Министерство образования и науки Украины

Государственное высшее учебное заведение

«Приднепровская государственная академия строительства и архитектуры»

Кафедра прикладной математики

Курсовая работа по ООП

Тема: «Класс прямоугольник»

Выполнил:

Ст. гр. КН-13

Акулов Геннадий

Принял:

Власенко Ю. Е.

г. Днепропетровск-2015

По приведенной ниже таблице согласно номеру варианта выберите заданные параметры (помечены символом \*). Создайте класс, определяющий остальные параметры, площадь заданного прямоугольника и его периметр.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **№** | **A** | **B** | **C** | **a** | **d** | **x1** | **y1** | **x2** | **y2** | **x3** | **y3** | **x4** | **y4** |
| **1** | \* | \* |  | \* | \* |  |  |  |  | \* | \* |  |  |



Ах+Ву+С=0

У

Х

(х2, у2)

(х1, у1)

(х3, у3)

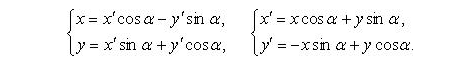
(х4, у4)

**a**

**d**

Для того, чтобы не использовать многочисленные преобразования формул вверху для нахождения всех неизвестных, можно повернуть оси координат на угол наклона заданной прямой, тогда ось Y будет перпендикулярна прямой. И остальные неизвестные точки можно будет найти с помощью простых операций сложения и вычитания из известной точки длин сторон прямоугольника.

Формулы поворота осей коодинат на заданный угол:



где, x,y-координаты старых точек; - коодринаты новых.

Преобразуем прямую к виду y = kx+b : Заданная прямая пересечет ось оY в точке

Новые коэффициенты заданной прямой: A=0(возле х) , B=1(возле y) ,

C = .

Вычисление угла наклона прямой: φ=

*Дано: A, B, a, d, ,*

*Найти: C, , ,, , , площадь и периметр прямоугольника.*

Последовательность дейсвий:

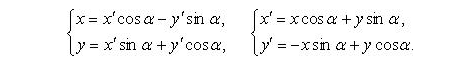
* Найдем коэффициент прямой С. Подставим в общее уравнение прямой х3 и y3, выразив из неё коэффициент С. A\*x3+B\*y3+С=0;

Получим С = -A\*x3-B\*y3.

* Повернуть оси, собственно преобразовать координаты точек х3 и у3:

Т.е. задать значения координат точки таким образом, чтобы угол наклона прямой стал прямым относительно оси Y(по формулам поворота вверху). Это действие объясняется тем, что после преобразования лечге находить координаты остальных точек, т.к. пары X-cов и и Y-ков будут равны, если находятся на одной прямой (а остальные неизвестные точки можно будет найти с помощью простых операций сложения и вычитания из известной точки длин сторон прямоугольника).

Формулы поворота осей коодинат на заданный угол:



Тут заданный угол – угол наклона прямой φ=.

* Найти значения остальных точек (с этим условием вызываются методы класса 3,4,5):
  + Точки 1
  + Точки 2
  + Точки 4
* Посчитать площадь и периметр, используя длины известных a и d.
* **Преобразовать значения всех найденных точек до поворота осей:**

Возвращаем значения всех точек в заданную систему координат, т.е. с углом наклона заданной прямой (φ =arctg(-A/B)) относительно оси оХ.

* Вывод результатов.

Математическо решение:

1)С=-A\*x3-B\*y3;

2)x3’ = x3\*cos(atan(-itsA/itsB))+y3\*sin(atan(-itsA/itsB));

y3’= -x3\*sin(atan(-itsA/itsB))+y3\*cos(atan(-itsA/itsB));

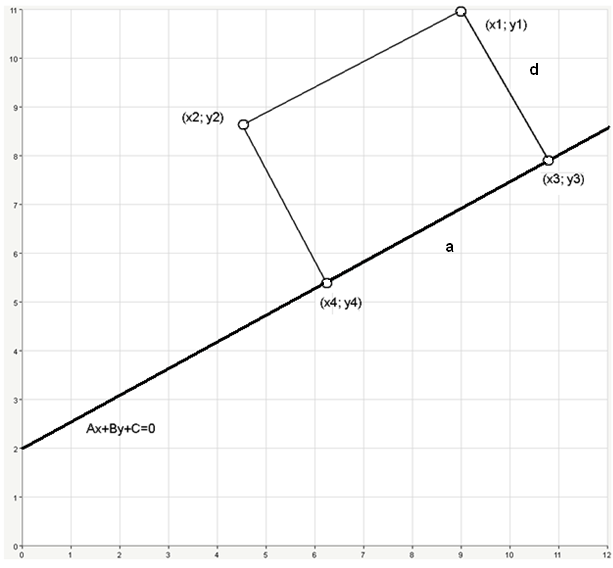
Из-за поворота некоторые координаты стали равны между собой, потому что лежат на одной прямой, которая перпендикулярна одной оси X или Y:

3) x1=x3, y1=y3+d;

4)x2=x3-a; y2=y1;

5)x4=x2; y4=y3;

6) S=a\*d; P=(a+d)\*2;



Пусть Уравнение прямой: 6\*x-11\*y+21.5=0;

A = 6; B=-11; С= 21.5; x3=10.9; y3=7.9; a=5; d=3.5;

Программа будет состоять из 2 файлов: основного (rect.cpp) и заголовочного (rect.h).

//начало заголовочного файла

#include <iostream>

using namespace std;

class Point {//коодинаты x,y

public:

void SetX(float x) { itsX = x; }

void SetY(float y) { itsY = y; }

float GetX()const { return itsX;}

float GetY()const { return itsY;}

private:

float itsX;

float itsY;

};

//конец объявления класса Point

class Rectangle{

public:

//Rectangle(A, B, a, d, x3, y3);

Rectangle::Rectangle(float, float, float ,float, float, float);//конструктор

~Rectangle () {} //деструктор

float GetTop() const { return itsTop; }

float GetLeft() const { return itsLeft; }

float GetBottom() const { return itsBottom; }

float GetRight() const { return itsRight; }

Point GetUpperLeft() const { return itsUpperLeft; }

Point GetLowerLeft() const { return itsLowerLeft; }

Point GetUpperRight() const { return itsUpperRight; }

Point GetLowerRight() const { return itsLowerRight; }

void SetTop(float top);

void SetLeft (float left);

void SetBottom (float bottom);

void SetRight (float right);

void Find\_C();

//новые коорд.точек x3, x4 из-за поворота

void Rectangle::Cord\_Calc\_New\_LowerRight();//x3,y3

void Cord\_Calc\_New\_UpperRight(float low\_rect);//x1,y1

void Cord\_Calc\_New\_UpperLeft();//x2,y2

void Cord\_Calc\_New\_LowerLeft();//x4,y4

void Ret\_Back();//посчитать значения до поворота

float GetitsC() const;

float Getits\_a()const;

float Getits\_d()const;

float GetArea() const;

float GetPerim()const;

private:

Point itsOldLowerRight;//старые коорд.заданной точки 3

Point itsUpperLeft;

Point itsUpperRight;

Point itsLowerLeft;

Point itsLowerRight;

float itsTop;

float itsLeft;

float itsBottom;

float itsRight;

//стороны прямоугольника а и d

float its\_a;

float its\_d;

//коэффициенты заданной прямой

float itsA;

float itsB;

float itsC;

};

//конец заголовочного файла

//начало основного

#include "rect.h"

#include <iostream>

#include <cmath>

using namespace std;

//-----------------------------------------------------------------------+++

/\*Конструктор, создает объект прямоугольник, инициализирует переменные,

которые заданы по условию.\*/

Rectangle::Rectangle(float A, float B, float a, float d, float x3, float y3) {

itsA=A;

itsB=B;

its\_a=a;

its\_d=d;

itsOldLowerRight.SetX(x3);//заданный x3

itsOldLowerRight.SetY(y3);//заданный x3

}

//-----------------------------------------------------------------------+++

//Вычисляет площадь прямоугольника и возвращает это значение.

float Rectangle::GetArea() const {

return (its\_a\*its\_d);

}

//-----------------------------------------------------------------------+++

//Вычисляет периметр прямоугольника и возвращает это значение.

float Rectangle::GetPerim() const {

return ( (its\_a+its\_d)\*2 );

}

//-----------------------------------------------------------------------+++

//Возвращает значение длины стороны a.

float Rectangle::Getits\_a() const {

return its\_a;

}

//-----------------------------------------------------------------------+++

//Возвращает значение длины стороны d.

float Rectangle::Getits\_d()const{

return its\_d;

}

//-----------------------------------------------------------------------+++

//Возвращает значение коэф. прямой С.

float Rectangle::GetitsC()const{

return itsC;

}

//-----------------------------------------------------------------------+++

//Найдем коэф. прямой C. Подставим в общее уравнение прямой х3 и y3, выразив из неё коэф. C

void Rectangle::Find\_C(){

float x3, y3, A, B;

x3=itsOldLowerRight.GetX();

y3=itsOldLowerRight.GetY();

A=itsA;

B=itsB;

//вычисляем C, подставив координаты точки 3 в уравнение прямой.

itsC=-A\*x3-B\*y3;

}

//-----------------------------------------------------------------------+++

/\*Преобразование нижней правой точки (x3,y3): т.е. задаёт значения этой точки, таким образом,

чтобы угол наклона прямой стал прямым, относительно осей.\*/

void Rectangle::Cord\_Calc\_New\_LowerRight(){

float x3,y3,x,y;//x3,y3,новый х,y

x3 =itsOldLowerRight.GetX();//достать заданную кодинату x3

y3 =itsOldLowerRight.GetY();//достать заданную кодинату y3

x = x3\*cos(atan(-itsA/itsB))+y3\*sin(atan(-itsA/itsB));

y= -x3\*sin(atan(-itsA/itsB))+y3\*cos(atan(-itsA/itsB));

itsRight=x;//правая точка x3

itsBottom=y;//нижняя точка y3

itsLowerRight.SetX(x);//новый x3

itsLowerRight.SetY(y);//новый y3

}

//-----------------------------------------------------------------------+++

/\*Преобразование верхней правой точки(x1,y1): т.е. задаёт значения этой точки,

когда угол наклона прямой стал прямым относительно осей.\*/

void Rectangle::Cord\_Calc\_New\_UpperRight(float low\_rect){

//x1=x3, y1=y3+d

itsUpperRight.SetX(itsRight);//x1

itsTop=itsBottom+its\_d ;//y1

itsTop -= low\_rect\*2\*(itsTop-itsBottom);

/\*если нужен нижний прямоугольник,

когда в главной программе low\_rect=1, то

расстояние вниз: верх -(верх-низ)\*2.

Если low\_rect=0, то itsTop не изменится.

\*/

itsUpperRight.SetY(itsTop);//y1

}

//-----------------------------------------------------------------------+++

/\*Преобразование верхней левой точки(x2,y2): т.е. задаёт значения этой точки,

когда угол наклона прямой стал прямым относительно осей.\*/

void Rectangle::Cord\_Calc\_New\_UpperLeft(){

//x2=x3-a, y2=y1;

itsLeft=itsRight-its\_a;

itsUpperLeft.SetX(itsLeft);//x2

itsUpperLeft.SetY(itsTop);//новый y2

}

//-----------------------------------------------------------------------+++

/\*Преобразование нижней левой точки(x4,y4): т.е. задаёт значения этой точки,

когда угол наклона прямой стал прямым относительно осей.\*/

void Rectangle::Cord\_Calc\_New\_LowerLeft(){

//x4=x2, y4=y3;

itsLowerLeft.SetX(itsTop);//x4

itsLowerLeft.SetY(itsBottom);//новый y4

}

//-----------------------------------------------------------------------+++

/\*Преобразуем значения всех точек в заданную систему координат, т.е. с углом наклона заданной прямой (Fi=atan(-itsA/itsB))\*/

void Rectangle::Ret\_Back(){ //посчитать значения координат до поворота

float x,y;

float sinP = sin(atan(-itsA/itsB));

float cosPr = cos(atan(-itsA/itsB));

x=itsRight\*cosPr-itsTop\*sinP;//x1

y=itsRight\*sinP+itsTop\*cosPr;//y1

itsUpperRight.SetX(x);//преобразовать обратно x1

itsUpperRight.SetY(y);//преобразовать обратно y1

x=itsLeft\*cosPr-itsTop\*sinP;//x2

y=itsLeft\*sinP+itsTop\*cosPr;//y2

itsUpperLeft.SetX(x);//преобразовать обратно x2

itsUpperLeft.SetY(y);//преобразовать обратно y2

x=itsRight\*cosPr-itsBottom\*sinP;//x3

y=itsRight\*sinP+itsBottom\*cosPr;//y3

itsLowerRight.SetX(x);//преобразовать обратно x3

itsLowerRight.SetY(y);//преобразовать обратно y3

x=itsLeft\*cosPr-itsBottom\*sinP;//x4

y=itsLeft\*sinP+itsBottom\*cosPr;//y4

itsLowerLeft.SetX(x);//преобразовать обратно x4

itsLowerLeft.SetY(y);//преобразовать обратно y4

}

void main(){

float low\_rect=0;

// low\_rect=1; //если нужен нижний прямоугольник

//Rectangle(A, B, a, d, x3, y3);

Rectangle MyRectang( 6, -11, 5, 3.5, 10.9, 7.9);

MyRectang.Find\_C(); //посчитать коэф. прямой С

MyRectang.Cord\_Calc\_New\_LowerRight();//установить в новых осях точку 3

MyRectang.Cord\_Calc\_New\_UpperRight(low\_rect);//преобразовать точку 1

MyRectang.Cord\_Calc\_New\_UpperLeft();//преобразовать точку 2

MyRectang.Cord\_Calc\_New\_LowerLeft();//преобразовать точку 4

MyRectang.Ret\_Back();//преобразовать значения всех точек до поворота

float a = MyRectang.Getits\_a();

float d = MyRectang.Getits\_d();

float C = MyRectang.GetitsC();

float Area = MyRectang.GetArea();

float Per = MyRectang.GetPerim();

cout<<" Results : \n\n";

cout<<" Known : \n\n";

cout <<" a: " << a << '\n';

cout <<" d: " << d << "\n\n";

cout<<" Calculated : \n\n";

cout <<" C: " << C << '\n';

cout <<" Area: " << Area << '\n';

cout << " Perimeter: " << Per << "\n\n";

cout << " Upper Rigth X Coordinate (x1): ";

cout << MyRectang.GetUpperRight().GetX()<<'\n';

cout << " Upper Right Y Coordinate (y1): ";

cout << MyRectang.GetUpperRight().GetY()<<"\n";

cout << " Upper Left X Coordinate (x2) : ";

cout << MyRectang.GetUpperLeft().GetX()<<'\n';

cout << " Upper Left Y Coordinate (y2) : ";

cout << MyRectang.GetUpperLeft().GetY()<<'\n';

cout << " Lower Rigth X Coordinate (x3): ";

cout << MyRectang.GetLowerRight().GetX()<<'\n';

cout << " Lower Right Y Coordinate (y3): ";

cout << MyRectang.GetLowerRight().GetY()<<'\n';

cout << " Lower Rigth X Coordinate (x4): ";

cout << MyRectang.GetLowerLeft().GetX()<<'\n';

cout << " Lower Left Y Coordinate (y4) : ";

cout << MyRectang.GetLowerLeft().GetY()<<'\n';

system ("pause");

}

//конец основного файла

Полученные результаты:

**Верхний прямоугольник**

Results :

Known :

a: 5

d: 3.5

Calculated :

C: 21.5

Area: 17.5

Perimeter: 17

Upper Rigth X Coordinate (x1): 9.22402

Upper Right Y Coordinate (y1): 10.9726

Upper Left X Coordinate (x2) : 4.83454

Upper Left Y Coordinate (y2) : 8.57837

Lower Rigth X Coordinate (x3): 10.9

Lower Right Y Coordinate (y3): 7.9

Lower Rigth X Coordinate (x4): 6.51052

Lower Left Y Coordinate (y4) : 5.50574

Для продолжения нажмите любую клавишу . . .

**Нижний прямоугольник:**

Results :

Known :

a: 5

d: 3.5

Calculated :

C: 21.5

Area: 17.5

Perimeter: 17

Upper Rigth X Coordinate (x1): 12.576

Upper Right Y Coordinate (y1): 4.82737

Upper Left X Coordinate (x2) : 8.1865

Upper Left Y Coordinate (y2) : 2.4331

Lower Rigth X Coordinate (x3): 10.9

Lower Right Y Coordinate (y3): 7.9

Lower Rigth X Coordinate (x4): 6.51052

Lower Left Y Coordinate (y4) : 5.50574

Для продолжения нажмите любую клавишу . . .

**Результаты совпадают с графиком.**