МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ им. Н. И. ЛОБАЧЕВСКОГО»

ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МАТЕМАТИКИ И МЕХАНИКИ

Отчёт по лабораторной работе №7

**Классы для работы с геометрическими объектами**

Выполнила:

студентка ф-та ИТММ ПМИ – 381903-3

Исхакова Лана Руслановна

Проверил:

ассистент кафедры МОСТ

Лебедев Илья Геннадьевич

Нижний Новгород

2020 г.

Содержание

[Введение 3](#_Toc24628114)

[Постановка задачи 4](#_Toc24628115)

[Руководство пользователя 5](#_Toc24628116)

[Руководство программиста 6](#_Toc24628117)-9

[Эксперименты 10](#_Toc24628118)

[Заключение 11](#_Toc24628119)

[Литература 12](#_Toc24628120)

[Приложение 13-30](#_Toc24628121)

# 1.Введение

Число является одним из основных понятий математики. Понятие числа развивалось в тесной связи с изучением величин; эта связь сохраняется и теперь. Во всех разделах современной математики и информатики приходится рассматривать разные величины и пользоваться числами.

В этой лабораторной работе мы столкнемся с такими понятиями, как геометрические фигуры. Будет продемонстрирована работа с простейшими геометрическими фигурами, такими, как: точка, линия, круг, квадрат, многоугольник, куб, треугольник.

Для того чтобы работать с геометрическими объектами в программировании необходимы классы. **Класс** — это тип структуры, позволяющий включать в описание типа не только элементы данных, но и функции (функции-элементы или методы).

Также нам понадобятся шаблоны функций и шаблоны классов**. Шаблоны функций** – это обобщенное описание поведения функций, которые могут вызываться для объектов разных типов. **Шаблоны классов** – обобщенное описание пользовательского типа, в котором могут быть параметризованы атрибуты и операции типа.

# 2.Постановка задачи

1. Написать структуру данных для работы с геометрическими объектами в N мерном пространстве. Тип данных определяется пользователем (шаблоны).
2. В программе должны быть реализованы простейшие геометрические объекты, такие, как: точка, линия, круг, квадрат, многоугольник, куб и т.д.
3. Кроме самих геометрических объектов должен быть реализован класс, осуществляющий обобщение действий со всеми созданными пользователями объектами - "контейнер". Контейнер должен иметь следующие функции: добавить новый объект, удалить существующий объект, отобразить все имеющиеся объекты, отобразить выбранный объект и т.п.
4. Продемонстрировать (написать в main пример) работоспособность.
5. Должны быть использованы и продемонстрированы: абстракция, инкапсуляция, наследование, полиморфизм, перегрузка операций, шаблоны.
6. Иерархия должна содержать не менее 7 классов.

# 3.Руководство пользователя

Данная программа написана с помощью Microsoft Visual Studio 2019 на языке C++.

Как пользоваться:

1. Запустить консоль программы «figure».
2. Создать объекты типа point, line, circle, rectangle, square, cube, triangle, в зависимости от того, какие объекты рассматриваются пользователем
3. Наполнить выбранные геометрические объекты, используя конструкторы, также существует возможность в дальнейшем работать с помощью сеттеров
4. Упаковать объекты в контейнер
5. Произвести необходимую работу с объектами

# 

# 4.Руководство программиста

***Описание структуры программы***

Программа состоит из одного решения, которое называется «lab\_GeometricShapes».

В решении содержится 10 элементов: GeometricShapes.cpp, circle.h, container.h, cube.h, line.h, point.h, rectangle.h, shapes.h, square.h, triangle.h.

* В **GeometricShapes.cpp** определена стандартная функция int main()
* В **circle.h** определен класс Circle
* В **container.h** определен класс Container
* В **cube.h** определен класс Cube
* В **line.h** определен класс Line
* В **point.h** определен класс Point
* В **rectangle.h** определен класс Rectangle
* В **shapes.h** определен класс Shapes
* В **square.h** определен класс Square

***Описание структуры программы***

В программе определен один абстрактный базовый класс под названием Shapes.

Внутри этого класса определены следующие виртуальные функции:

• virtual ~ Shapes(){} – виртуальный деструктор;

• virtual float Perimetr() = 0 - виртуальный метод вычисления периметра;

• virtual ostream& print(ostream& off) = 0; - виртуальный метод вывода «Print»;

• friend ostream& operator<<(ostream& off, Shapes& opt) – виртуальный метод вывода;

• bool operator > (Shapes& opt) – перегруженный оператор больше, который сравнивает данные фигуры по периметру;

• bool operator < (Shapes& opt) – перегруженный оператор меньше, который сравнивает данные фигуры по периметру.

В программе также определено 7 шаблонных классов: Point, Line, Circle, Square, Rectangle, Cube, Simplex.

Внутри класса Point определено следующее поле:

• T val – шаблонное значение точки;

Внутри класса Point определены следующий набор public-методов и конструкторы:

• Point () – конструктор по умолчанию, не принимает никаких параметров, инициализирует все два поля 0 через списки инициализации;

• Point (T val) – конструктор-инциализатор, принимает на вход шаблонный параметр и инициализирует поле value значением, переданным в конструктор;

• Point (const Point& point) – конструктор копирования, принимает на вход объект типа Point, создает объект с теми же характеристиками;

• bool operator ==(const Point& opt) – перегрузка оператора = =;

• ostream& print(ostream& off) override - override функция вывода;

• T Get() – метод возвращающий значение точки;

• float Perimetr() override – override функция подсчета периметра;

Внутри класса Line определены следующие поля:

• int length – длина отрезка;

• T\* arr – шаблонный динамический массив значений отрезка;

Внутри класса Line определены следующий набор public-методов и конструкторы с деструктором:

• Line () – конструктор по умолчанию, не принимает никаких параметров, инициализирует поле length восьмеркой, а массив заполняет символом «-»;

• Line (int leng, T val) – конструктор-инциализатор, принимает на вход параметр тип int и один шаблонный параметр, инициализирует поле length и заполняет массив;

• Line (const Line& line) – конструктор копирования, принимает на вход объект типа Line, создает объект с теми же характеристиками;

• ~Line () – деструктор;

• int GetLeng () – метод возвращающий длину отрезка;

• T GetValue () – метод возвращающий значение массива;

• bool operator ==(const Line& opt) – перегрузка оператора ==;

• ostream& print(ostream& off) override – override функция вывода;

• float Perimetr() override - метод возвращающий периметр отрезка;

Внутри класса Circle определены следующие поля:

• int Diametr –диаметр круга;

• T\*\* arr – шаблонный двойной динамический массив значений круга;

Внутри класса Circle определены следующий набор public-методов и конструкторы с деструктором:

• Circle () – конструктор по умолчанию, не принимает никаких параметров, инициализирует поле Diametr двойкой, а массив заполняет символом «@»;

• Circle (int diam, T val) – конструктор-инциализатор, принимает на вход параметр типа int и шаблонный параметр, инициализирует поле Diametr и заполняет массив;

• Circle (const Circle& circle) – конструктор копирования, принимает на вход объект типа Circle, создает объект с теми же характеристиками;

• ~Circle () – деструктор;

• int GetDiametr () – метод, возвращающий диаметр круга;

• T GetValue () – метод, возвращающий значение массива;

• void SetDiametr (int \_Diametr) – метод, изменяющий диаметр круга;

• bool operator ==(const Circle& opt) – перегрузка оператора ==;

• ostream& print(ostream& off) override – override функция вывода;

• float Perimetr() override - метод возвращающий периметр круга;

Внутри класса Square определены следующие поля:

• int storona – сторона квадрата;

• T\*\* arr – шаблонный двойной динамический массив значений квадрата;

Внутри класса Square определены следующий набор public-методов и плюс конструкторы с деструктором:

• Square () – конструктор по умолчанию, не принимает никаких параметров, инициализирует поле storona двойкой, а массив заполняет символом «0»;

• Square (int s, T val) – конструктор-инциализатор, принимает на вход параметр типа int и шаблонный параметр, инициализирует поле storona и заполняет массив;

• Square (const Square & square) – конструктор копирования, принимает на вход объект типа Square, создает объект с теми же характеристиками;

• ~Square () – деструктор;

• int Getstorona() – метод, возвращающий длину стороны квадрата;

• T GetValue () – метод, возвращающий значение массива;

• void Setstorona(int \_storona) – метод, изменяющий длину стороны квадрата;

• bool operator ==(const Square& opt) – перегрузка оператора ==;

• ostream& print(ostream& off) override – override функция вывода;

• float Perimetr() override - метод, возвращающий периметр квадрата;

Внутри класса Simplex определены следующие поля:

• int height – ребро прямоугольного симплекса +1 (в виде тетраэдра);

• T\*\* arr – шаблонный двойной динамический массив значений симплекса;

Внутри класса Simplex определены следующий набор public-методов и конструкторы с деструктором:

• Simplex() – конструктор по умолчанию, не принимает никаких параметров, инициализирует поле height четверкой, а массив заполняет символом «^»;

• Simplex (int \_height, T val) – конструктор-инциализатор, принимает на вход параметр типа int и шаблонный параметр, инициализирует поле height и заполняет массив;

• Simplex (const Simplex& simplex) – конструктор копирования, принимает на вход объект типа Simplex, создает объект с теми же характеристиками;

• ~ Simplex() – деструктор;

• int GetHeight () – метод, возвращающий длину высоты симплекса;

• T GetValue () – метод, возвращающий значение массива;

• void SetHeight(int \_height) – метод, изменяющий длину высоты симплекса;

• bool operator ==(const Simplex& opt) – перегрузка оператора ==;

• ostream& print(ostream& off) override – override функция вывода (возвращает проекцию симплекса на плоскость);

• float Perimetr() override - метод, возвращающий периметр поверхности симплекса;

Внутри класса Rectangle определены следующие поля:

• int storona1 – первая сторона прямоугольника;

• int storona2 – вторая сторона прямоугольника, отличная от предыдущей;

• T\*\* Arr – шаблонный двойной динамический массив значений прямоугольника;

Внутри класса Rectangle определены следующий набор public-методов и конструкторы с деструктором:

• Rectangle () – конструктор по умолчанию, не принимает никаких параметров, инициализирует поле storona1 и storona2 единицей, а массив заполняет символом «1»;

• Rectangle (int \_storona1, int \_storona2, T val) – конструктор-инциализатор, принимает на вход два параметра типа int и шаблонный параметр, инициализирует поля storona1 и storona2 и заполняет массив;

• Rectangle (const Rectangle & rectangle) – конструктор копирования, принимает на вход объект типа Rectangle, создает объект с теми же характеристиками;

• ~ Rectangle () – деструктор;

• int Getstorona1() – метод, возвращающий длину первой стороны прямоугольника;

• int Getstorona2() – метод, возвращающий длину второй стороны прямоугольника;

• T GetValue () – метод, возвращающий значение массива;

• void SetStorona(int \_storona1, int \_storona2) – метод, изменяющий длины сторон прямоугольника;

• bool operator ==(const Rectangle & opt) – перегрузка оператора ==;

• ostream& print(ostream& off) override – override функция вывода;

• float Perimetr() override - метод, возвращающий периметр прямоугольника;

Внутри класса Cube определены следующие поля:

• int rebro – ребро куба;

• T\*\*\* arr – шаблонный тройной динамический массив значений куба;

Внутри класса Cube определены следующий набор public-методов и конструкторы с деструктором:

• Cube () – конструктор по умолчанию, не принимает никаких параметров, инициализирует поле rebro пятеркой, а массив заполняет символом «&»;

• Cube (int e, T val) – конструктор-инциализатор, принимает на вход параметр типа int и шаблонный параметр, инициализирует поле rebro и заполняет массив;

• Cube (const Cube & cube) – конструктор копирования, принимает на вход объект типа Cube, создает объект с теми же характеристиками;

• ~ Cube () – деструктор;

• int Getrebro () – метод, возвращающий длину ребра куба;

• T GetValue () – метод, возвращающий значение массива;

• void Setrebro (int \_rebro) – метод, изменяющий длину ребра куба;

• bool operator ==(const Cube & opt) – перегрузка оператора ==;

• ostream& print(ostream& off) override – override функция вывода (возвращает проекцию куба на плоскость);

• float Perimetr() override - метод, возвращающий периметр поверхности куба;

В программе определен один класс-контейнер под названием Container.

Внутри данного класса определены следующие поля:

• Shapes\*\* arr;

• int size;

Внутри класса Container определены следующий набор public-методов и конструкторы с деструктором:

• Container () – конструктор по умолчанию, не принимает никаких параметров;

• void Add(A& element) – принимает на вход шаблонный аргумент, ничего не возвращает, записывает этот объект в Container;

• void Delete(A& element) – принимает на вход шаблонный аргумент, ищет его в контейнере и удаляет его;

• ~Container () – деструктор;

• Shapes\* operator[](const int i) – перегрузка оператора возврата объекта по индексу;

• friend ostream& operator<<(ostream& off, Container& opt) – перегрузка оператора вывода для класса Container;

•void off(A& element) - принимает на вход шаблонный аргумент и выводит его на экран.

***Описание алгоритмов***

Алгоритмы в данной лабораторной работе не требуют подробного описания.

# 5.Эксперименты

Пример работы программы:

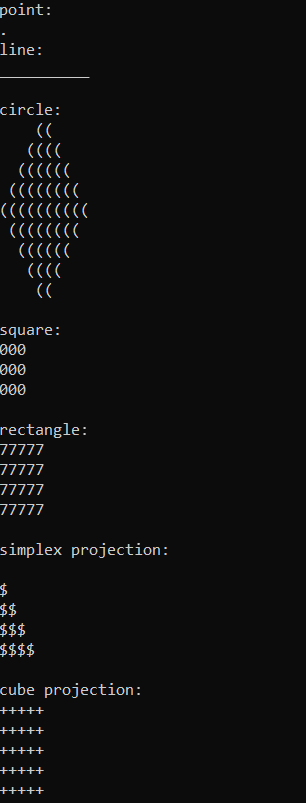


Рис.1. Пример работы программы

# 6.Заключение

Таким образом, в ходе лабораторной работы была создана программа, которая продемонстрировала работу с классами геометрических фигур в N-мерном пространстве. Были использованы ООП (абстракция, инкапсуляция, наследование и полиморфизм).

# 7.Литература

1. С. Н. Марков. Курс истории математики: Учебное пособие. - Иркутск: Издательство иркутского университета, 1995. 248 с.
2. <http://www.math24.ru/множества-чисел.html>
3. Павловская Т.А. C/C++, Программирование на языке высокого уровня, 2003
4. <https://ru.wikipedia.org>