МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ

ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего образования

**«Нижегородский государственный университет им. Н.И. Лобачевского»**

**Национальный исследовательский университет**

**Институт информационных технологий, математики и механики**

**Кафедра математического обеспечения и суперкомпьютерных технологий**

**ОТЧЕТ ПО УЧЕБНОЙ ПРАКТИКЕ**

**«Реализация плексов»**

**Выполнил:** студент группы 381706-1

Соболева Ю.А.

\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Научный руководитель:**

аспирант каф. МОСТ ИИТММ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Лебедев И.Г.

Нижний Новгород

2019.

Содержание

[1. Введение 3](#_Toc533027637)

[2. Постановка задачи 4](#_Toc533027638)

[3. Руководство пользователя 5](#_Toc533027639)

[4. Руководство программиста 6](#_Toc533027640)

4.[1. Описание структуры программы 6](#_Toc533027641)

4.[2. Описание структур данных 6](#_Toc533027642)

4.[3. Описание алгоритмов 7](#_Toc533027643)

[5. Заключение 8](#_Toc533027644)

[6. Литература 9](#_Toc533027645)

# **Введение**

**Структура данных** - программная единица, которая определяет метод хранения и обработки различных логически связанных данных в вычислительной технике. Знание структур данных позволяет наиболее компактно и практично расположить данные в памяти компьютера. В данной работе мы рассмотрим такую структуру данных, как **плекс**.

Под **плексом** в рамках этой работы называют графическое изображение моделируемого объекта, характеризующее его форму и размеры и выполненное по определенным правилам проектирования с применением общепринятых соглашений по изображению и обозначению графических элементам.

В качестве плекса будем рассматривать плоский геометрический объект, состоящий из отрезков прямых (линий) и граничных точек этих линий (представление других геометрических элементов может рассматриваться как тема дополнительных заданий). Набор линий, образующих плекс, должен быть связным, т.е. любая линия плекса должна иметь общую точку хотя бы с одной другой линией плекса**.**

**Целью данной лабораторной работы** является разработка структуры данных для хранения плексов с использованием точек и отрезков, а также освоение таких инструментов разработки программного обеспечения, как система контроля версий Git и фрэймворк для разработки автоматических тестов Google Test.

# **Постановка задачи**

Выполнение работы предполагает решение следующих задач:

1. Реализация класса точки TPoint.
2. Реализация класса отрезка TSection.
3. Реализация класса плекса TPlex.
4. Разработка интерфейса для данных классов.
5. Обеспечение работоспособности примера использования.
6. Реализация нескольких тестов на базе Google Test.

# **Руководство пользователя**

Пример использования **точек** (рис. 1):

* Сначала программа запрашивает координаты точки А у пользователя. Затем выводит на экран.
* Затем программа выводит на экран координаты точек В и С (они заданы в программе).

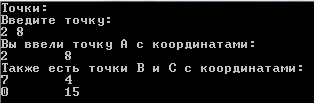


Рис. 1

Пример использования **отрезков** (рис. 2):

* Программа выводит на экран отрезок АВ.



Рис. 2

Пример использования **плексов** (рис. 3):

* Программа информирует пользователя о создании плекса из точки А и В.
* К плексу добавляются точки с координатами (0, 15) и (3, 17).
* Затем программа предлагает пользователю ввести собственную точку. Она добавляется к плексу.
* Получившийся плекс выводится на экран.

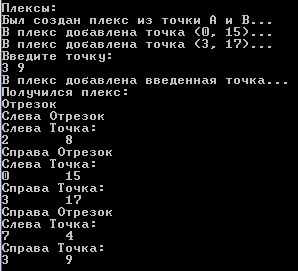


Рис. 3

# **Руководство программиста**

## **Описание структуры программы**

**Проект “Plex ” состоит из следующих файлов:**

1. Plex.cpp (в нём находится main)

**Проект “ PlexLib ” состоит из следующих файлов:**

1. PlexLib.h (описание класса “TPlex” реализация методов класса “TPlex” и перегрузка операций )
2. PlexLib.cpp
3. PointLib.h (описание класса “TPoint” реализация методов класса “TPoint” и перегрузка операций )
4. SectionLib.h (описание класса “TSection” реализация методов класса “TSection” и перегрузка операций )

**Проект “ PlexTest” состоит из следующих файлов:**

1. test\_plexlib.cpp (реализация тестов для класса TPlex)
2. test\_main.cpp

## **Описание структур данных**

***Класс TPoint:***

*Поля:*

* double x, y; - координаты точки;

*Конструкторы:*

* TPoint (double \_x = 0, double \_y = 0); - конструктор инициализации;
* TPoint (TPoint &A); - конструктор копирования;

*Методы:*

* void SetX(double \_x); - установить ключ;
* void SetY(double \_y); - установить ключ;
* double GetX(); - получить ключ;
* double GetY(); - получить ключ;
* virtual void show(); - вывести точку на экран;

*Перегруженные операторы:*

* bool operator == (TPoint &A); - оператор присваивания;
* TPoint& operator = (TPoint &A); - оператор равно;

***Класс TSection:***

*Поля:*

* TPoint A, B; - точки на концах отрезка;

*Конструкторы и деструктор:*

* TSection (TPoint \_A = 0, TPoint \_B = 0); - конструктор инициализации;
* TSection (TSection &S); - конструктор копирования;

*Методы:*

* virtual void show(); - вывести отрезок на экран;

*Перегруженные операторы:*

Нет перегруженных операторов.

***Класс TPlex:***

*Поля:*

* TPoint \*l, \*r; - указатели на левую и правую точку плекса;

*Конструкторы и деструктор:*

* TPlex(); - конструктор по умочанию;
* TPlex (TPoint \*\_l, TPoint \*\_r); - конструктор инициализации;
* TPlex (TPlex& P); - конструктор копирования;

*Методы:*

* void SetL (TPoint \*\_l); - установить левую точку;
* void SetR (TPoint \*\_r); - установить правую точку;
* TPoint\* GetL(); - получить левую точку;
* TPoint\* GetR(); - получить правую точку;
* virtual void show(); - вывести плекс на экран;
* void Add (TPoint \*\_A, TPoint \*\_B); - добавить точку к плексу;
* TPoint\* Search(TPoint \*A); - найти точку в плексе;

*Перегруженные операторы:*

Нет перегруженных операторов.

## **Описание алгоритмов**

Метод **Search**:

Если левая точка равна искомой, возвращаем указатель на неё. Аналогично поступаем с правой точкой. Иначе рекурсивно выполняем метод Search для левого и правого плекса. В случае если точка не будет найдена, бросаем исключение.

Метод **Add**:

Если один из параметров функции равен левой точке – левая точка становится плексом (новая точка добавляется справа). В случае с точкой справа, новая точка добавляется слева. Иначе рекурсивно выполняем метод Add для левого и правого плексов. В случае если точка не будет найдена, бросаем исключение.

Метод **Show:**

Начинаем вывод плекса с отрезка. Если слева находится плекс, то рекурсивно выполняем метод Show для него. Если слева находится точка, то выводим её координаты на экран и переходим на правую ветку плекса.

# **Заключение**

В ходе выполнения лабораторной работы я смогла реализовать такую структуру данных, как стэк. Вместе с ней, согласно заданному интерфейсу, я реализовала класс TPoint, TSection и TPlex. Написание нескольких своих тестов, помогло мне разобраться с системой автоматических тестов Google Test.

В результате проделанной работы у меня получилось

1. Реализовать класс точки TPoint.

2. Реализовать класс отрезка TSection.

3. Реализовать класс плекса TPlex.

4. Разработать интерфейс для данных классов.

5. Обеспечить работоспособность примера использования.

6. Реализация нескольких тестов на базе Google Test.

Таким образом, данная лабораторная работа отвечает всем поставленным задачам. Это означает, что цель работы была достигнута.

# **Литература**

1. Лабораторный практикум. Составители: Барышева И.В., Мееров И.Б., Сысоев А.В., Шестакова Н.В. Под редакцией Гергеля В.П. Учебно-методическое пособие. – Нижний Новгород: Нижегородский госуниверситет, 2017. – 105с.

URL: <http://www.unn.ru/books/met_files/Pract_ADS.pdf>

1. Национальный открытый университет «Интуит». Курс «Основы программирования». Лекция 11: Структуры данных: общее понятие, реализация. Простейшие структуры данных: очередь, стек. Использование стека и обратная польская запись.

URL: <https://www.intuit.ru/studies/courses/2193/67/lecture/1980?page=3>