Федеральное агентство по образованию Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского» (ННГУ им. Н.И. Лобачевского)

Институт информационных технологий, математики и механики

Направление подготовки: «Фундаментальная информатика и информационные технологии»

Отчёт по лабораторной работе

Создание структуры данных для работы с геометрическими объектами в пространстве.

Выполнила:

студентка ф-та ИИТММ

Гордей М.В

Проверил:

ассистент к.математического обеспечения и суперкомпьютерных технологий

Лебедев И.Г.

Нижний Новгород

2020 г.

Содержание

1.[Введение 3](#_Toc24469347)

2.[Постановка задачи 4](#_Toc24469348)

3[Руководство пользователя 5](#_Toc24469349)

4[Руководство программиста 6](#_Toc24469350)

4.1.[Описание структуры программы 6](#_Toc24469351)

4.2[Описание структур данных 6](#_Toc24469352)

4.3[Описание алгоритмов 6](#_Toc24469353)

5.[Эксперименты 7](#_Toc24469354)

6.[Заключение 8](#_Toc24469355)

7.[Литература 9](#_Toc24469356)

8.[Приложения 10](#_Toc24469357)

[Приложение 1 10](#_Toc24469358)

# 1.Введение

Реализация работы с геометрическими объектами в N- мерном пространстве – важная задача в объектно-ориентированном программировании. Данная работа позволяет собрать и объединить сведения о геометрических объектах, осуществить работу с простейшими из них, таких, как точка, прямая, круг и так далее.

# 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, симплекс и т.д.

# 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. Иерархия должна содержать не менее классов.

# 3.Руководство пользователя

Действия, исполняемые пользователем для выполнения работы программы:

1. Создать объекты типа Point, Line, Circle, Square, Sphere, Cone или Triangle, в зависимости от необходимости.
2. Наполнить создаваемые объекты при помощи прописанных в программе методов, реализованных в программе
3. При необходимости упаковать объекты в определенный класс Contanier
4. Произвести с объектами необходимые действия

**4.Руководство программиста**

4.1. Описание структуры программы

### *4.1. Описание структуры программы*

Программа состоит из одного решения.

В решении Vector определено 18 модулей main.cpp,Base.h, Point.h, Point\_i.h, Line.h, Line\_i.h, Circle.h, Circle\_i.h, Square.h, Square\_i.h, Triangle.h, Triangle\_i.h,Cone.h, Cone\_i.h, Sphere.h, Sphere\_i.h, Container.h Container\_i.h

* В модуле main.cpp определена стандартная функция int main(), внутри которой содержится набор действий с фигурами.
* В модуле Point.h определен класс Point,
* В модуле Point\_i.h объявлены все его методы и их определения.
* В модуле Line.h определен класс Point,
* В модуле Line\_i.h объявлены все его методы и их определения.
* В модуле Circle.h определен класс Circle
* В модуле Circle\_i.h объявлены все его методы и их определения.
* В модуле Square.h определен класс Square
* В модуле Square\_i.h объявлены все его методы и их определения.
* В модуле Triangle.h определен класс Triangle
* В модуле Triangle\_i.h объявлены все его методы и их определения.
* В модуле Cone.h определен класс Cone
* В модуле Cone\_i.h объявлены все его методы и их определения.
* В модуле Sphere.h определен класс Sphere
* В модуле Sphere\_i.h объявлены все его методы и их определения.
* В модуле Base.h определен базовый абстрактный класс, и объявлены абстрактные методы класса.
* В модуле Container.h определен класс Container
* В модуле Container \_i.h объявлены все его методы и их определения.

*4.2.Описание структур данных*

В программе определен один абстрактный класс:

Base

Внутри класса Base определены следующие виртуальные функции:

* ~Base(){} –деструктор
* virtual ostream& print(ostream& os) = 0; виртуальный метод вывода принт
* friend ostream& operator<<(ostream& os, Base& other) – виртуальный метод вывода

Внутри класса Point определено следующее поле:

* T\*A – шаблонный указатель;
* Int dim – размерность пространства

Внутри класса Point определен следующий набор public-методов

(плюс конструкторы):

* Point () – конструктор по умолчанию, не принимает никаких параметров, инициализирует все два поля 0, через списки инициализации;
* Point (int n) – конструктор инциализатор, принимает на вход шаблонный параметр и инициализирует поле mark значением переданным в конструктор соответственно;
* Point (const Point& p) – конструктор копирования, принимает на вход объект типа Point, создает объект с теми же характеристиками;
* virtual ~Point()- виртуальный деструктор;
* ostream& print(ostream& os) override - override функция вывода;
* T Get A() – метод возвращающий координаты точки;
* int GetDim()- const метод возвращающий размерность точки
* T\*& SetA()- метод изменения координат точки
* int& SetDim() ()- метод изменения размерности точки;

Внутри класса Line, наследника класса Point определено следующее поле:

* T\*B – шаблонный динамический массив значений отрезка

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 () – конструктор по умолчанию, не принимает никаких параметров, инициализирует поле length 10, а массив заполняет символом \*;
* Line (int n) – конструктор инциализатор, принимает на вход параметр тип int инициализирует поле n
* Line (const Line& l) – конструктор копирования, принимает на вход объект типа Line, создает объект с теми же характеристиками;
* virtual~Line () – виртуальный деструктор;
* double GetLength () – метод возвращающий длину отрезка;
* T GetB () – метод возвращающий значение массива;
* T\*& SetB()- метод изменения отрезка;
* ostream& print(ostream& os) override – override функция вывода;

Внутри класса Circle, наследника класса Point определено следующее поле:

* Line<T>

Внутри класса Circle определен следующий набор public-методов

(плюс конструкторы и деструктор):

* Circle () – конструктор по умолчанию, не принимает никаких параметров, инициализирует поле radius двойкой, а массив заполняет символом \*;
* Circle (int n) – конструктор инциализатор, принимает на вход параметр типа int
* Circle (const Circle& c) – конструктор копирования, принимает на вход объект типа Circle, создает объект с теми же характеристиками;
* virtual~Circle () – деструктор;
* double GetArea () - метод возвращающий площадь круга;
* double GetRadius() – метод возвращающий радиус круга;
* ostream& print(ostream& os) override – override функция вывода;

Внутри класса Square определены следующие поля:

Triangle<T>

* T\*\* D – шаблонный двойной динамический массив значений квадрата;

Внутри класса Square определен следующий набор public-методов

(плюс конструкторы и деструктор):

* Square () – конструктор по умолчанию, не принимает никаких параметров, инициализирует поле side единицей, а массив заполняет символом \*;
* Square (int n) – конструктор инциализатор
* Square (const Square & q) – конструктор копирования, принимает на вход объект типа Square, создает объект с теми же характеристиками;
* virtual~Square () – деструктор;
* T\* GetD () – метод возвращающий значение массива;
* void SetD– метод изменяющий длину стороны квадрата;
* ostream& print(ostream& os) override – override функция вывода;
* double GetArea () override - метод возвращающий площадь квадрата;
* double GetPerimeter() - метод возвращающий периметр квадрата;

Внутри класса Triangle определены следующие поля:

Line<T>

* T\*\* C – шаблонный двойной динамический массив значений квадрата;

Внутри класса Triangle определен следующий набор public-методов

(плюс конструкторы и деструктор):

* Triangle () – конструктор по умолчанию, не принимает никаких параметров, инициализирует поле side единицей, а массив заполняет символом \*;
* Triangle (int n) – конструктор инциализатор
* Triangle (const Square & q) – конструктор копирования, принимает на вход объект типа Square, создает объект с теми же характеристиками;
* virtual~ Triangle () – деструктор;
* T\* GetD() const– метод возвращающий значение массива;
* T\*& SetD() метод изменяющий длину стороны треугольника;
* double GetPerimeter() const метод возвращающий периметр треугольника;
* double GetArea() const - override - метод возвращающий площадь треугольника;

Внутри класса Cone определено следующее поле: Triangle<T>

Внутри класса Cone определен следующий набор public-методов

(плюс конструкторы и деструктор):

* Cone() – конструктор по умолчанию, не принимает никаких параметров, инициализирует поле side1 и side2 единицей, а массив заполняет символом \*;
* Cone (int n) – конструктор инциализатор
* Cone (const Cone & e) – конструктор копирования, принимает на вход объект типа Cone, создает объект с теми же характеристиками;
* virtual~ Cone () – деструктор;
* ostream& print(ostream& os) override – override функция вывода;
* double GetRadius() const – метод, возвращающий радиус;
* double GetHeight() const метод, возвращающий высоту;
* double GetVolume() const метод, возвращающий объем;
* double GetArea() const метод, возвращающий площадь;

Внутри класса Sphere определено следующее поле: Circle<T>

Внутри класса Sphere определен следующий набор public-методов

(плюс конструкторы и деструктор):

* Sphere () – конструктор по умолчанию, не принимает никаких параметров, инициализирует поле side единицей, а массив заполняет символом \*;
* Sphere (int n) – конструктор инциализатор, принимает на вход параметр типа int и шаблонный параметр, инициализирует поле side, и заполняет массив соответственно;
* Sphere (const Sphere &s) – конструктор копирования, принимает на вход объект типа Sphere, создает объект с теми же характеристиками;
* ~ Sphere () – деструктор;
* double GetVolume() const – метод, возвращающий объем;
* ostream& print(ostream& os) override – override функция вывода (возвращает проекцию сферы на плоскость);

В программе определен один класс-контейнер: Container

Внутри класса Container, определены следующие поля:

* Base \*\* Array;
* int size;

Внутри класса Container определен следующий набор public-методов

(плюс конструкторы и деструктор):

* Container () – конструктор по умолчанию, не принимает никаких параметров;
* void Add(A& elem) – принимает на вход шаблонный аргумент, ничего не возвращает, записывает этот объект в Container;
* void Remove(int r) – принимает на вход шаблонный аргумент, ищет его в контейнере, и удаляет его;
* ~Container () – деструктор;
* Objects\* operator[](const int i) – перегрузка оператора возврата объекта по индексу;
* void ShowAll() – показыват список;
* void Show(int i) показывает объект;
  1. Описание алгоритмов

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

# 

**5.Эксперименты.**

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

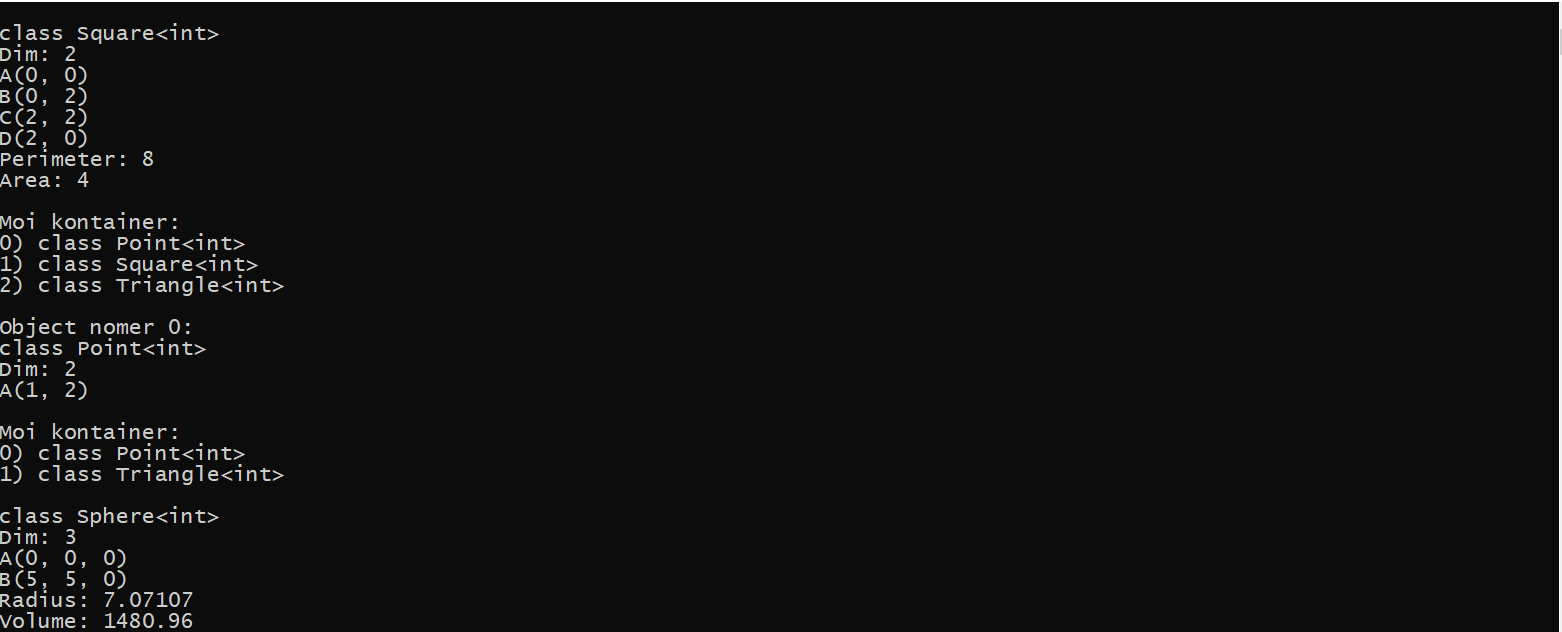


Рисунок 1: Результаты выполнения функции Main.

**6.Заключение**

Таким образом, в рамках данной лабораторной работы была успешно создана программа, которая позволяет работать с геометрическими объектами, такими, как точка, линия, круг, квадрат, прямоугольник, куб в N-мерном пространстве. Были использованы и продемонстрированы основные принципы объектно-ориентированного программирования. Мне удалось создать программу, которая создает структуру данных для работы с геометрическими фигурами. У пользователя есть возможность объединять созданные фигуры в контейнере, а также производить с фигурами различные операции. Используя принципы ООП (абстракцию, инкапсуляцию, наследование и полиморфизм) были созданы удобные объекты для работы с геометрическими фигурами и содержащим их контейнером.

# 

**7.Литература**

Павловская Т.А. C/C++, Программирование на языке высокого уровня, 2003.

Вирт Н. Алгоритмы + структуры данных = программы. — М.: Мир, 1985. — С. 406.

Официальный сайт программы PMBK. – Режим доступа <http://www.pmbk.ru/pr/chto-takoe-massivy-v-programmirovanii.html>

Официальный сайт Habr. – Режим доступа <https://habr.com/ru/post/339656/>

Официальный сайт Microsoft. – Режим доступа <https://docs.microsoft.com/ru-ru/cpp/c-language/main-function-and-program-execution?view=vs-2019>

**8.Приложение**