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

**Реализация класса GeometricShapes.**

Выполнил:

студент ИИТММ гр. 381906-2

Кулемин П. А.

Проверил:

ассистент каф. математического обеспечения

и суперкомпьютерных технологий

Лебедев И.Г.

Нижний Новгород

2019 г.

Содержание

I. [Введение 3](#_Toc24457830)

II. [Постановка задачи](#_Toc24457831) 4

III. [Руководство пользователя](#_Toc24457832) 5

IV. [Руководство программиста](#_Toc24457833) 6

1. [Описание структуры программы](#_Toc24457834) 6

2. [Описание структур данных](#_Toc24457835) 7

3. [Описание алгоритмов](#_Toc24457836) 7

V. [Эксперименты](#_Toc24457837) 10

VI. [Заключение](#_Toc24457838) 11

VII. [Литература](#_Toc24457839) 12

VIII. [Приложение 1](#_Toc24457840)3

# 1. Введение

Работа с геометрическими объектами в N-мерном пространстве, это важная и сложная задача. Автоматизация процессов расчёта различных геометрических объектов и тел крайне важна, и в этом смысле концепция ООП предоставляет лучшие возможности. Именно для этого необходимо продумать возможность работать на прямую с объектами типа «Точка», «Прямая». «Квадрат» и т д.

# 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, Circle, Square, Cube или Polygon, в зависимости от необходимости.
2. Наполнить выбранные объекты, используя конструкторы либо методы доступа типа «Сеттеры»
3. Произвести необходимые действия с объектами, при необходимости упаковать их в определенный класс Container.

# 4. Руководство программиста

## 4.1. Описание структуры программы

Программа состоит из одного решения.

В решении Laba4 определены 18 модулей main.cpp, lin\_null.h, Point.h, point\_doing.h, Line.h, line\_doing.h, Triangle.h, triangle\_doing.h, Square.h, square\_doing.h, Cube.h, cube\_doing.h, Circle.h, circle\_doing.h, Polygon.h, polygon\_doing.h, Container.h, container\_doing.h

* В модуле main.cpp определена стандартная функция int main(), внутри которой содержится набор действий с геометрическими фигурами.
* В модуле lin\_null.h определен класс базовый класс, а также объявлены все виртуальные методы.
* В модуле Point.h определен одноименный класс- наследник lin\_null.
* В модуль point\_doing.h вынесены все определения методов класса Point.
* В модуле Line.h определен одноименный класс- наследник Point.
* В модуль line\_doing.h вынесены все определения методов класса Line.
* В модуле Circle.h определен одноименный класс- наследник Point.
* В модуль circle\_doing.h вынесены все определения методов класса Circle.
* В модуле Triangle.h определен одноименный класс- наследник Line.
* В модуль triangle\_doing.h вынесены все определения методов класса Triangle.
* В модуле Square.h определен одноименный класс- наследник Point.
* В модуль square\_doing.h вынесены все определения методов класса Square.
* В модуле Line.h определен одноименный класс- наследник Point.
* В модуль line\_doing.h вынесены все определения методов класса Line.
* В модуле Cube.h определен одноименный класс- наследник Square.
* В модуль cube\_doing.h вынесены все определения методов класса Square.
* В модуле Polygon.h определен одноименный класс- наследник Line.
* В модуль polygon\_doing.h вынесены все определения методов класса Polygon.

## Описание структур данных

В программе определен один абстрактный класс:

lin\_null

Внутри класса lin\_null определены следующие виртуальные функции:

* virtual ~lin\_null(){} – виртуальный деструктор
* virtual float Area() = 0 - виртуальный метод вычисления площади
* virtual float V() = 0 – виртуальный метод вычисления объёма
* virtual ostream& print(ostream& out) = 0 - виртуальный метод вывода принт
* friend ostream& operator<<(ostream& out, lin\_null& rhs) – 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\* koorF– шаблонный указатель.
* int mer– 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(First&, int) – конструктор инциализатор, принимает на вход два параметра типа First& и типа int создает динамический массив заданных размеров, заполняет его соответственным элементом переданным в конструктор, поле mer инициализируется значением переданным в конструктор соответственно.
* Point(Point&) – конструктор копирования, принимает на вход объект типа Point, создает объект с теми же характеристиками.
* Point(First\* we, int size) – конструктор, принимает на вход массив и его размер, инициализирует свои поля соответственно.
* ~Point() – деструктор, очищает выделенную методами и конструкторами память.
* bool operator !=(const Point& rhs) – перегрузка оператора !=.
* ostream& print(ostream& out)override - override функция вывода.
* Point& operator =(const Point& rhs) – перегрузка оператора присвоения для объекта типа Point.
* Point& operator =(const int rhs) – перегрузка оператора присвоения для обьектов типа int.
* First\* get\_Koor() – метод возвращающий массив с координатами точками.
* int get\_size() – метод возвращающий размерность точки.
* float Area()override – override функция подсчета площади
* float V()override – override функция подсчета обьема.
* void set\_Point(First\* er, int size) – метод изменения координат и размерности точки, «Сеттер».

Внутри класса Line, наследника класса Point определено следующее поле:

* Point koorS

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 koorS вторым параметром.
* Line(Line&) – конструктор копирования, принимает на вход объект типа Line, создает объект с теми же характеристиками.
* ~Line () – деструктор.
* Point<D> get\_PointA() – метод возвращающий начало отрезка.
* Point<D> get\_PointB() - метод возвращающий конец отрезка.
* float module() – метод возвращающий длину отрезка.
* Line& operator =(Line& rhs) – перегрузка оператора присвоения.
* ostream& print(ostream& out) override – override функция вывода
* float Area () override - метод возвращающий площадь отрезка.
* float V()override – override функция подсчета обьема.
* void set\_Line (Point&, Point&) – метод изменения начала и конца точки, «Сеттер».

Внутри класса Circle, наследника класса Point определено следующее поле:

Line<A> length

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<A>&) – конструктор инциализатор, принимает на вход два параметра типа Point& и типа Line<A>& вызывает конструктор копирования Point от первого параметра и инициализирует поле length вторым параметром.
* Circle(Circle&) – конструктор копирования, принимает на вход объект типа Circle, создает объект с теми же характеристиками.
* ~Circle () – деструктор.
* Line<A> get\_Line () – метод возвращающий радиус окружности.
* Point<D> get\_centr () - метод возвращающий центр окружности.
* Circle& operator =(Circle & rhs) – перегрузка оператора присвоения.
* ostream& print(ostream& out) override – override функция вывода
* float Area () override - метод возвращающий площадь отрезка.
* float V()override – override функция подсчета объёма.

Внутри класса Square, наследника класса Point определено следующее поле:

* Point<B> RN

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 RN нулем.
* Square(Point&, Point&) – конструктор инциализатор, принимает на вход два параметра типа Point&, вызывает конструктор копирования Point от первого параметра и инициализирует поле RN вторым параметром.
* Square(Square&) – конструктор копирования, принимает на вход объект типа Square, создает объект с теми же характеристиками.
* ~Square() – деструктор.
* Line<B> get\_Diag () – метод возвращающий начало отрезка.
* Square& operator =(Square& rhs) – перегрузка оператора присвоения.
* ostream& print(ostream& out) override – override функция вывода
* float Area () override - метод возвращающий площадь квадрата.
* float V()override – override функция подсчета обьема.
* void set\_square (Square&) – метод изменения квадрата, «Сеттер».

Внутри класса Triangle, наследника класса Line определено следующее поле:

* Point Top

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<M>&, Point<M>&, Point<M>&) – конструктор инциализатор, принимает на вход три параметра типа Point&, вызывает конструктор Line от первых двух параметров и инициализирует поле Top третьим параметром.
* Triangle(Triangle&) – конструктор копирования, принимает на вход объект типа Triangle, создает объект с теми же характеристиками.
* ~Triangle () – деструктор.
* Point<D> get\_PointA() – метод возвращающий точку основания треугольника.
* Point<D> get\_PointB() - метод возвращающий вторую точку основания.
* Point<D> get\_Top() – метод возвращающий вершину треугольника.
* Triangle& operator =(Trinagle& rhs) – перегрузка оператора присвоения.
* ostream& print(ostream& out) override – override функция вывода
* float Area () override - метод возвращающий площадь отрезка.
* float V()override – override функция подсчета обьема.

Внутри класса Cube, наследника класса Square определено следующее поле:

* Square up

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 up получает значение пустого Square.
* Cube(Point&, Point&, Point&, Point&, Point&, Point&, Point&, Point&) – конструктор инциализатор, принимает на вход восемь параметров типа Point&, вызывает конструктор Square от первого и третьего параметра присваивает значению up 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 Square\_low – метод возвращающий нижний квадрат.
* Square Square\_Up - метод возвращающий верхний квадрат.
* Cube& operator =(Cube & rhs) – перегрузка оператора присвоения.
* ostream& print(ostream& out) override – override функция вывода
* float Area () override - метод возвращающий площадь отрезка.
* float V()override – override функция подсчета обьема.

Внутри класса Polygon, наследника класса Line определены следующие поля:

* Line\* sides
* Int size

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 get\_size– метод возвращающий размерность многочлена.
* Float\* sides\_length - метод возвращающий массив длин сторон многоугольника.
* Line\* sides\_array – метод возвращающий массив сторон многоугольника.
* ostream& print(ostream& out) override – override функция вывода
* float Area () override - метод возвращающий площадь отрезка.
* float V()override – override функция подсчета обьема.
* Polygon& operator = (Polygon&) – перегрузка оператора присвоить.

Внутри класса Container, определены следующие поля:

* Lin\_null\*\* Array
* Int size

Внутри класса Polygon определен следующий набор public-методов

(плюс конструкторы и деструктор):

* Polygon() – конструктор по умолчанию, не принимает никаких параметров.
* Void insert(T&) – принимает на вход шаблонный аргумент, ничего не возвращает, записывает этот объект в Container.
* Void Delete (J&) – принимает на вход шаблонный аргумент, ищет его в контейнере, и удаляет его.
* ~Container () – деструктор.
* Int get\_length– метод возвращающий количество объектов в контейнер.
* Lin\_null\* operator [] – перегрузка оператора возврата объекта по индексу.
* Line\* sides\_array – метод возвращающий массив сторон многоугольника.
* Friend ostream& operator<<(ostream& out, Container& rhs)– перегрузка оператора вывода для класса Container
* Void out(E&) - принимает на вход шаблонный аргумент и выводит его на экран.

## Описание алгоритмов

Как таковых, каких то серьезных алгоритмов в нашем проекте не использовалось.

# Эксперименты

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

# 6. Заключение

По итогу проведения данной работы, нам удалось реализовать удобные объекты для работы с геометрическими фигурами и содержащим их контейнером.

# 7. Литература

1. Павловская Т.А. C/C++, Программирование на языке высокого уровня, 2003.
2. <https://ru.wikipedia.org>
3. <https://ru.stackoverflow.com>
4. <https://learnc.info/c/memory_allocation.html>

# 8. Приложение