@ICP:~/workspace/OOP/lab7/ex2$ ./ex2
Circle c1:
Area: 28.26
Perimeter: 18.84
Draw:
|01234567890123456789|
9                    9
8                    8
7          *         7
6        *   *       6
5       *     *      5
4       *     *      4
3       *     *      3
2        *   *       2
1          *         1
0                    0
9                    9
8                    8
7                    7
6                    6
5                    5
4                    4
3                    3
2                    2
1                    1
0                    0
|01234567890123456789|

Rect r1:
Area: 80
Perimeter: 36

Rect r2:
Area: 16
Perimeter: 16
Draw:
|012345|
5 *****5
4 *   *4
3 *   *3
2 *   *2
1 *****1
0      0
|012345|
```



```
Triangle t1:
Area: 16
Perimeter: 19.31
Draw:
|0123456789|
9    *     9
8   * *    8
7  *   *   7
6 *     *  6
5********* 5
4          4
3          3
2          2
1          1
0          0
|0123456789|

Triangle t2:
Area: 9
Perimeter: 14.49

Draw:
|01234567890123456789|
9                    9
8                    8
7          *         7
6        *   *       6
5       *     *      5
4       *     *      4
3       *     *      3
2        *   *       2
1          *         1
0                    0
9     ***********    9
8     *         *    8
7     *         *    7
6     *    *    *    6
5     *   * *   *    5
4     *  *   *  *    4
3     * ******* *    3
2     *         *    2
1     ***********    1
0                    0
|01234567890123456789|
TA_Amy@ICP:~/workspace/OOP/lab7/ex2$ 
```

✓ Reference

✧ How to sort a vector:

https://www.geeksforgeeks.org/sorting-a-vector-in-c/