Московский финансово-юридический университет

Ярославский филиал Аккредитованного образовательного частного

учреждения высшего образования "Московский финансово-юридический

университет МФЮА"

Лабораторная работа №2

По предмету "Алгоритмизация и программирование"

"Вариант № 6"

Выполнил: студент группы 31ПРД4410

Поспелов И.И. и Федоров К.А.

Проверил: Фурмавнин С.А.

Ярославль, 2025

1. Цель

Построение фигуры с помощью кода. Треугольник → Треугольная пирамида → Усеченная треугольная пирамида.

2. Теоретические сведения

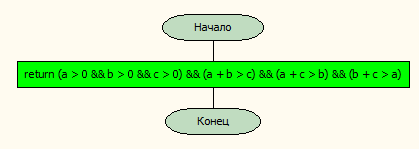
Неравенство треугольника - сумма любых двух сторон должна быть больше третьей. Формула Герона для вычисления площади: S = √(p(p-a)(p-b)(p-c)), где p = (a+b+c)/2 – полупериметр. Объем пирамиды: V = (1/3) \* S\_осн \* h. Объем усеченной пирамиды: V = (h/3) \* (S\_осн + S\_верх + √(S\_осн\*S\_верх)).

3. Расчетно-графическая часть

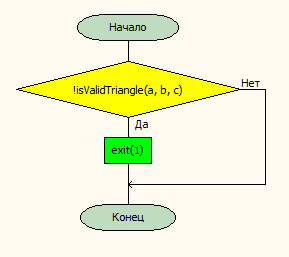
В файле geometry.h описаны следующие классы и функции:

Класс Triangle: хранит информацию о трёх переменных – сторонах треугольника.

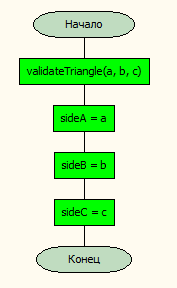
Функция isValidTriangle – условия существования треугольника.



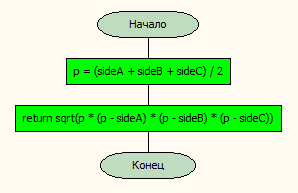
Функция validateTriangle – проверка ссуществования треугольника с заданными сторонами.



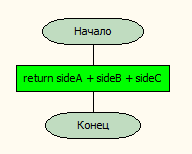
Конструктор Triangle – принимает стороны треугольника a, b и c, проверяет корректность.



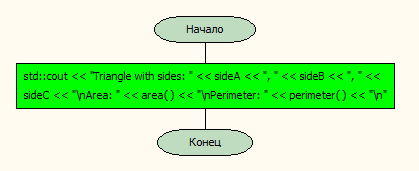
Функция area – вычисление площади треугольника.



Функция perimeter – вычисление периметра треугольника.

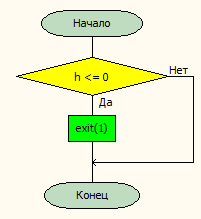


Функция printInfo – вывод информации о сторонах треугольника, площади и периметре.

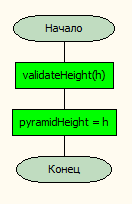


Класс TriangularPyramid: хранит информацию о сторонах пирамиды и её высоте.

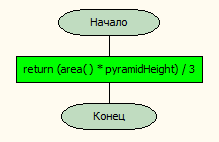
Функция validateHeight – проверка правильности высоты.



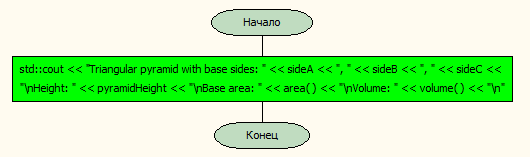
Конструктор TriangularPyramid – принимает стороны пирамиды a, b, c и её высоту h, проверяет корректность.



Функция volume – вычисляет объём пирамиды.

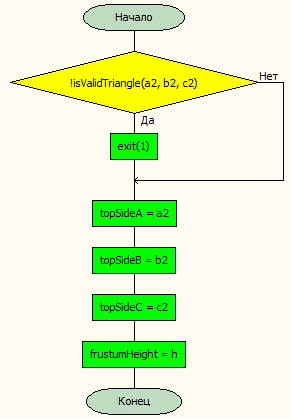


Функция printInfo – вывод информации о сторонах пирамиды, площади основания и объёме.

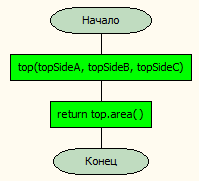


Класс FrustumTriangularPyramid: хранит информацию о сторонах нижнего и верхнего оснований пирамиды, а также о её высоте.

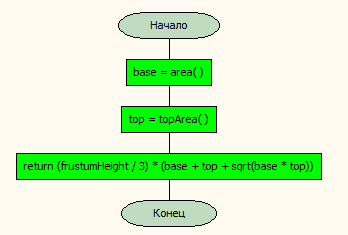
Конструктор FrustumTriangularPyramid – принимает стороны оснований и высоту, проверяет корректность.



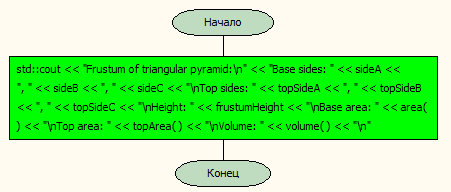
Функция topArea – считает площадь верхнего основания пирамиды.



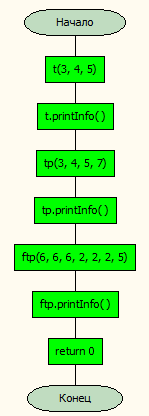
Функция volume – вычисляет объём усечённой пирамиды.



Функция printInfo – вывод информации о сторонах нижнего и верхнего оснований пирамиды, площади оснований и объёме.



В файле main.cpp находится основная часть кода. Создаём треугольник, треугольную пирамиду и усечённую треугольную пирамиду, а также выводим информацию о них:



4. Выводы по работе.

В ходе лабораторной работы нами был разработан код, который выводит основную информацию о трёх фигурах: треугольнике, треугольной пирамиде и усечённой треугольной пирамиде. Программа выполняет все нужные вычисления и выводит правильный результат, что было проверено вручную.

Приложение 1 – Листинг кода программы

**geometry.h:**

#ifndef GEOMETRY\_H

#define GEOMETRY\_H

#include <iostream>

#include <cmath>

class Triangle {

protected:

double sideA, sideB, sideC;

bool isValidTriangle(double a, double b, double c) const {

return (a > 0 && b > 0 && c > 0) &&

(a + b > c) && (a + c > b) && (b + c > a);

}

void validateTriangle(double a, double b, double c) const {

if (!isValidTriangle(a, b, c)) {

std::cerr << "Error: Invalid triangle sides\n";

exit(1);

}

}

public:

Triangle(double a, double b, double c) {

validateTriangle(a, b, c);

sideA = a;

sideB = b;

sideC = c;

}

double area() const {

double p = (sideA + sideB + sideC) / 2;

return sqrt(p \* (p - sideA) \* (p - sideB) \* (p - sideC));

}

double perimeter() const {

return sideA + sideB + sideC;

}

void printInfo() const {

std::cout << "Triangle with sides: " << sideA << ", " << sideB

<< ", " << sideC << "\nArea: " << area()

<< "\nPerimeter: " << perimeter() << "\n";

}

};

class TriangularPyramid : public Triangle {

private:

double pyramidHeight;

void validateHeight(double h) const {

if (h <= 0) {

std::cerr << "Error: Height must be positive\n";

exit(1);

}

}

public:

TriangularPyramid(double a, double b, double c, double h) : Triangle(a, b, c) {

validateHeight(h);

pyramidHeight = h;

}

double volume() const {

return (area() \* pyramidHeight) / 3;

}

void printInfo() const {

std::cout << "Triangular pyramid with base sides: " << sideA << ", "

<< sideB << ", " << sideC << "\nHeight: " << pyramidHeight

<< "\nBase area: " << area() << "\nVolume: " << volume() << "\n";

}

};

class FrustumTriangularPyramid : public TriangularPyramid {

private:

double topSideA, topSideB, topSideC;

double frustumHeight;

public:

FrustumTriangularPyramid(double a1, double b1, double c1,

double a2, double b2, double c2,

double h) : TriangularPyramid(a1, b1, c1, h) {

if (!isValidTriangle(a2, b2, c2)) {

std::cerr << "Error: Invalid top triangle sides\n";

exit(1);

}

topSideA = a2;

topSideB = b2;

topSideC = c2;

frustumHeight = h;

}

double topArea() const {

Triangle top(topSideA, topSideB, topSideC);

return top.area();

}

double volume() const {

double base = area();

double top = topArea();

return (frustumHeight / 3) \* (base + top + sqrt(base \* top));

}

void printInfo() const {

std::cout << "Frustum of triangular pyramid:\n"

<< "Base sides: " << sideA << ", " << sideB << ", " << sideC

<< "\nTop sides: " << topSideA << ", " << topSideB << ", " << topSideC

<< "\nHeight: " << frustumHeight

<< "\nBase area: " << area()

<< "\nTop area: " << topArea()

<< "\nVolume: " << volume() << "\n";

}

};

#endif

**main.cpp:**

#include "geometry.h"

int main() {

// Создаем треугольник

Triangle t(3, 4, 5);

t.printInfo();

// Создаем треугольную пирамиду

TriangularPyramid tp(3, 4, 5, 7);

tp.printInfo();

// Создаем усеченную треугольную пирамиду

FrustumTriangularPyramid ftp(6, 6, 6, 2, 2, 2, 5);

ftp.printInfo();

return 0;

}