Разработать ООП для вычисления декартовых координат вершин правильного треугольника с заданной длиной стороны, центр которого находится в точке (0;0), а одна из вершин лежит на оси X. Длина стороны треугольника должна передаваться программе через аргумент командной строки, а полученные координаты должны отображаться через стандартный вывод. Программная реализация вычислений должна быть основана на разработке класса точки с приватными полями её декартовых координат и публичными методами доступа к ним и конструктором инициализации их значений. Кроме того нужно предусмотреть статический метод конструирования точки по её полярным координатам.

1: #include <math.h>

2: #include <iostream>

3: using namespace std;

4:

5: class Point2

6: {

7: private:

8: float x;

9: float y;

10: Point2(float \_x, float \_y) : x(\_x), y(\_y) { };

11: public:

12: float getx() { return x; };

13: float gety() { return y; };

14: static Point2 Polar(float, float);

15: Point2() { x=y=0.0; };

16: };

17:

18:

19: inline Point2 Point2::Polar(float R, float F)

20: {

21: Point2 p(R\*cos(F), R\*sin(F));

22: return p;

23: }

24:

25: int main(int argc, char \*argv[])

26: {

27: float edge = 1.0;

28: float angle = 0.0;

29: float pi;

30: pi = acos(-1.0);

31: float radius;

32: int i = 0;

33: Point2 vertex[3];

34: if(argc > 1)

35: edge = atof(argv[1]);

36: radius = edge / (2\*sin(pi/3.0));

37: while (i<3)

38: {

39: vertex[i] = Point2::Polar(radius, angle);

40: angle += ((2.0\*pi)/3.0);

41: i++;

42: }

43: do

44: {

45: --i;

46: cout << vertex[i].getx() << ';' << vertex[ i ].gety() << endl;

47: } while(i > 0);

48: return 0;

49: }

Разработать ООП для вычисления среднего расстояния от каждой точки заданного набора до остальных точек этого набора. Декартовы координаты точек набора должны передаваться программе парами целых чисел через аргументы командной строки. Полученные значения средних расстояний должны отображаться строками стандартного вывода. Программная реализация вычислений должна быть основана на разработке класса точки с приватными полями ее декартовых координат и конструктором инициализации их значений, публичными методами доступа к ним, а также дружественной функции вычисления расстояния между 2-мя точками. Кроме того, в программе должно быть предусмотрено динамическое распределение памяти для всех точек заданного набора и массива их адресов.

1: #include <stdio.h>

2: #include <math.h>

3:

4: class Point2

5: {

6: private:

7: int x;

8: int y;

9: public:

10: Point2(int \_x, int \_y) : x(\_x), y(\_y) { };

11: int getx() { return x; };

12: int gety() { return y; };

13: friend double distance(Point2\*, Point2\*);

14: };

15:

16: double distance(Point2\* r, Point2\* q)

17: {

18: double d2=(r->x-q->x)\*(r->x-q->x)+(r->y-q->y)\*(r->y-q->y);

19: return (sqrt(d2));

20: }

21:

22: int main(int argc, char\* argv[])

23: {

24: double sum;

25: Point2\*\* p;

26: Point2\* t;

27: int x,y;

28: int i=0;

29: int j=0;

30: if(argc < 2)

31: return(-1);

32: p = new Point2\*[argc];

33: while(++i < argc)

34: {

35: sscanf(argv[i], "(%d%\*c%d)", &x, &y);

36: p[j]=new Point2(x,y);

37: j++;

38: }

39: p[j]=NULL;

40: for (i=0; p[i] != NULL; i++)

41: {

42: sum = 0.0;

43: t = p[i];

44: for (j=0; p[ j ] != NULL; j++)

45: sum += distance(t, p[j]);

46: printf("Dcp(%d;%d)=%f\n", t->getx(), t->gety(), sum/(argc-2));

47: }

48:

49: for(j=0; p[j] != NULL; j++)

50: delete p[j];

51: delete [ ]p;

52: return 0;

53: }