ФИО: Ребдев Павел Александрович
Группа: 5130904/30008
Лабораторная работа: «Простые классы»»

Постановка задачи

Разработать детальные требования и тест план для следующей задачи:

Point. Написать программу, принимающую на вход координаты точек, выводящую расстояние между двумя производными точками, находятся ли эти точки внутри окружность и самая близкую к началу координат точку

Детальные требования

1. Координаты всех точек должны быть заданы корректно:
   1.1. x и y — числа. Если x или y хотя бы одной точки не являются числами, то выводится сообщение об ошибке и программа завершается с кодом возврата 1
2. Координаты точек заданы корректно:
   2.1. Если x и y всех точек заданы корректно, то программа должна завершится с выводом в консоль сообщения, содержащего номера двух произвольных точек, их координаты, расстояние между ними, находятся ли эти точки внутри окружность и самая близкую к началу координат точку

Тест-план

Проверка детальных требований с помощью тест-плана:

#	Описание	Результат

1.1	x и y — числа. Если x или y хотя бы одной точки не являются числами, то выводится сообщение об ошибке и программа завершается с кодом возврата 1	**Input:**
		`1.2 2.3 3.4 4.3 a.0 12.3 0.0 0.0`
		Expected:
		`Bad input!`

		Input:
		`1.2 2.3 3.dfss 4.3 0.0 12,3 0.0 0.0`
		Expected:
		`Bad input!`

2.1	Если x и y всех точек заданы корректно, то программа должна завершится с выводом в консоль сообщения, содержащего номера двух произвольных точек, их координаты, расстояние между ними, находятся ли эти точки внутри окружность и самая близкую к началу координат точку	**Input:**
		`50.94 63.40`
		`8.35 76.53`
		`7.34 90.53`
		`22.34 93.26`
		`3.63 70.11`
		`54.51 46.74`
		`62.77 88.31`
		`19.72 75.69`
		`90.62 66.98`
		`0.0 0.0`
		Expected:
		`Number of points: 0 7`
		`x and y cordinats: 50.94 63.4  19.72 75.69`
		`Points is equal: 0`
		`Distance between points: 33.5519`
		`Points is in circle with radius 93: 1 1`
		`The point closest to the origin: 3.63 70.11`

		Input:
		`383.886 777.915`
		`793.335 386.492`
		`649.421 362.27`
		`690.59 763.926`
		`540.426 172.736`
		`211.368 567.429`
		`782.530 862.123`
		`67.135 929.802`
		`22.58 69.167`
		`393.456 11.42`
		`0.0 0.0`
		Expected:

		Number of points: 3 8 x and y cordinats: 690.59 763.926 22.58 69.167 Points is equal: 0 Distance between points: 963.809 Points is in circle with radius 87: 0 1 The point closest to the origin: 22.58 69.167

Исходные тексты программы

Файлы с исходными текстами лабораторной работы (полагаем <ROOT> для папки в котором располагаются исходные тексты):

./<ROOT>/main.cpp

```
#include <iostream>
#include <cstddef>
#include <ctime>
#include <limits>
#include "Point.h"
#include "pointFunction.hpp"

int main()
{
  Point * pointArr = nullptr;
  Point * newArr = nullptr;
  size_t numberOfPoints = 0;
  do
  {
    newArr = new Point[numberOfPoints + 1];
    for (size_t i = 0; i < numberOfPoints; ++i)
    {
      newArr[i].setX(pointArr[i].getX());
      newArr[i].setY(pointArr[i].getY());
    }

    try
    {
      input(newArr[numberOfPoints]);
    }
    catch (const std::logic_error & e)
    {
      delete[] pointArr;
      delete[] newArr;

      std::cerr << e.what();
      return 1;
    }

    delete[] pointArr;
    pointArr = newArr;
    newArr = nullptr;

    numberOfPoints += 1;
     }  while  ((pointArr[numberOfPoints  -  1].getX()  !=  0.0)  ||
(pointArr[numberOfPoints - 1].getY() != 0.0));

  srand(time(NULL));
```

```
    size_t numberOfFirstPoint = (rand() % (numberOfPoints - 1)),
numberOfSecondPoint = (rand() % (numberOfPoints - 1));
  while (numberOfSecondPoint == numberOfFirstPoint)
  {
    numberOfSecondPoint = (rand() % (numberOfPoints - 1));
  }
   std::cout << "Number of points: " << numberOfFirstPoint << " " <<
numberOfSecondPoint << '\n';

  std::cout << "x and y cordinats: ";
  output(pointArr[numberOfFirstPoint]);
  std::cout << " ";
  output(pointArr[numberOfSecondPoint]);
  std::cout << '\n';

          std::cout     <<     "Points     is     equal:     "     <<
pointArr[numberOfFirstPoint].isEqual(pointArr[numberOfSecondPoint])   <<
'\n';

         std::cout    <<    "Distance    between    points:    "    <<
pointArr[numberOfFirstPoint].getDistance(pointArr[numberOfSecondPoint])
<< '\n';

  size_t radius = (rand() % 100);
   std::cout << "Points is in circle with radius " << radius << ": " <<
isPointInCircle(pointArr[numberOfFirstPoint], radius);
   std::cout  <<  "  "  <<  isPointInCircle(pointArr[numberOfSecondPoint],
radius) << '\n';

  Point startPoint(0.0, 0.0);
  size_t IndexOfNearestPoint = 0;
  double minDistance = std::numeric_limits< double >::max();

  for (size_t i = 0; i < (numberOfPoints - 1); ++i)
  {
    double distance = startPoint.getDistance(pointArr[i]);
    if (distance < minDistance)
    {
      minDistance = distance;
      IndexOfNearestPoint = i;
    }
  }

  std::cout << "The point closest to the origin: ";
  output(pointArr[IndexOfNearestPoint]);
  std::cout << '\n';

  return 0;
}
```

./<ROOT>/Point.h

```
#ifndef POINT_H
#define POINT_H
class Point
```

```
{
  public:
    Point();
    Point(double x, double y);
    Point(Point & point);
    ~Point();

    void setX(double x);
    void setY(double y);
    double getX() const;
    double getY() const;

    bool isEqual(Point point);
    double getDistance(Point point);
    void move(double distance);
  private:
    double x_, y_;
};
#endif
```

./<ROOT>/Point.cpp

```
#include "Point.h"
#include <cmath>
#include <iostream>

Point::Point():
 x_(0.0),
 y_(0.0)
 {};

Point::Point(double x, double y):
 x_(x),
 y_(y)
 {};

Point::Point(Point & point):
 x_(point.getX()),
 y_(point.getY())
 {};

Point::~Point()
{
 x_ = 0.0;
};

void Point::setX(double x)
{
 x_ = x;
};

void Point::setY(double y)
{
 y_ = y;
};

double Point::getX() const
{
```

```
 return x_;
};
double Point::getY() const
{
 return y_;
};

bool Point::isEqual(Point point)
{
 return (x_ == point.getX()) && (y_ == point.getY());
};
double Point::getDistance(Point point)
{
   return std::sqrt((x_ - point.getX()) * (x_ - point.getX()) + (y_ - point.getY()) * (y_ - point.getY()));
};
void Point::move(double distance)
{
 x_ += distance;
 y_ += distance;
};
```

./<ROOT>/pointFunction.hpp

```
#ifndef POINTFUNCTION_HPP
#define POINTFUNCTION_HPP
#include <cstddef>
#include "Point.h"

bool isPointInCircle(const Point & point, size_t radius);
void input (Point & point);
void output (const Point & point);
#endif
```

./<ROOT>/pointFunction.cpp

```
#include "pointFunction.hpp"
#include <iostream>
#include <cmath>

bool isPointInCircle(const Point & point, size_t radius)
{
  double x = point.getX(), y = point.getY();
  return std::sqrt(x * x + y * y) <= radius;
};

void input (Point & point)
{
  double x = 0.0, y = 0.0;
  std::cin >> x >> y;
  if (!std::cin)
  {
    throw std::logic_error("Bad input!");
  }
  point.setX(x);
  point.setY(y);
};

void output (const Point & point)
```

```
{
  std::cout << point.getX() << " ";
  std::cout << point.getY() << " ";
};
```