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**

**«Структура хранения геометрических объектов»**

**Выполнил:** студент группы 381706-1

Нечаева Екатерина Владимировна

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Научный руководитель:**

ассистент каф. МОСТ ИИТММ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Лебедев И.Г

Нижний Новгород

2019.

Содержание

[1. Введение 3](#_Toc560926)

[2. Постановка задачи 4](#_Toc560927)

[3. Руководство пользователя 5](#_Toc560928)

[4. Руководство программиста 6](#_Toc560929)

[4.1 Описание структуры программы 6](#_Toc560930)

[4.2 Описание структур данных 6](#_Toc560931)

[4.3 Описание алгоритмов 7](#_Toc560932)

[5. Заключение 10](#_Toc560933)

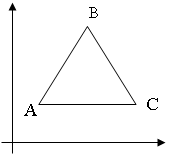
[6. Литература 11](#_Toc560934)

# Введение

Тема лабораторной работы относится к важной области задач обработки графических данных, потребность в анализе которых встречается при рассмотрении многих научнотехнических и производственных задач. Данная лабораторная работа включает в себя представление графической информации – чертежа.

Под чертежом 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 чертежа 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 чертеж, должен быть связным, т.е. любая линия чертежа должна иметь общую точку хотя бы с одной другой линией чертежа.



*Рисунок 1. Пример чертежа.*

Для представления данных, описывающих структуру чертежа, используется структура хранения типа плекс. Основу плексов для представления чертежей, состоящих из линий и точек, составляют вершины (узлы), в каждой из которых располагается информация о той или иной линии чертежа.

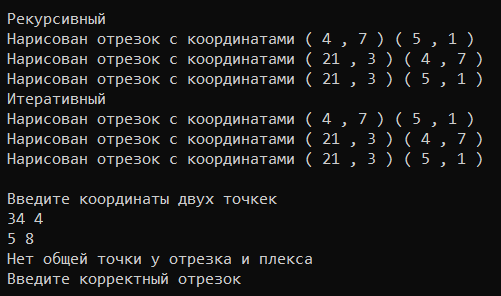
# Постановка задачи

В рамках лабораторной работы ставится задача разработки программного комплекса для представления конкретного типа графических данных – чертежей, образуемых из ограниченного набора различных геометрических элементов (точек, линий, окружностей и т.п.). Разрабатываемый комплекс должен обеспечивать:

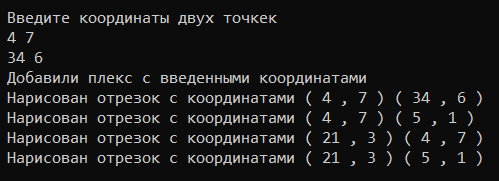
* Отображение чертежа на экран;
* редактирование чертежей (вставка линий);

# Руководство пользователя

Программа начинает свою работы с вывода на экран произвольного чертежа с помощью двух алгоритмов обхода плекса: итеративный и рекурсивный. После чего пользователю предлагается ввести координаты двух точек. Если пользователь ввел некорректный отрезок, то есть ни одна из точек не является общей для плекса, программа выведет на экран сообщение об ошибке и попросит ввести координаты заново.

*  
Рисунок 2. Пример вывода на экран чертежа.*

Когда пользователь введет правильные координаты, программа добавит к данному плексу этот отрезок и выведет на экран полученный плекс. После чего работа завершится.

  
*Рисунок 3. Добавление отрезка к плексу.*

# Руководство программиста

## 4.1 Описание структуры программы

Программа состоит из следующих модулей:

* Модуль plex. Содержит пример использования класса плекс (mainplex.cpp)
* Модуль libplex – статическая библиотека. Содержит заголовочные файлы point.h и plex.h, в первом описан класс точка, во втором класс плекс. Исполнительные файлы point.cpp и plex.cpp, в которых реализованы методы описанных выше классов.
* Модуль plexTest. Содержит тесты, проверяющие правильность работы методов (plexTest.cpp, pointTest.cpp)
* Модуль libthrows – библиотека исключений. Содержит заголовочный файл throws.h, в котором он реализован.

## Описание структур данных

#### Класс TException – класс исключений.

Содержит одно защищенное поле *string str,* переменная строкового типа, которая хранит сообщение об исключении.

Содержит публичные методы

* TException(std::string \_str); //конструктор-инициализатор
* void Print(); //вывод данного исключения на консоль

#### Класс TPoint.

Защищенные поля:

* double x, y; //координаты точки

Содержит публичные методы:

* TPoint(double \_x=0, double \_y=0); //конструктор инициализатор
* double GetX(); //получить первую координату точки
* double GetY(); //получить вторую координату точки
* void SetX(double \_x); // установить первую координату точки
* void SetY(double \_y); // установить вторую координату точки
* virtual void Show(); //отображение точки на экран

#### Класс TPlex.

Защищенные поля:

* TPoint \*first, \*last; //узлы плекса

Содержит публичные методы:

* TPlex(); //конструктор по умолчанию
* TPlex(TPoint \*\_first, TPoint \*\_last); //конструктор инициализатор
* TPlex(TPlex &A); //конструктор копирования
* void Add(TPoint \*\_first, TPoint \*\_last); //добавление линии к плексу
* TPoint\* GetFirst(); //получить левый узел плекса
* TPoint\* GetLast(); //получить правый узел плекса
* void SetFirst(TPoint \*\_first); //установить левый узел плекса
* void SetLast(TPoint \*\_last); //установить правый узел плекса
* void Show(); //рекурсивный метод отображения плекса
* TPoint\* Show(TPlex \*p); //итеративный метод отображения плекса

## Описание алгоритмов

## Рекурсивный метод отображения плекса.

На вход поступает указатель на объект класса TPex.

# Если у плекса его левая и правая ветви равны нулю, бросается исключении о том, что плекс отсутствует.

# Иначе, производятся проверки с ветками плекса:

## Если левая ветка является точкой и правая ветка – точка, выводим на экран полученный отрезок.

## Иначе, если одна из веток указывает на плекс, вызываем для нее рекурсивный метод, передав эту ветку.

# Возвращается правая точка.

**Итеративный метод отображения плекса.**

1. Ветви, определяемые при обходе плекса, помещаются в стек.
2. Пока стек не пуст
   1. У ветви, извлекаемой из стека, последовательно определяются левая и правая точки:
      1. Определение левой точки: Если получаемый указатель указывает на плекс, обработка текущей ветви откладывается (помещается в стек) и начинается анализ новой ветви, данная процедура повторяется итеративно до получения ветви с известной левой точкой
      2. Определение правой точки: Если получаемый указатель указывает на плекс, обработка текущей ветви откладывается (помещается в стек); после это формируется описание новой, которая также помещается в стек, и алгоритм обхода повторяется с 2.1.1.
   2. При получении ветви с определенными граничными точками выводится на экран.

2.3. Из стека извлекается новая линия, для которой правая точка обработанной ветви используется в качестве первой неизвестной граничной точки.

**Добавление отрезка к плексу.**

На вход подаются два узла: левый указатель на точку и правый.

# Если мы хотим добавить отрезок в самом начале создания плекса, то его левый узел приравнивается к левому из входных данных, а правая – к правому.

# Иначе:

## Создаем два стека: указатель на класс плекс и указатель на класс точка. Кладем в первый стек существующий плекс дважды (это делается для того, чтобы извлечение из обоих стеков выполнялось параллельно), а во второй стек входные данные, то есть два указателя на точку. И флаг.

## В цикле, пока флаг равен истине и один из стеков не пуст, достаем из второго стека указатель на точку и из первого указатель на плекс.

### Если данная точка совпадает с одной из входных данных, добавляем ее к плексу по правилу: если точка совпадает с левой точкой из входных данных, то вместо данной точки добавляется новый плекс, полученный из входных данных, поменяв в новом плексе левую и правую точки местами. Если данная точка совпадает с правой точкой из входных данных, производится аналогичное действие, только точки из входных данных местами не меняются. Флаг переводим в состояние ложь

### Если данная точка указывает на плекс, кладем ее в первый стек, а ее ветки кладем во второй стек. И переходим снова к действию 2.2.1

## Если цикл прошел, а флаг остался равен истине, бросается исключение о том, что входные данные некорректны.

# Заключение

В ходе выполнения данной лабораторной была реализована структура хранения геометрических объектов с помощью плексов. Были реализованы два алгоритма обхода плекса: простой в реализации и понимании, но не производительный рекурсивный алгоритм, и труднопонимаемый, но быстродейственный итеративный алгоритм и добавление к существующему плексу отрезок.

Были разработаны тесты, проверяющие корректную работу методов данных классов, с помощью Google C++ Testing Framework.

# Литература

1. Учебно-методическое пособие [Электронный ресурс]

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

1. Геометрические объекты [Электронный ресурс]

[https://mylektsii.ru/8-121030.html](http://www.intuit.ru/studies/courses/683/539/lecture/12151?page=3)

1. Структуры хранения геометрических объектов [Электронный ресурс]

[https://docviewer.yandex.ru/view/339975272/?\*=huRAbG%%3D%3D&lang=ru](https://docviewer.yandex.ru/view/339975272/?*=huRAbG%%3D%3D&lang=ru)