Федеральное агентство по образованию Российской Федерации

Государственное образовательное учреждение

высшего профессионального образования

Нижегородский государственный университет им. Н.И. Лобачевского

Институт информационных технологий математики и механики

Направление подготовки: «Фундаментальная информатика и информационные технологии»

Отчёт по лабораторной работе

**Geometric**

Выполнил:

студент ИИТММ гр. 381906-2

Назаров Н.С.

Проверил:

Нижний Новгород

2020 г.

Содержание

I. [Введение 3](file:///D:\загрузки\Laba4.docx#_Toc24457830)

II. [Постановка задачи](file:///D:\загрузки\Laba4.docx#_Toc24457831) 4

III. [Руководство пользователя](file:///D:\загрузки\Laba4.docx#_Toc24457832) 5

IV. [Руководство программиста](file:///D:\загрузки\Laba4.docx#_Toc24457833) 6

1. [Описание структуры программы](file:///D:\загрузки\Laba4.docx#_Toc24457834) 6

2. [Описание структур данных](file:///D:\загрузки\Laba4.docx#_Toc24457835) 7

3. [Описание алгоритмов](file:///D:\загрузки\Laba4.docx#_Toc24457836) 7

V. [Эксперименты](file:///D:\загрузки\Laba4.docx#_Toc24457837) 10

VI. [Заключение](file:///D:\загрузки\Laba4.docx#_Toc24457838) 11

# 1. Введение

# Вычислительная геометрия – это раздел информатики, изучающий алгоритмы решения геометрических задач. Такие задачи возникают в компьютерной графике, проектировании интегральных схем, технических устройств и др. Исходными данными в такого рода задачах могут быть множество точек, набор отрезков, многоугольники и т.п. Результатом может быть либо ответ на какой-то вопрос, либо какой-то геометрический объект.

# 2. Постановка задачи

Написать структуру данных для работы с геометрическими объектами в N мерном пространстве. Тип данных определяется пользователем (шаблоны). Должны быть реализованы простейшие геометрические объекты, такие как: точка, линия, круг, квадрат, многоугольник, куб. Кроме самих геом. объектов должен быть реализован класс осуществляющий обобщение действий со всеми созданными пользователями объектами - "контейнер". Контейнер дол иметь следующие функции: добавить новый объект, удалить существующий объект, отобразить все имеющиеся объекты, отобразить выбранный объект и т.д., и т.п. Продемонстрировать (написать в main пример) работоспособность.

Должны быть использованы и продемонстрированы: абстракция, инкапсуляция, наследование, полиморфизм, перегрузка операций, шаблоны.

# 3. Руководство пользователя

Шаги, совершаемые пользователем:

1. Создать объекты типа Point, Line, Triangle, Circle, Square, Cube Polygon.
2. Работать с объектами, используя поля и методы классов, а также доступ с закрытым полям классов.
3. Получить результат.

# 4. Руководство программиста

## 4.1. Описание структуры программы

* В модуле main2.cpp определена стандартная функция int main(), внутри которой содержится набор действий с геометрическими фигурами.
* В модуле Nothing определен класс базовый класс, а также объявлены все виртуальные методы.
* В модуле Point.h определен одноименный класс - наследник Nothing.
* В модуль Point2.h вынесены все определения методов класса Point.
* В модуле Line.h определен одноименный класс - наследник Point.
* В модуль Line2.h вынесены все определения методов класса Line.
* В модуле Circle.h определен одноименный класс- наследник Point.
* В модуль Circle2.h вынесены все определения методов класса Circle.
* В модуле Triangle.h определен одноименный класс- наследник Line.
* В модуль Triangle2.h вынесены все определения методов класса Triangle.
* В модуле Square.h определен одноименный класс- наследник Point.
* В модуль Square2.h вынесены все определения методов класса Square.
* В модуле Cube.h определен одноименный класс- наследник Square.
* В модуль Cube2.h вынесены все определения методов класса Square.
* В модуле Polygon.h определен одноименный класс- наследник Line.
* В модуль Polygon2.h вынесены все определения методов класса Polygon.

## Описание структур данных

В программе определен один абстрактный класс:

Nothing

Внутри класса Nothing определены следующие виртуальные функции:

* Nothingl(){} – деструктор
* virtual double S () = 0 - виртуальный метод вычисления площади
* virtual double V() = 0 – виртуальный метод вычисления объёма
* virtual ostream& Cout(ostream& os) = 0 - виртуальный метод вывода Cout
* friend ostream& operator<<(ostream& out, Nothing& a) – виртуальный метод вывода

В программе определены восемь следующих шаблонных классов:

Class Point

Class Line

Class Circle

Class Triangle

Class Square

Class Cube

Class Polygon

Class Container

Внутри класса Point определены следующие поля:

* First\* m– шаблонный указатель.
* int tmp300– целочисленное количество координат точки

Внутри класса Point определен следующий набор public-методов

(плюс конструкторы и деструктор):

* Point() – конструктор по умолчанию, не принимает никаких параметров, инициализирует все два поля 0, через списки инициализации.
* Point(T&, int) – конструктор инциализатор, принимает на вход два параметра типа T& и типа int создает динамический массив заданных размеров, заполняет его соответственным элементом переданным в конструктор, поле tmp300 инициализируется значением переданным в конструктор соответственно.
* Point(Point&) – конструктор копирования, принимает на вход объект типа Point, создает объект с теми же характеристиками.
* Point(T\* mb, int ab) – конструктор, принимает на вход массив и его размер, инициализирует свои поля соответственно.
* ~Point() – деструктор, очищает выделенную методами и конструкторами память.
* bool operator !=(const Point& aa) – перегрузка оператора !=.
* ostream& Cout(ostream& os) - функция вывода.
* Point& operator =(const Point& aa) – перегрузка оператора присвоения для объекта типа Point.
* Point& operator =(const int a) – перегрузка оператора присвоения для обьектов типа int.
* T\* Getm() – метод возвращающий массив с координатами точками.
* int Gettmp300() – метод возвращающий размерность точки.
* double S () – функция подсчета площади
* double V () - функция подсчета обьема.
* void SetPoint(T\* md, int ab) – метод изменения координат и размерности точки, «Сеттер».

Внутри класса Line, наследника класса Point определено следующее поле:

* Point mq

Внутри класса Line определен следующий набор public-методов

(плюс конструкторы и деструктор):

* Line() – конструктор по умолчанию, не принимает никаких параметров, вызывает конструктор по умолчанию от класса Point, инициализирует вторую точку пустой точкой.
* Line(Point&, Point&) – конструктор инциализатор, принимает на вход два параметра типа Point&, вызывает конструктор копирования Point от первого параметра и инициализирует поле mq вторым параметром.
* Line(Line&) – конструктор копирования, принимает на вход объект типа Line, создает объект с теми же характеристиками.
* ~Line () – деструктор.
* Point<H> GetPointA() – метод возвращающий начало отрезка.
* Point<H> GetPointB() - метод возвращающий конец отрезка.
* Double Length() – метод возвращающий длину отрезка.
* Line& operator =(Line& c) – перегрузка оператора присвоения.
* ostream& Cout(ostream& os) –функция вывода
* double S () – функция подсчета площади
* double V () - функция подсчета обьема.
* void SetLine (Point&, Point&) – метод изменения начала и конца точки, «Сеттер».

Внутри класса Circle, наследника класса Point определено следующее поле:

Line<T> line

Внутри класса Circle определен следующий набор public-методов

(плюс конструкторы и деструктор):

* Circle() – конструктор по умолчанию, не принимает никаких параметров, вызывает конструктор по умолчанию от класса Point и инициализирует поле радиуса пустым отрезком.
* Circle(Point&, Line<T>&) – конструктор инциализатор, принимает на вход два параметра типа Point& и типа Line<T>& вызывает конструктор копирования Point от первого параметра и инициализирует поле line вторым параметром.
* Circle(Circle&) – конструктор копирования, принимает на вход объект типа Circle, создает объект с теми же характеристиками.
* ~Circle () – деструктор.
* Line<T> GetLine () – метод возвращающий радиус окружности.
* Point<T> GetCentr () - метод возвращающий центр окружности.
* Circle& operator =(Circle & a) – перегрузка оператора присвоения.
* ostream& Cout(ostream& os) – функция вывода
* double S () – функция подсчета площади
* double V () - функция подсчета обьема.

Внутри класса Square, наследника класса Point определено следующее поле:

* Point<Q> DD

Внутри класса Square определен следующий набор public-методов

(плюс конструкторы и деструктор):

* Square() – конструктор по умолчанию, не принимает никаких параметров, вызывает конструктор по умолчанию от класса Point инициализирует поле DD нулем.
* Square(Point&, Point&) – конструктор инциализатор, принимает на вход два параметра типа Point&, вызывает конструктор копирования Point от первого параметра и инициализирует поле DD вторым параметром.
* Square(Square&) – конструктор копирования, принимает на вход объект типа Square, создает объект с теми же характеристиками.
* ~Square() – деструктор.
* Line<Q> GetDiag () – метод возвращающий начало отрезка.
* Square& operator =(Square& a) – перегрузка оператора присвоения.
* ostream& Cout(ostream& os) – функция вывода
* double S () – функция подсчета площади
* double V () - функция подсчета обьема.
* void SetSquare (Square&) – метод изменения квадрата, «Сеттер».

Внутри класса Triangle, наследника класса Line определено следующее поле:

* Point CC

Внутри класса Triangle определен следующий набор public-методов

(плюс конструкторы и деструктор):

* Triangle() – конструктор по умолчанию, не принимает никаких параметров, вызывает конструктор по умолчанию от класса Line, инициализирует поле Top нулем.
* Triangle(Point<R>&, Point<R>&, Point<R>&) – конструктор инциализатор, принимает на вход три параметра типа Point&, вызывает конструктор Line от первых двух параметров и инициализирует поле CC третьим параметром.
* Triangle(Triangle&) – конструктор копирования, принимает на вход объект типа Triangle, создает объект с теми же характеристиками.
* ~Triangle () – деструктор.
* Point<R> GetPointA() – метод возвращающий точку основания треугольника.
* Point<R> GetPointB() - метод возвращающий вторую точку основания.
* Point<R> GetPointTop() – метод возвращающий вершину треугольника.
* Triangle& operator =(Trinagle& p) – перегрузка оператора присвоения.
* ostream& Cout(ostream& os) - функция вывода
* double S () – функция подсчета площади
* double V () - функция подсчета обьема.

Внутри класса Cube, наследника класса Square определено следующее поле:

* Square sq

Внутри класса Cube определен следующий набор public-методов

(плюс конструкторы и деструктор):

* Cube() – конструктор по умолчанию, не принимает никаких параметров, вызывает конструктор по умолчанию от класса Square, поле sq получает значение пустого Square.
* Cube(Point<Y>&, Point<Y>&, Point<Y>&, Point<Y>&, Point<Y>&, Point<Y>&, Point<Y>&, Point<Y>&) – конструктор инциализатор, принимает на вход восемь параметров типа Point&, вызывает конструктор Square от первого и третьего параметра присваивает значению sq Square от последнего и предпоследнего аргумента.
* Cube(Square&, Square&) – конструктор, принимает на вход два квадрата создает куб с нижней и верхней границей из этих кубов.
* Cube(Cube& ) – конструктор копирования, принимает на вход объект типа куба и создает объект с теми же характеристиками.
* ~Cube () – деструктор.
* Square WidthSq – метод возвращающий нижний квадрат.
* Square LengthsSq - метод возвращающий верхний квадрат.
* Cube& operator =(Cube & s) – перегрузка оператора присвоения.
* ostream& Cout(ostream& os) - функция вывода
* double S () – функция подсчета площади
* double V () - функция подсчета обьема.

Внутри класса Polygon, наследника класса Line определены следующие поля:

* Line\* Lin
* Int l

Внутри класса Polygon определен следующий набор public-методов

(плюс конструкторы и деструктор):

* Polygon() – конструктор по умолчанию, не принимает никаких параметров, вызывает конструктор по умолчанию от класса Line.
* Polygon(Point\*, int) – конструктор инциализатор, принимает массив точек и их количество.
* Polygon(Line\*, int) – конструктор, на вход массив сторон и их количество.
* Polygon(Polygon&) – конструктор копирования принимает на ход объект типа Polygon и создает объект с теми же характеристиками.
* ~Polygon () – деструктор.
* Int GetL– метод возвращающий размерность многочлена.
* double\* LengthsLin - метод возвращающий массив длин сторон многоугольника.
* Line\* ArrayLin – метод возвращающий массив сторон многоугольника.
* ostream& Cout(ostream& os) - функция вывода
* double S () – функция подсчета площади
* double V () - функция подсчета обьема.
* Polygon& operator = (Polygon&) – перегрузка оператора присвоить.

Внутри класса Cont, определены следующие поля:

* Nothing\*\* Array
* Int tmp400

Внутри класса Polygon определен следующий набор public-методов

(плюс конструкторы и деструктор):

* Polygon() – конструктор по умолчанию, не принимает никаких параметров.
* Void FOO1(U&) – принимает на вход шаблонный аргумент, ничего не возвращает, записывает этот объект в Container.
* Void FOO2 (G&) – принимает на вход шаблонный аргумент, ищет его в контейнере, и удаляет его.
* ~Container () – деструктор.
* Int GetLength– метод возвращающий количество объектов в контейнер.
* Nothing\* operator [] – перегрузка оператора возврата объекта по индексу.
* Friend ostream& operator<<(ostream& os, Container& a)– перегрузка оператора вывода для класса Container
* Void cout(D&) - принимает на вход шаблонный аргумент и выводит его на экран.

# Эксперименты

В функции Main приведен пример работы с классами геометрических фигур, и содержащим их контейнером.

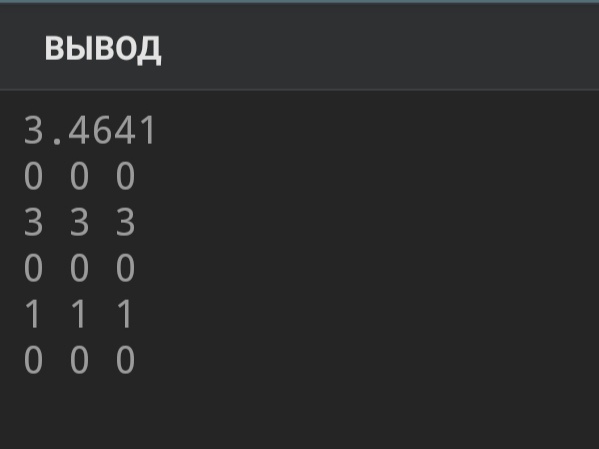


Рисунок 1: Результаты выполнения функции Main.

# 6. Заключение

Значимость моей работы заключается в том, что многие ученики, познакомившись с данным исследованием, на вопрос о необходимости изучать геометрию - ответят положительно! И, конечно же, будут с интересом изучать сложную, но увлекательную науку – геометрию!