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: Плекс»**

**Выполнил:** студент группы 381706-1

Денисов Владислав Львович

\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Научный руководитель:**

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

Нижний Новгород

2018.

Содержание

[1. Введение 3](#_Toc8329309)

[2. Постановка задачи 4](#_Toc8329310)

[3. Руководство пользователя 5](#_Toc8329311)

[4. Руководство программиста 6](#_Toc8329312)

[4.1 Описание структуры программы 6](#_Toc8329313)

[4.2 Описание структур данных 6](#_Toc8329314)

[4.3 Описание алгоритмов 9](#_Toc8329315)

[5. Заключение 11](#_Toc8329316)

[6. Литература 12](#_Toc8329317)

# Введение

Лабораторная работа направлена на разработку системы хранения геометрических объектов – плексов.

Плекс будет представлять из себя некий чертеж. И в рамках этой лабораторной работы условимся чертежом называть графическое изображение моделируемого геометрического объекта. Однако рассматривать будем только плоские геометрические объекты, состоящие из отрезков прямых (линий) и граничных точек этих линий. Набор линий, образующих чертеж, должен быть связным, т.е. любая линия чертежа должна иметь общую точку хотя бы с одной другой линией чертежа.

# Постановка задачи

В рамках лабораторной работы ставится задача эффективной реализации системы для работы с плексами, описание которых было дано выше.

Для работы будут разработаны:

* Методы, позволяющие создать точку, отрезок, плекс, круг.
* Метод добавления отрезка к плексу.
* Методы вывода на консоль каждого из перечисленных объектов.

Программное решение будет выглядеть следующим образом:

1. Класс точки – TPoint.
2. Класс отрезка – TLine,
3. Класс плекса – TPlex.
4. Класс круга – TCicle.
5. Класс для обработки исключений – TException, которые могут возникнуть при выполнении различных операций.
6. Программа, демонстрирующая работу класса TPlex.
7. Набор автоматических тестов с использованием Google C++ Testing Framework.

# Руководство пользователя

В качестве примера использования разработанной библиотеки предлагается следующее.

После запуска тестовой программы пользователь вводит координаты двух точек, из которых строится плекс. Производится итеративный вывод плекса на консоль.

Затем пользователь вводит координаты еще одной точки. Производится добавление отрезка, построенного с помощью уже существующей второй точки и только что введенной точки. Полученный плекс аналогично выводится на консоль.

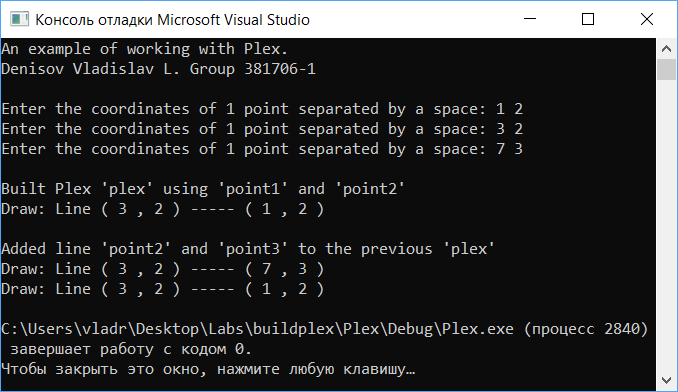


Рисунок 1 Пример работы демонстрационной программы.

# Руководство программиста

## 4.1 Описание структуры программы

Программа состоит из следующих модулей:

* Модуль Plex. Содержит пример использования плексов. Реализация в файле *main\_Plex.cpp.*
* Модуль PlexLib – статическая библиотека. Содержит файлы *Plex.h*, *Point.h*, *Line.h, Circle.h,* в которых описан интерфейс классов *TPlex*, *TPoint, TLine, TCircle* соответственно. А также файлы *Plex.cpp*, *Point.cpp, Line.cpp*, *Circle.cpp,* содержащие реализацию указанных классов.
* Модуль PlexTest. Содержит суммарно 37 тестов (9 для класса плексов, 5 для класса точки, 10 для класса линии, 13 для класса круга), описанных в файлах *PlexTest.cpp*, *PointTest.cpp, LineTest.cpp*, *CircleTest.cpp* и разработанных с помощью использования Google C++ Testing Framework.
* Модуль ExceptionLib – библиотека, позволяющая создавать собственные исключения.

## Описание структур данных

#### Класс TException – класс исключений.

Класс содержит 1 **private** поле *std::string msg* – переменная, хранящая сообщение об ошибке в виде строки.

И содержит 2 **public** элемента:

*TException(std::string \_str) : msg(\_msg)* – конструктор с одним параметром.

*void Print()* – метод отображения ошибки на консоль.

#### Класс TPoint – класс точки.

Рассмотрим класс *TPoint* подробно.

**Элементы класса, объявленные со спецификатором protected:**

double x, y – координаты точки.

**Элементы класса, объявленные со спецификатором public:**

TPoint(double \_x=0, double \_y=0) – конструктор инициализатор.

double GetX() – получение координаты x.

double GetY() – получение координаты y.

void SetX(double \_x) – изменение координаты x.

void SetY(double \_y) – изменение координаты y.

virtual void Show() – метод вывода точки на консоль.

#### Класс TLine – класс отрезка.

Рассмотрим класс *TLine* подробно.

**Элементы класса, объявленные со спецификатором protected:**

TPoint\* point1 – указатель на левый конец отрезка.

TPoint\* point2 – указатель на правый конец отрезка.

bool del\_flag\_1 – флаг, разрешающий удаление левой точки.

bool del\_flag\_2 – флаг, разрешающий удаление правой точки.

**Элементы класса, объявленные со спецификатором public:**

TLine() – конструктор по умолчанию.

TLine(const TPoint\* \_point1, const TPoint\* \_point2) – конструктор инициализатор от двух точек.

TLine(const double x1, const double y1, const double x2, const double y2) – конструктор инициализатор от координат двух точек.

TLine(const TLine& line) – конструктор копирования.

virtual ~TLine() – деструктор.

void SetX1(const double \_x1) – метод изменения координаты x левого конца отрезка.

void SetY1(const double \_y1) – метод изменения координаты y левого конца отрезка.

void SetX2(const double \_x2) – метод изменения координаты x правого конца отрезка.

void SetY2(const double \_y2) – метод изменения координаты y правого конца отрезка.

void SetPoint1(TPoint\* \_point1) – метод изменения левого конца отрезка.

void SetPoint2(TPoint\* \_point2) – метод изменения правого конца отрезка.

double GetX1() – метод получения координаты x левого конца отрезка.

double GetY1() – метод получения координаты y левого конца отрезка.

double GetX2() – метод получения координаты x правого конца отрезка.

double GetY2() – метод получения координаты y правого конца отрезка.

TPoint\* GetPoint1() – метод получения указателя на левый конец отрезка.

TPoint\* GetPoint2() – метод получения указателя на правый конец отрезка.

void Show() – метод вывода отрезка на консоль.

#### Класс TCircle – класс круга.

Рассмотрим класс *TCircle* подробно.

**Элементы класса, объявленные со спецификатором protected:**

TPoint\* center – указатель на точку, являющуюся центр круга.

double radius – значение радиуса круга.

**Элементы класса, объявленные со спецификатором public:**

TCircle() – конструктор по умолчанию.

TCircle(const TPoint\* point, double rad) – конструктор инициализатор от точки и радиуса.

TCircle(const double \_x, const double \_y, const double rad) – конструктор инициализатор от координат точки и радуса.

TCircle(const TCircle& circle) – конструктор копирования.

virtual ~TCircle() – деструктор.

void SetX(const double \_x) – метод изменения координаты x для центра круга.

void SetY(const double \_y) – метод изменения координаты y для центра круга.

void SetRadius(const double rad); //изменить радиус

double GetX(); //получить координату X для центра

double GetY(); //получить координату Y для центра

double GetRadius(); //получить радиус

void Show(); //вывод круга на консоль

#### 4.2.5 Класс TPlex – класс плекса.

Рассмтрим класс *TPlex* подробно.

**Элементы класса, объявленные со спецификатором protected:**

TPoint \*right, \*left – указатели на левую и правую сторону плекса.

**Элементы класса, объявленные со спецификатором public:**

TPlex() – конструктор по умолчанию.

TPlex(TPoint \*\_right, TPoint \*\_left) – конструктор инициализатор от двух точек.

TPlex(TPlex& plex) – конструктор копирования.

TPoint\* GetRight() – метод получения правую стороны плекса.

TPoint\* GetLeft() – метод получения левой стороны плекса.

void Add(TPoint \*\_right, TPoint \*\_left) – метод добавления отрезка к плексу.

void SetRight(TPoint \*\_right) – метод изменения указателя на правую сторону плекса.

void SetLeft(TPoint \*\_left) – метод изменения указателя на левую сторону плекса.

void Show() – метод итеративного вывода плекса на консоль.

TPoint\* Show(TPlex \*plex) – метод рекурсивного вывода плекса на консоль.

## Описание алгоритмов

Рассмотрим некоторые алгоритмы, работа которых не очевидна на первый взгляд.

**Рекурсивный метод вывода плекса на консоль.**

Получаем на вход указатель на объект класса TPlex.

1. Если у плекса его левая и правая ветви равны нулю, то возбуждается исключение в связи с тем, что плекс не существует, а именно был создан, но не заполнен данными.
2. Если ветви не равны нулю, то выполняем проверку для каждой из них:

2.1. В случае, если левая ветвь является точкой и правая ветка является точкой, то на экран производится вывод полученного отрезка

2.2. В противном случае если одна из веток указывает на плекс, то для вызывается рекурсивный метод вывода плекса на консоль, с параметром в виде указателя на эту ветку-плекс.

1. Производится возврат правой точки.

**Итеративный метод вывода плекса на консоль.**

1. Ветви, которые определяются при обходе плекса, помещаем в стек.
2. В цикле до тех пор, пока стек не пуст выполняем следующие действия
   1. Извлекаем из стека ветвь и последовательно определяем ее левую и правую точки:

2.1.1 В случае обработки левой точки делаем следующее. Если получаемый указатель соответствует плексу, то обработку текущей ветви откладываем (помещаем ее в стек) и начинаем рассмотрение следующей ветви. Повторяем данные действия до тех пор, пока не будет найдена ветвь с известной левой точкой.

2.1.2 В случае обработки правой точки делаем следующее. Если получаемый указатель соответствует плексу, то обработку текущей ветви откладываем (помещаем ее в стек). Затем переходим к новой ветви, которую также помещаем в стек. Алгоритм обхода повторяем, начиная снова с пункта 2.1.1.

2.2 При получении ветви с конечными граничными точками производится вывод очередного отрезка на экран.

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

1. Плекс успешно выведен на консоль.

**Добавление отрезка к плексу.**

Метод получает на вход два аргумента: указатели на левый и правый концы отрезка.

Выполняем следующие действия:

1. Если выполняется добавление отрезка в самом начале создания плекса, то указатель на его левую ветвь приравнивается к левому концу отрезка, а указатель на правую ветвь – к правому.
2. В противном случае:
   1. Выполняем создание двух стеков, которые будут хранить указатели на объекты класса TPlex и указатели на объекты класса TPoint. В первый стек дважды добавляем существующий плекс. Такое действие объясняется тем, что необходимо извлечение из обоих стеков выполнялось параллельно. Во второй стек помещаем входные данные, а именно два указателя на точки (на концы отрезка).
   2. Устанавливаем в истинное значение флага, который будет показывать, что необходимо продолжать поиск подходящей позиции.
   3. В цикле до тех пор, пока флаг равен истине и один из стеков не пуст, извлекаем из второго стека указатель на точку и из первого указатель на плекс.
      1. Если извлеченная из стека точка совпадает с одной из тех, что были переданы в качестве входных данных, то добавляем ее к плексу по следующему правилу.

Если точка совпадает с левой точкой из входных данных, то вместо данной точки добавляется новый плекс, который можно создать с помощью входных точек путем перемены в новом плексе левой и правой точек местами.

Если точка совпадает с правой точкой из входных данных, то производится аналогичное действие, однако входные точки в ходе построения плекса местами не меняются.

Значение флага переводится в состояние ложь, т.к. отрезок успешно добавлен к плексу.

* + 1. Если извлеченная из стека точка указывает на плекс, помещаем ее в первый стек, а ее ветви помещаем во второй стек. Переходим к выполнению действий, описанных в пункте 2.2.1

1. Если цикл завершился, но значение флаг осталось равно истине, то возбуждается исключение по причине того, что входные данные оказались некорректными. Хотя бы одна из точек, должен содержаться в существующем плексе.

# Заключение

В результате лабораторной работы была разработана библиотека, реализующая необходимые классы для работы с плексами. Она позволяет создать объект класса TPlex и выполнить с ним некоторые действия, задача реализации которых была поставлена в начале данной лабораторной работы.

Были разработаны и доведены до успешного выполнения тесты, разработанные для данного программного проекта с использованием Google C++ Testing Framework.

Программное решение было продемонстрировано с помощью простейшего набора действий над плексами. Описание примера работы с плексами было представлено в разделе «Руководство пользователя».

# 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.