|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ *Робототехника и комплексная автоматизация*

КАФЕДРА *Системы автоматизированного проектирования (РК-6)*

**ОТЧЕТ О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ**

по дисциплине: «Объектно-ориентированное программирование»

|  |  |  |
| --- | --- | --- |
| Студент |  | Камалов Антон Павлович |
| Группа |  | РК6-25Б |
| Тип задания |  | Лабораторная работа №2 |
| Вариант |  | D08 |

Студент **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_ Камалов А. П. \_**

*подпись, дата фамилия, и.о.*

Преподаватель **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_ Родионов С. В. \_\_**

*подпись, дата фамилия, и.о.*

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Москва, 2023 г.*

# Вариант задания

# Разработать объектно-ориентированную программу поиска пары наиболее отдаленных точек из любого заданного набора точек на плоскости. Декартовые координаты всех его точек должны быть специфицированы парами целых чисел в аргументах командной строки вызова программы. Координаты полученной пары наиболее отдаленных точек и расстояние между ними должны отображаться строкой потока стандартного вывода. В программе должен быть реализован класс точки с приватными полями для ее декартовых координат, публичными методами доступа к ним и конструктором инициализации их значений, а также метод для вычисления расстояния до любой другой заданной точки. Кроме того, в программе должно быть предусмотрено динамическое распределение памяти для всех точек заданного набора и массива их адресов.

# Описание программы

**Входные данные**: целочисленные координаты точек из аргументов командной строки.

**Выходные данные**: наибольшее расстояние и координаты соответствующих точек.

**Класс Point**

*Информационные поля:*

\_x (float), \_y (float) – координаты точки;

*Конструкторы и деструкторы:*

Point (float, float) – инициализации;

Point () – по умолчанию;

*Методы:*

float getX() – публичный метод доступа к приватному полю x;

float getY() – публичный метод доступа к приватному полю y;

double distance(Point\*) – метод вычисления расстояния между точками;

Программа принимает на вход целочисленные координаты точек из аргументов командной строки. После вычисляет расстояние между всеми парами точек и сохраняет самую длинную дистанцию и координаты соответствующих точек. Расстояние между двумя точками вычисляется по формуле:

, где  – координаты начальной точки,  *–* координаты конечной точки. Затем программа выводит координаты найденных точек и расстояние между ними на экран.

**Список литературы**

1. Лекционный и семинарский материалы
2. Бьёрн Страуструп «Программирование: принципы и практика с использованием С++» 2 издание.
3. Бьёрн Страуструп «Язык программирования С++» второе дополненное издание

# Приложение 1

# Текст программы

|  |  |
| --- | --- |
| 01:  02:  03:  04:  05:  06:  07:  08:  09:  10:  11:  12:  13:  14:  15:  16:  17:  18:  19:  20:  21:  22:  23:  24:  25:  26:  27:  28:  29:  30:  31:  32:  33:  34:  35:  36:  37:  38:  39:  40:  41:  42:  43:  44:  45:  46:  47:  48:  48:  49:  50:  51:  52:  53: | #include <iostream>  #include <cmath>  #include <cstdio>  class Point {  private:  int \_x, \_y;  public:  Point (int x, int y): \_x(x),\_y(y) {};  Point(): \_x(0), \_y(0) {};  int getX() {return \_x;}  int getY() {return \_y;}  double distance(Point\*);  };  // Расстояние между двумя точками  double Point::distance(Point\* q){  double \_d = sqrt(pow(q->getX() - \_x, 2) + pow(q->getY() - \_y, 2));  return \_d;  }  int main(int argc, char\*\* argv) {  double d = -1.0, local\_d;  Point \*\*p;  Point \*f, \*s;  int x, y;  int i, j;  if (argc < 3) {  std::cout << "Invalid count of arguments\n";  return -1;  }  p = new Point\*[argc];  for (i = 1, j = 0; i < argc; i++, j++){  sscanf(argv[i],"%d%\*c%d", &x, &y);  p[j] = new Point(x, y);  }  for (i = 0; i < argc-1; i++){  for (j = i+1; j < argc - 1; j++){  local\_d = p[i]->distance(p[j]);  if (local\_d > d){  d = local\_d;  f = p[i];  s = p[j];  }  }  }  printf("The longest distance is between points with coordinates (%d;%d) and (%d;%d) - %.3f\n", f->getX(), f->getY(), s->getX(), s->getY(), d);  // Очищение массива указателей  for (i = 0; i < argc-1; i++){  delete p[i];  }  delete []p;  } |

# Приложение 2

Результаты работы программы

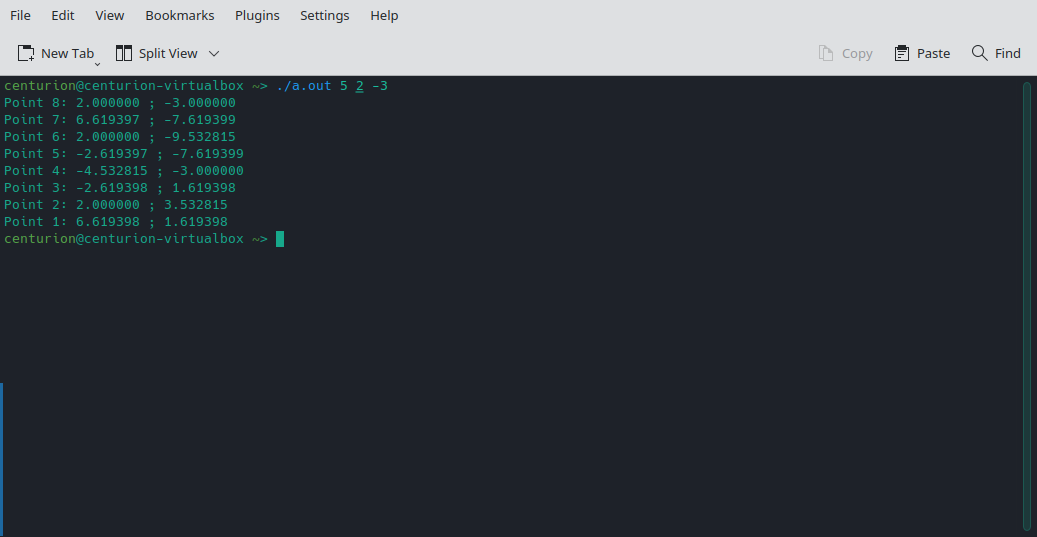


Рисунок 1. Выполнение программы с корректными аргументами

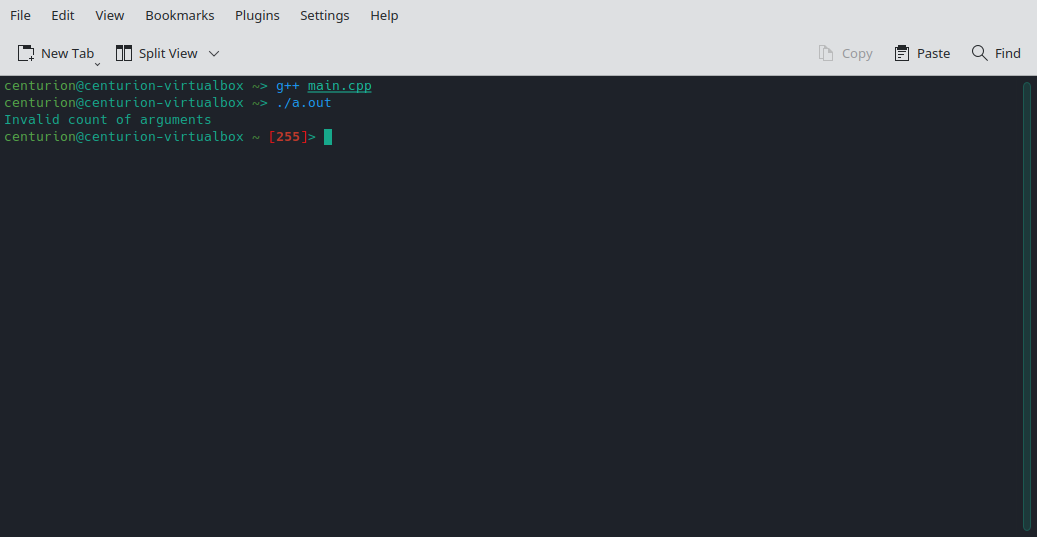


Рисунок 2. Выполнение программы с недостаточным количеством аргументов

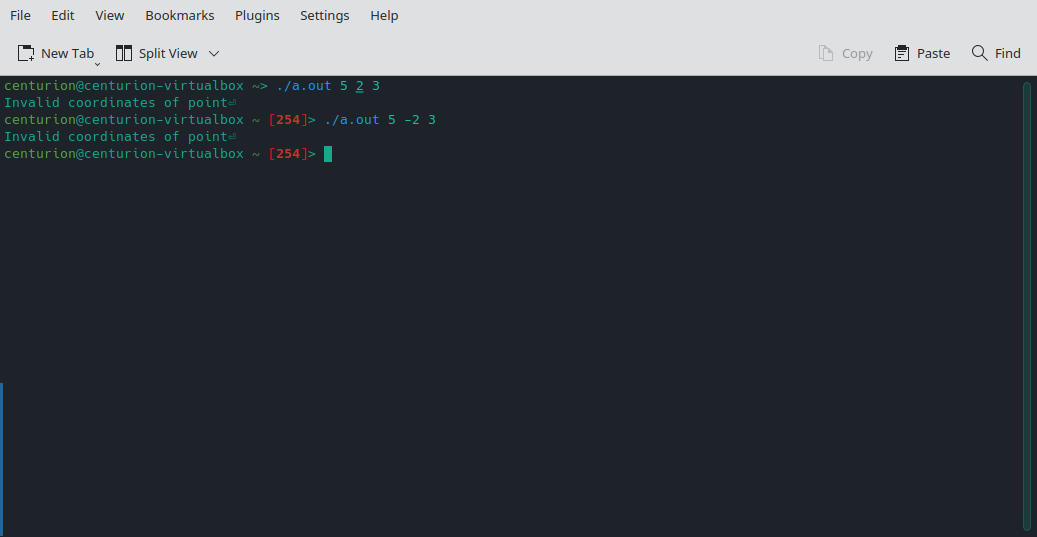


Рисунок 3. Выполнение программы с неправильным расположением центра правильного восьмиугольника