Computational Physics Lab

Introduction to Object-Oriented Programming with C++

02/19/2009

Outline

Homework Assignment

1. Read Chapter 8
   - *"Control logic and iteration"*
2. Assignments of Section 8.10: Part I, (1) - (11)
   - Due next Tuesday, February 24

Some Comments on Grading Scheme for Projects

1. The projects-related files must be posted on your `comphy` web site ...
2. The projects should be properly documented.
3. Your programs must compile.
4. All your programs should be coded according to our programming standards:

   Variables need to be initialized!
5. The programs must do what is required.
6. ...

Outline

Combining Data with Functionality

```
// Simple C style 2D point
double Point[2];    // Data array for x,y

Point[0] = 0.0;    // x value
Point[1] = 0.0;    // y value

void addPoints(double P1[], double P2[],
                    double Psum[]) {
    Psum[0] = P1[0] + P2[0];
    Psum[1] = P1[1] + P2[1];
}

void printPoint(double P[]) {
   cout << "point(x,y) is (" << P[0]
        << "," << P[1] << ")" << endl;
}
```

```
#include <iostream.h>

int main() {
    double P1[2], P2[2], P[2];
    P1[0] = 1.0;
    P1[1] = 1.0;
    P2[0] = 2.0;
    P2[1] = 2.0;

    addPoints(P1, P2, P);

    printPoint(P1);
    printPoint(P2);
    printPoint(P);
}
```

Outline

Defining an Object via C++ Class Declaration

A class is a template for containing

1. Data (often hidden from user)
2. Member functions
   ➜ Constructors, Destructors, Getters, & Setters
3. Scope

OOP Syntax: Point2D

```
class Point2D   {
        double iX;
        double iY;

        public:
        Point2D();
        Point2D(double aX, double aY);
        void setX(double aX) { iX = aX; }
        void setY(double aY) { iY = aY; }
        double x() { return iX; }
        double y() { return iY; }
        Point2D operator+(Point2D &aP);
        void print();
}
```

Function Definition

```
Point2D::print() {
        cout << "Point(x,y) is "
                << x() << " , "
                << y() << " , " << endl;
}
```

OOP Syntax: Point2D

```
class Point2D   {
      double iX;       double iY;
      public:
      Point2D();
      Point2D(double aX, double aY);
      void setX(double aX) { iX = aX; }
      void setY(double aY) { iY = aY; }
      double x() { return iX; }
      double y() { return iY; }
      Point2D operator+(Point2D &aP);
      void print();
}
```

➜ “private”

Function Definition

```
Point2D::print() {
      cout << “Point(x,y) is ”
            << x() << “ , ”
            << y() << “ , ” << endl;
}
```

OOP Syntax: Point2D

```
class Point2D   {
        double iX;
        double iY;

        public:
        Point2D();
        Point2D(double aX, double aY);
        void setX(double aX) { iX = aX; }
        void setY(double aY) { iY = aY; }
        double x() { return iX; }
        double y() { return iY; }
        Point2D operator+(Point2D &aP);
        void print();
}
```

Function Definition

```
Point2D::print() {
        cout << "Point(x,y) is "
                << x() << " , "
                << y() << " , " << endl;
}
```

OOP Syntax: Point2D

```
class Point2D   {
      double iX;
      double iY;

      public:
      Point2D();
      Point2D(double aX, double aY);
      void setX(double aX) { iX = aX; }
      void setY(double aY) { iY = aY; }
      double x() { return iX; }
      double y() { return iY; }
      Point2D operator+(Point2D &aP);
      void print();
}
```

Function Definition

```
Point2D::print() {
      cout << "Point(x,y) is "
            << x() << " , "
            << y() << " , " << endl;
}
```

Constructors

```
Point2D::Point2D() {
      setX(0.0);
      setY(0.0);
}
```

```
Point2D::Point2D(double aX, double aY) {
      setX(aX);
      setY(aY);
}
```

OOP Example: Point2D

```
#include <iostream>

using namespace std;

int main()   {

    Point2D P1;
    Point2D P2(1.0, 1.0);
    Point2D P3(2.0, 2.0);

    P1.print()
    P2.print()
    P3.print()

    P1 = P2 + P3;
    P1.print();
}
```

```
comphy > g++ points.cc -o points
comphy > ./points
    Point(x,y) is (0,0)
    Point(x,y) is (1,1)
    Point(x,y) is (2,2)
    Point(x,y) is (3,3)
comphy >
```

Data Encapsulation

If we want to represent the Point2D data by (r, Θ) rather than by (x, y) then very little changes are need in the class definition and **NO** changes will be needed in the user code.

Extending Objects: Line2D

```
class Line2D   {

    Point2D iP1;
    Point2D iP2;

    public:
    Line2D(Point2D aP1, Point2D aP2);
    void setP1(Point2D aP1);
    void setP2(Point2D aP2);

    ...
}
```

Line2D has a Point2D!

Inheritance: Point3D

```
class Point3D : public Point2D   {
    double iZ;

    public:
    Point3D();
    Point3D(double aX, double aY, double aZ);
    void setZ(double aZ) { iZ = aZ; }
    double z() { return iZ; }
    Point3D operator+(Point3D &aP);
    void print();
}

Point3D::Point3D(double aX, double aY, double aZ) {
    setX(aX);
    setY(aY);
    setZ(aZ);
}
```

Point3D is a Point2D!

Outline

This Week's Project

Write a program that contains a class **Rectangle** with two private double precision members **iLength** and **iWidth**, public set and get member functions for the two members, a two-argument constructor that sets the length and width to any two user-specified values, and a void function **area()** that computes the area and prints this value by inserting it into the **cout** output stream. Write a **main** function that creates a **Rectangle** with a length of 10 and width of 20 and computes its area.