ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ ИМ. Р.Е. АЛЕКСЕЕВА

ИНСТИТУТ РАДИОЭЛЕКТРОНИКИ И ИНФОРМАЦИОННЫХ

ТЕХНОЛОГИЙ

Кафедра «Информационные радиосистемы»

КУРСОВАЯ РАБОТА

«Набор содержит геометрические точки, заданные в полярных координатах (R, alpha). Определить прямоугольник в плоскости Oxy, в который попадают все указанные точки x=R*cos(a), y=R*sin(a). Стороны прямоугольника выбрать ориентированными вдоль координатных осей.»

Выполнил:

студент группы 24-Рз И. И. Иванов

Проверил:

доцент кафедры ИРС С. Б. Сидоров

Нижний Новгород

2025

Содержание

Введение

В работе рассматривается решение задачи разработки прикладной программы обработки набора геометрических точек, заданных в полярных координатах *(R, α)*. Каждая точка преобразуется в декартовы координаты по формулам $x = R \cdot \cos(\alpha)$ и $y = R \cdot \sin(\alpha)$. Целью разработки является определение прямоугольника на плоскости Oxy, ориентированного вдоль координатных осей, внутри которого располагаются все указанные точки — то есть вычисление минимальных и максимальных значений координат *x* и *y*.

Одним из способов упрощения решения данной задачи является использование отдельных функций, отвечающих за преобразование координат и анализ полученного множества точек. Такой подход позволяет декомпозировать задачу на ряд последовательных, более простых операций. Применение модульного подхода уменьшает затраты на отладку и повышает надёжность конечной программы. Важным аспектом также является использование структурированных типов данных для корректного представления точек и их координат в программной модели.

В начале основной части отчёта приводится точная формализованная постановка задачи с указанием основного набора действий, которые должна обеспечивать прикладная программа: ввод исходных данных, вычисление декартовых координат, а также определение минимальных и максимальных значений x и y, формирующих границы прямоугольника.

В руководстве пользователя раскрывается назначение программы, её функциональные возможности и реализуемые операции. Подробно описываются правила работы: способы ввода исходных данных, формат отображения найденного прямоугольника и примеры взаимодействия пользователя с программой.

В руководстве программиста рассматриваются вопросы внутренней организации программы, включая перечень функций и схему их взаимодействия,

а также наиболее важные алгоритмы обработки — такие как преобразование полярных координат и вычисление ограничивающего прямоугольника.

В заключении формулируются выводы о полноте решения поставленной задачи и приводятся возможные направления развития программы, такие как расширение форматов исходных данных или добавление вычисления дополнительных характеристик множества точек. Также включена техническая информация, содержащая листинги программного кода.

1. Постановка задачи

Рассматривается модель данных, описывающих множество геометрических точек, заданных в полярной системе координат. Каждая точка представлена совокупностью свойств: расстоянием до начала координат R и углом α, определяющим направление на точку. В области программной реализации модель отдельной точки имеет вид структурированного типа данных, содержащего пару численных значений.

Рассматривается набор точек с конечным количеством элементов. Требуется получить программную реализацию заданной обработки такого набора структурированных данных.

Все данные, необходимые для обработки, запрашиваются у пользователя: количество точек, а также значения R и α (в градусах) для каждой из них. Эти входные данные являются исходной информацией, необходимой для корректного выполнения последующей обработки.

Обработка должна быть реализована отдельной функцией. При этом вся необходимая информация для выполнения вычислений передаётся в функцию через список аргументов. Функция выполняет преобразование полярных координат в декартовы и определяет минимальные и максимальные значения координат x и y, формирующие оси ориентированный ограничивающий прямоугольник. Результат обработки — границы прямоугольника — также должен передаваться из функции через список аргументов или возвращаемое значение.

Полученные результаты обработки должны быть выведены на монитор, то есть на стандартное устройство вывода.

Задан набор точек в полярных координатах. Требуется определить прямоугольник, ориентированный вдоль координатных осей, в который попадают все точки. Для этого необходимо преобразовать каждую точку из

полярной формы в декартову по формулам $x = R \cdot cos(\alpha)$ и $y = R \cdot sin(\alpha)$, после чего вычислить минимальные и максимальные значения x и y, определяющие стороны искомого прямоугольника.

2. Руководство пользователя

Программа предназначена для обработки набора геометрических точек, заданных в полярных координатах, в режиме диалога с пользователем.

Программа позволяет пользователю задать количество точек, ввести значения R и α (в градусах) для каждой точки, после чего автоматически вычисляет ограничивающий прямоугольник — минимальные и максимальные значения координат x и y, полученных после преобразования точек в декартову систему координат.

Программа является интерактивным консольным приложением. Весь диалог с пользователем происходит в текстовом режиме. Результаты обработки выводятся на экран монитора.

Запуск программы осуществляется либо вводом в командной строке имени исполняемого файла программы, полученного в результате компиляции, с последующим нажатием клавиши Enter, либо иным способом, зависящим от используемой операционной системы.

После запуска программы на экране появляется краткое информационное сообщение:

Polar points bounding rectangle program!

После информационного сообщения пользователю предлагается указать количество точек, которые будут вводиться:

Enter number of points:

После ввода количества точек для каждой точки последовательно выводится запрос на ввод её данных. Пример диалога:

Enter points (R α_deg), one per line:

Point 1:

Пользователь должен ввести пару значений: расстояние R и угол α в градусах.

Данный процесс повторяется для каждой точки до тех пор, пока не будут введены данные для всего набора.

После завершения ввода всех точек программа автоматически выполняет обработку: каждая точка преобразуется по формулам:

$x = R \cdot \cos(\alpha),\ y = R \cdot \sin(\alpha),$

после чего определяются минимальные и максимальные значения координат x и y, формирующие оси ориентированный ограничивающий прямоугольник.

После завершения обработки на экран выводятся результаты в следующем виде:

Bounding rectangle (axis-aligned):

left =

right =

bottom=

top =

Дополнительно выводятся ширина, высота и площадь прямоугольника:

Width = Height =

Area =

Таким образом отображаются:

- минимальное значение координаты x,
- максимальное значение координаты x,
- минимальное значение y,
- максимальное значение y,
- вычисленные геометрические характеристики прямоугольника.

Если точки не были введены или данные некорректны, выводится сообщение:

No points or invalid rectangle.

После завершения вывода результатов программа завершает свою работу.

3. Руководство программиста

3.1 Структура программы

Прикладная программа разработана с использованием принципов императивного программирования. Она представляет собой совокупность взаимодействующих функций, каждая из которых выполняет строго определённую часть обработки данных. Структура программы представлена на рисунке 1.

Рисунок 1: Структура программы

Программа состоит из следующих функций:

1. *main* — основная функция приложения; осуществляет общий алгоритм работы: вызов ввода данных, обработку точек и вывод результата.

2. *input* — функция ввода набора точек, представленных в полярных координатах. Выполняет интерактивный диалог с пользователем и сохраняет введённые значения во внутреннюю структуру данных.

3. *process_compute_rect* — функция обработки данных. Преобразует каждую точку из полярной формы в декартову и вычисляет границы ограничивающего прямоугольника.

4. *output* — функция вывода результата на экран: отображает найденные границы прямоугольника, а также вычисляет и выводит его ширину, высоту и площадь.

3.2 Структуры данных

Для представления результата обработки используется структурный тип данных rect.

Ограничивающий прямоугольник определяется четырьмя параметрами:

• минимальная координата x (min_x),

• максимальная координата x (max_x),

• минимальная координата y (min_y),

• максимальная координата y (max_y).

Определение структурного типа данных:

```
typedef struct rect {
    double min_x;
    double max_x;
    double min_y;
    double max_y;
    bool valid() const { return min_x <= max_x; }
} rect;
```

Для хранения исходного набора точек используется вектор:

```
std::vector<std::pair<double,double>> points;
```

Каждый элемент содержит:

• первое значение — расстояние R,

• второе значение — угол α в градусах.

Выполнение преобразования в декартову систему координат выполняется в функции process_compute_rect.

3.3 Алгоритм вычисления ограничивающего прямоугольника

Определение ограничивающего прямоугольника сводится к преобразованию каждой полярной точки в координаты (x, y) и поиску минимальных и максимальных значений этих координат.

Алгоритм имеет следующую последовательность действий:

1. Инициализировать границы прямоугольника:

    • $min\_x, min\_y = +\infty$,

    • $max\_x, max\_y = -\infty$.

2. Для каждой точки входного набора:

    • считать значения R и α;

    • преобразовать α из градусов в радианы;

    • вычислить декартовы координаты:

        • $x = R \cdot cos(\alpha)$

        • $y = R \cdot sin(\alpha)$

3. Обновить границы прямоугольника:

    • *если* $x < min\_x \rightarrow$ *обновить* min_x;

    • *если* $x > max\_x \rightarrow$ *обновить* max_x;

    • *если* $y < min\_y \rightarrow$ *обновить* min_y;

    • *если* $y > max\_y \rightarrow$ *обновить* max_y.

4. После обработки всех точек заполнить структуру rect полученными значениями.

5. Функция output дополнительно вычисляет и выводит:

    • ширину прямоугольника (max_x − min_x),

    • высоту (max_y − min_y),

• площадь.

6. В случае отсутствия данных или некорректного результата функция *rect::valid()* позволяет определить ошибку и вывести соответствующее сообщение.

Заключение

В данной работе задача разработки прикладной программы для обработки набора измерений, представленных в виде полярных координат, была решена с использованием принципов императивного программирования. Реализованная программа выполняет ввод данных от пользователя, преобразование измерений из полярной формы в декартову и вычисление осево-ориентированного минимального прямоугольника, ограничивающего все полученные точки.

На основании проведённой отладки и испытаний с применением контрольных примеров можно сделать вывод, что разработанная прикладная программа корректно и в полном объёме решает поставленную задачу — определение минимального и максимального значений координат (X и Y) и формирование границ ограничивающего прямоугольника для заданного набора измерений.

Список литературы

1. Керниган, Брайан У., Ритчи, Деннис М. Язык программирования C, 2-е издание. [пер. с анг.] / Б.У. Керниган, Д.М. Ритчи – М.: Вильямс, 2007.
2. Павловская, Т.А. C/C++. Программирование на языке высокого уровня: учебник для ВУЗов / Т.А. Павловская. – СПб.: Питер, 2009.
3. Орлов, С.А. Технологии разработки программного обеспечения. учеб. пособие. 2-е изд./ С.А. Орлов, – СПб.: Питер, 2003. – 480 с.: ил.
4. Борисенко, В.В. Основы программирования / В.В.Борисенко, – Интернет-университет информационных технологий – ИНТУИТ.ру, 328 стр. – 2005 г.
5. Шилдт, Г. Полный справочник по C: учеб. пособие / Г. Шилдт. – 4-е изд. – М.: Изд. дом "Вильямс", 2008.
6. Костюкова, Н.И. Язык Си и особенности работы с ним / Н.И. Костюкова, Н.А. Калинина – Интернет-университет информационных технологий – ИНТУИТ.ру, 208 стр. – 2006 г.

Приложение А

Заголовочный файл – header.h

```
#ifndef EX16_RECT_H
#define EX16_RECT_H

#include <vector>
#include <utility>
#include <iostream>
#include <cmath>
#include <iomanip>
#include <limits>

typedef struct rect {
      double min_x;
      double max_x;
      double min_y;
      double max_y;
      bool valid() const { return min_x <= max_x; }
} rect;

rect process_compute_rect(const std::vector<std::pair<double,double>> &points);

bool input(std::vector<std::pair<double,double>> &points);

bool output(const rect &result);

#endif // EX16_RECT_H
```

Основная программа – main.cpp

```
#include "header.h"

rect process_compute_rect(const std::vector<std::pair<double,double>> &points){

    const double deg2rad = M_PI / 180.0;

    rect r;
    r.min_x = std::numeric_limits<double>::infinity();
    r.max_x = -std::numeric_limits<double>::infinity();
    r.min_y = std::numeric_limits<double>::infinity();
    r.max_y = -std::numeric_limits<double>::infinity();

    for (const std::pair<double,double> &p : points){
        double R = p.first;
        double alpha = p.second;

        alpha *= deg2rad;

        double x = R * std::cos(alpha);
        double y = R * std::sin(alpha);

        if (x < r.min_x) r.min_x = x;
        if (x > r.max_x) r.max_x = x;
        if (y < r.min_y) r.min_y = y;
        if (y > r.max_y) r.max_y = y;
```

```
    }

    return r;
}

bool input(std::vector<std::pair<double,double>> &points){

    int n;
    std::cout << "Enter number of points: " << std::endl;

    if (!(std::cin >> n) || n <= 0) {
        std::cerr << "Invalid number of points" << std::endl;
        return false;
    }

    points.clear();
    points.reserve(n);
    std::cout << "Enter points (R alpha_deg), one per line:" << std::endl;

    for (int i = 0; i < n; ++i) {
        double r, alpha_deg;
        std::cout << "Point " << (i+1) << ": " << std::endl;
        if (!(std::cin >> r >> alpha_deg)) {
            std::cerr << "Invalid input at point " << (i + 1) << std::endl;
            return false;
        }
        points.emplace_back(r, alpha_deg);
    }

    return true;
}

bool output(const rect &result){

    if (!result.valid()){
        std::cout << "No points or invalid rectangle." << std::endl;
        return false;
    }

    std::cout << std::fixed << std::setprecision(6);
    std::cout << "Bounding rectangle (axis-aligned):" << std::endl;
    std::cout << "  left  = " << result.min_x << std::endl;
    std::cout << "  right = " << result.max_x << std::endl;
    std::cout << "  bottom= " << result.min_y << std::endl;
    std::cout << "  top   = " << result.max_y << std::endl;
    std::cout << "Width  = " << (result.max_x - result.min_x) << "  Height = "
<< (result.max_y - result.min_y) << std::endl;
    std::cout << "Area   = " << ((result.max_x - result.min_x) * (result.max_y -
result.min_y)) << std::endl;

    return true;
}

int main() {

    std::vector<std::pair<double,double>> points;

    if (!input(points)){
        return 1;
    }
```

```
    rect result = process_compute_rect(points);

    if (!output(result)){
        return 0;
    }

    return 0;
}
```