

#include<iostream>

#include<string>

using namespace std;

class Date

{

int year, month, day;

public:

Date(int y = 0, int m = 0, int d = 0)

{

setYear(y);

setMonth(m);

setDay(d);

}

void setYear(int y) { this->year = y; }

void setMonth(int m) { this->month = m; }

void setDay(int d) { this->day = d; }

int getYear() { return this->year; }

int getMonth() { return this->month; }

int getDay() { return this->day; }

};

class FinalTest

{

string title;

Date date;

public:

FinalTest(string t = "Java", Date d = Date(2014, 1, 1))

{

setDate(d);

setTitle(t);

}

void print()

{

printf("Title: %s\n", title.c\_str());

printf("Test Date: %d-%d-%d\n", date.getYear(), date.getMonth(), date.getDay());

}

void setDate(Date d) { this->date = d; }

void setTitle(string t) { this->title = t; }

};

int main()

{

FinalTest item1("C++ Test", Date(2014, 6, 2));

item1.print();

FinalTest item2("Java");

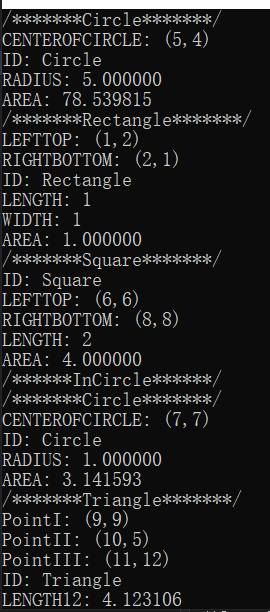
item2.print();

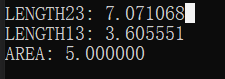
item2.setDate(Date(2014, 6, 10));

item2.print();

}

问题与思考：初始化器只能用于构造函数





#include<iostream>

using namespace std;

#define Pi 3.1415926

#include<cmath>

#include<cstdlib>

class Shape

{

string id;

public:

Shape(string i = "Shape"): id(i){}

~Shape() {}

double area() { return 0;}

void print()

{

printf("/\*\*\*\*\*\*\*Shape\*\*\*\*\*\*\*/\n");

printf("ID: %s\nAREA: %lf\n", id.c\_str(), area());

}

};

/\*\*\*\*\*\*Base Class\*\*\*\*\*\*/

class Point:public Shape

{

int x, y;

public:

Point(int a = 0, int b = 0):x(a),y(b) {}

Point operator+(const Point& p2)

{

return Point(this->x + p2.x, this->y + p2.y);

}

Point operator/(const int value)

{

return Point(this->x / value, this->y / value);

}

Point operator-(const Point& p2)

{

return Point(this->x - p2.x, this->y - p2.y);

}

int getX() { return this->x; }

int getY() { return this->y; }

};

/\*\*\*\*Extra Subordinate Class\*\*\*\*/

class Circle:public Shape

{

string id;

Point centerofcircle;

double radius;

public:

Circle(string i = "Circle",Point p = Point(0,0),double ra = 0):id(i),centerofcircle(p),radius(ra) {}

~Circle() {}

double area() { return fabs(Pi \* radius \* radius); }

void print()

{

printf("/\*\*\*\*\*\*\*Circle\*\*\*\*\*\*\*/\n");

printf("CENTEROFCIRCLE: (%d,%d)\n", centerofcircle.getX(), centerofcircle.getY());

printf("ID: %s\nRADIUS: %lf\nAREA: %lf\n", id.c\_str(),radius,area());

}

};

class Triangle:public Shape

{

string id;

Point eins;

Point zwei;

Point drei;

public:

Triangle(string i = "Triangle", Point p1 = Point(0, 0), Point p2 = Point(0, 0),Point p3 = Point(0,0)):id(i),eins(p1),zwei(p2),drei(p3) {}

~Triangle() {}

double area()

{

int x1 = eins.getX(), x2 = zwei.getX(), x3 = drei.getX();

int y1 = eins.getY(), y2 = zwei.getY(), y3 = drei.getY();

return fabs((x1 \* y2 - x1 \* y3 + x2 \* y3 - x2 \* y1 + x3 \* y1 - x3 \* y2)/2);

}

double getLength12()

{

Point temp = eins - zwei;

int deltaX = abs(temp.getX());

int deltaY = abs(temp.getY());

return fabs(sqrt(deltaX \* deltaX + deltaY \* deltaY));

}

double getLength23()

{

Point temp = zwei - drei;

int deltaX = abs(temp.getX());

int deltaY = abs(temp.getY());

return fabs(sqrt(deltaX \* deltaX + deltaY \* deltaY));

}

double getLength13()

{

Point temp = eins - drei;

int deltaX = abs(temp.getX());

int deltaY = abs(temp.getY());

return fabs(sqrt(deltaX \* deltaX + deltaY \* deltaY));

}

void print()

{

printf("/\*\*\*\*\*\*\*Triangle\*\*\*\*\*\*\*/\n");

printf("PointI: (%d,%d)\n", eins.getX(), eins.getY());

printf("PointII: (%d,%d)\n", zwei.getX(), zwei.getY());

printf("PointIII: (%d,%d)\n", drei.getX(), drei.getY());

printf("ID: %s\nLENGTH12: %lf\nLENGTH23: %lf\nLENGTH13: %lf\nAREA: %lf\n", id.c\_str(),getLength12(),getLength23(),getLength13(), area());

}

};

class Rectangle:public Shape

{

Point lefttop;

Point rightbottom;

string id;

public:

Rectangle(string i = "Rectangle",Point p1 = Point(0,0),Point p2 = Point(0,0)):id(i),lefttop(p1),rightbottom(p2){}

~Rectangle() {}

int getLength()

{

Point temp = rightbottom - lefttop;

return abs(temp.getY());

}

int getWidth()

{

Point temp = rightbottom - lefttop;

return abs(temp.getX());

}

double area()

{

return fabs(getWidth() \* getLength());

}

void print()

{

printf("/\*\*\*\*\*\*\*Rectangle\*\*\*\*\*\*\*/\n");

printf("LEFTTOP: (%d,%d)\nRIGHTBOTTOM: (%d,%d)\n", lefttop.getX(), lefttop.getY(),rightbottom.getX(),rightbottom.getY());

printf("ID: %s\nLENGTH: %d\nWIDTH: %d\nAREA: %lf\n", id.c\_str(), getLength(),getWidth(), area());

}

};

/\*\*\*\*Derived Classes - hierarchy 1\*\*\*\*\*/

class Square:public Rectangle

{

string id;

Point lefttop;

Point rightbottom;

public:

Square(string i = "Square", Point p1 = Point(0, 0), Point p2 = Point(0, 0)):id(i),lefttop(p1), rightbottom(p2) {}

~Square() {}

Circle incircle()

{

return Circle ("Circle",(lefttop + rightbottom)/2,((rightbottom - lefttop).getX()) / 2);

}

int getLength()

{

Point temp = rightbottom - lefttop;

return temp.getX();

}

double area()

{

return fabs(getLength() \* getLength());

}

void print()

{

printf("/\*\*\*\*\*\*\*Square\*\*\*\*\*\*\*/\n");

printf("ID: %s\nLEFTTOP: (%d,%d)\nRIGHTBOTTOM: (%d,%d)\nLENGTH: %d\nAREA: %lf\n", id.c\_str(), lefttop.getX(),lefttop.getY(), rightbottom.getX(),rightbottom.getY(), getLength(), area());

printf("/\*\*\*\*\*\*InCircle\*\*\*\*\*\*/\n");

incircle().print();

}

};

int main()

{

Circle("Circle",Point(5,4),5).print();

Rectangle("Rectangle",Point(1,2),Point(2,1)).print();

Square("Square",Point(6,6),Point(8,8)).print();

Triangle("Triangle", Point(9, 9), Point(10, 5), Point(11, 12)).print();

}