Здесь будет титульник, листай ниже

СОДЕРЖАНИЕ

1 ПОСТАНОВКА ЗАДАЧИ

Перегрузка арифметических операций.

Перезагрузка операции для объекта треугольник.

У треугольника есть стороны a, b, c и они принимают только натуральные значения. Определяем операцию сложения и вычитания для треугольников.

+ сложить значения сторон, если допустимо.

- вычесть значения сторон, если допустимо.

Складываются и вычитаются соответствующие стороны треугольников. Т.е. a1 + a2, b1 + b2, c1 + c2. Если после выполнения операции получается недопустимый треугольник, то результатом операции берется первый аргумент.

Написать программу, которая выполняет операции над треугольниками.

В основной программе реализовать алгоритм:

1. Ввод количества треугольников n.
2. В цикле для каждого треугольника вводятся исходные длины сторон. Далее создается объект, в конструктор которого передаются значения длин сторон. Каждый объект треугольника получает свой номер от 1 до n.
3. В цикле, последовательно, построчно вводится «номер первого треугольника» «символ арифметической операции + или -» «номер второго треугольника»
4. После каждого ввода выполняется операция, результат присваивается первому аргументу (объекту треугольника).
5. Цикл завершается по завершению данных.
6. Выводится результат последней операции.

Гарантируется:

- Количество треугольников больше или равно 2;

- Значения исходных длин сторон треугольников задаются корректно.

Реализовать перегрузку арифметических операции «+» и «-» для объектов треугольника посредством самостоятельных не дружественных функций.

1.1 Описание входных данных

Первая строка содержит значение количества треугольников n:

«Натуральное значение»

Далее n строк содержат

«Натуральное значение» «Натуральное значение» «Натуральное значение»

Начиная с n + 2 строки:

«Натуральное значение» «Знак операции» «Натуральное значение»

1.2 Описание выходных данных

a = «Натуральное значение»; b = «Натуральное значение»; c = «Натуральное значение».

2 МЕТОД РЕШЕНИЯ

Для решения задачи используется:

- функция operator+ для Перегрузка оператора +;
- функция operator- для Перегрузка оператора -;
- vector - типизированный контейнер последовательностей.

Класс triangle:

- функционал:
    - метод getA — Геттер значения поля a;
    - метод getB — Геттер значения поля b;
    - метод getC — Геттер значения поля c.

3 ОПИСАНИЕ АЛГОРИТМОВ

Согласно этапам разработки, после определения необходимого инструментария в разделе «Метод», составляются подробные описания алгоритмов для методов классов и функций.

3.1 Алгоритм метода getA класса triangle

Функционал: Геттер значения поля a.

Параметры: нет.

Возвращаемое значение: int - значение поля.

Алгоритм метода представлен в таблице 1.

Таблица 1 – Алгоритм метода getA класса triangle

№	Предикат	Действия	№ перехода
1		Возврат значения поля a	∅

3.2 Алгоритм метода getB класса triangle

Функционал: Геттер значения поля b.

Параметры: нет.

Возвращаемое значение: int - значение поля.

Алгоритм метода представлен в таблице 2.

Таблица 2 – Алгоритм метода getB класса triangle

№	Предикат	Действия	№ перехода
1		Возврат значения поля c	∅

3.3 Алгоритм метода getC класса triangle

Функционал: Геттер значения поля c.

Параметры: нет.

Возвращаемое значение: int - значение поля.

Алгоритм метода представлен в таблице 3.

Таблица 3 – Алгоритм метода getC класса triangle

№	Предикат	Действия	№ перехода
1		Возврат значения поля c	∅

3.4 Алгоритм функции operator-

Функционал: Перегрузка оператора - для вычитания сторон двух треугольников.

Параметры: triangle t1, triangle t2 - вычитаемые треугольники.

Возвращаемое значение: triangle - новый треугольник.

Алгоритм функции представлен в таблице 4.

Таблица 4 – Алгоритм функции operator-

№	Предикат	Действия	№ перехода
1		Инициализация объекта t класса triangle с передачей параметризированному конструктору параметров значение вызова метода getA у объекта t1 + значение вызова метода getA у объекта t2, значение вызова метода getB у объекта t1 + значение вызова метода getB у объекта t2, значение вызова метода getC у объекта t1 +значение вызова метода getC у объекта t2.	2
2		Инициализация переменных t1a, t1b, t1c типа int	3

№	Предикат	Действия	№ перехода
		значениями вызова метода getA у объекта t1, вызова метода getB у объекта t1, вызова метода getC у объекта t1	
3		Инициализация переменных t2a, t2b, t2c типа int значениями вызова метода getA у объекта t2, вызова метода getB у объекта t2, вызова метода getC у объекта t2	4
4		Инициализация объекта t класса triangle с передачей t1a - t2a, t1b - t2b, t1c - t2c параметризированному конструктору	5
5		Инициализация переменных ta, tb, tc типа int значениями вызова метода getA у объекта t, вызова метода getB у объекта t, вызова метода getC у объекта t	6
6	ta + tb <= tc || ta + tc <= tb || tc + tb <= tc || ta <= 0 || tb <= 0 || tc <= 0	Возврат t1	∅
		Возврат t	∅

3.5 Алгоритм функции operator+

Функционал: Перегрузка оператора + для сложения сторон двух треугольников.

Параметры: triangle t1, triangle t2 - складываемые треугольники.

Возвращаемое значение: triangle - новый треугольник.

Алгоритм функции представлен в таблице 5.

Таблица 5 – Алгоритм функции operator+

№	Предикат	Действия	№ перехода
1		Инициализация объекта t класса triangle с передачей параметризированному конструктору параметров значение вызова метода getA у объекта t1 + значение вызова метода getA у объекта t2, значение вызова метода getB у объекта t1 + значение вызова метода getB у объекта t2, значение вызова метода getC у объекта t1 +значение вызова метода getC у объекта t2.	2
2		Инициализация переменных t1a, t1b, t1c типа int значениями вызова метода getA у объекта t1, вызова метода getB у объекта t1, вызова метода getC у объекта t1	3
3		Инициализация переменных t2a, t2b, t2c типа int значениями вызова метода getA у объекта t2, вызова метода getB у объекта t2, вызова метода getC у объекта t2	4
4		Иницализация объекта t класса triangle с передачей t1a + t2a, t1b + t2b, t1c + t2c параметризированному конструктору	5
5		Инициализация переменных ta, tb, tc типа int значениями вызова метода getA у объекта t, вызова метода getB у объекта t, вызова метода getC у объекта t	6
6	ta + tb <= tc || ta + tc <= tb || tc + tb <= tc	Возврат t1	∅
		Возврат t	∅

3.6 Алгоритм функции main

Функционал: Главная функция программы.

Параметры: нет.

Возвращаемое значение: int - код ошибки.

Алгоритм функции представлен в таблице 6.

Таблица 6 – Алгоритм функции main

№	Предикат	Действия	№ перехода
1		Объявление перменных n, a, b, c типа int	2
2		Ввод значения n	3
3		Объявление вектора значений ts типа triangle	4
4		Инициализация i = 0 тиа int	5
5	i < n	Ввод значений a, b, c	6
		Объявление переменных t1, t2 типа int и op типа string	9
6		Инициализация объекта t класса triange с передачей параметризированному конструктору значений a, b, c	7
7		Добавление объекта t в конец вектора ts	8
8		i++	5
9	cin << t1 << op << t2	Ввод значений t1, op, t2	10
			11
10	op == "+"	ts[t1 - 1] = ts[t1 - 1] + ts[t2 - 1]	9
		ts[t1 - 1] = ts[t1 - 1] - ts[t2 - 1]	9
11		Инициализация объекта lst класса triange равному ts[t1 + 1]	12
12		Вывод значений полей a, b, c объекта lst	13
13		Возврат значения 0	∅

4 БЛОК-СХЕМЫ АЛГОРИТМОВ

Представим описание алгоритмов в графическом виде на рисунках 1-5.

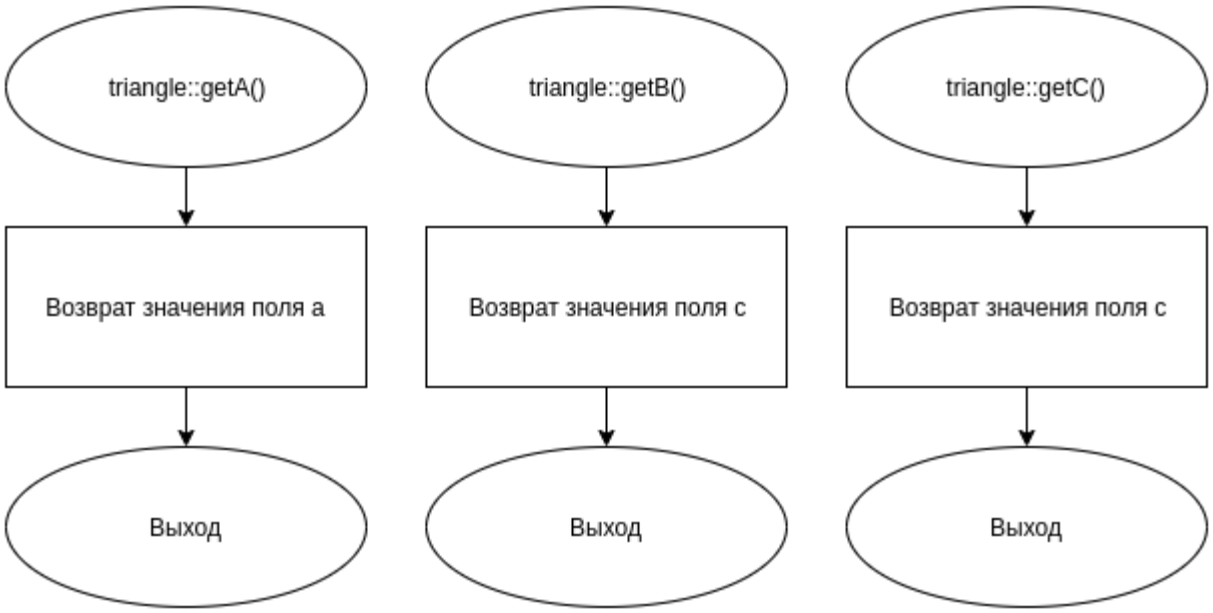

Рисунок 1 – Блок-схема алгоритма

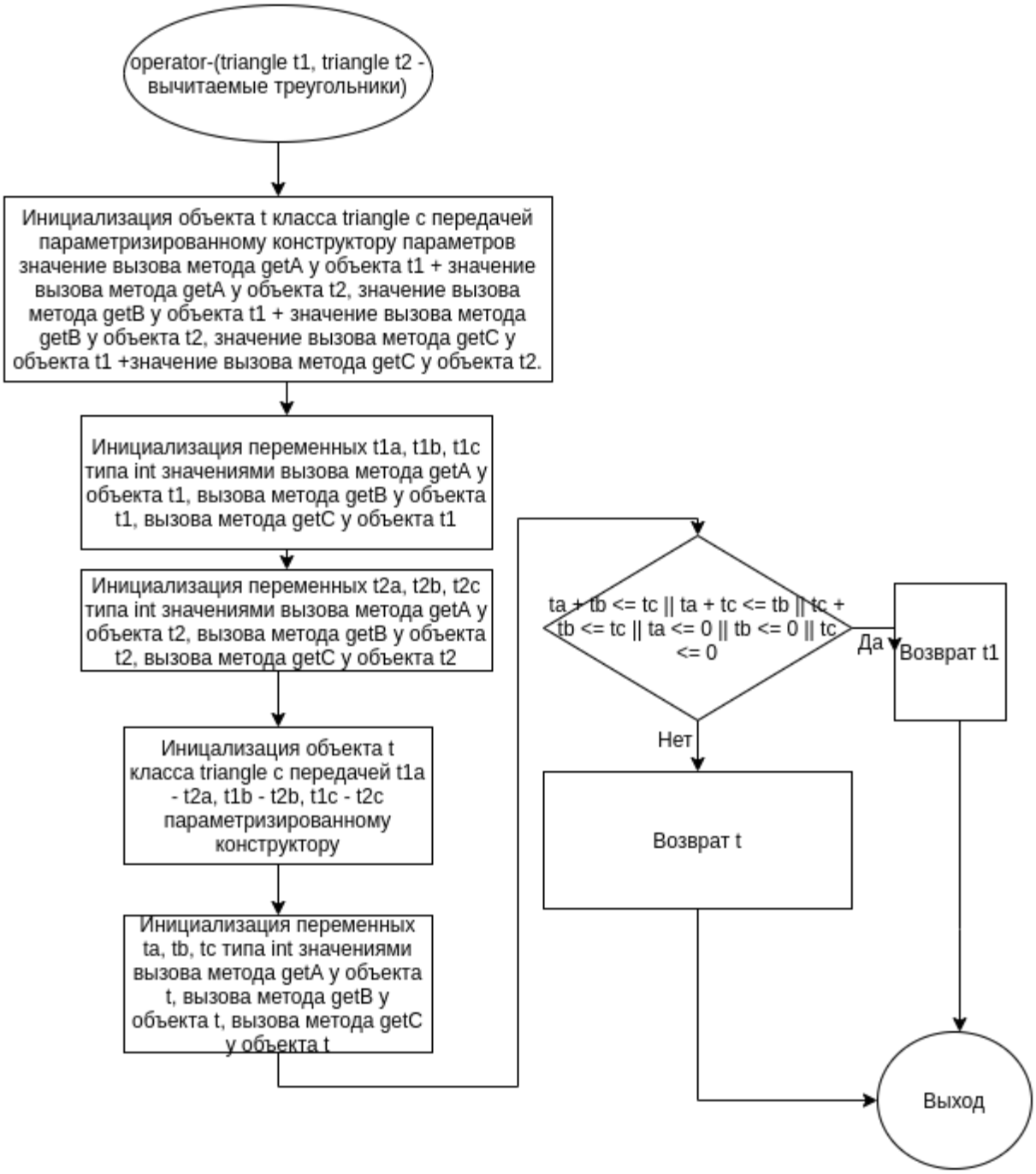

Рисунок 2 – Блок-схема алгоритма

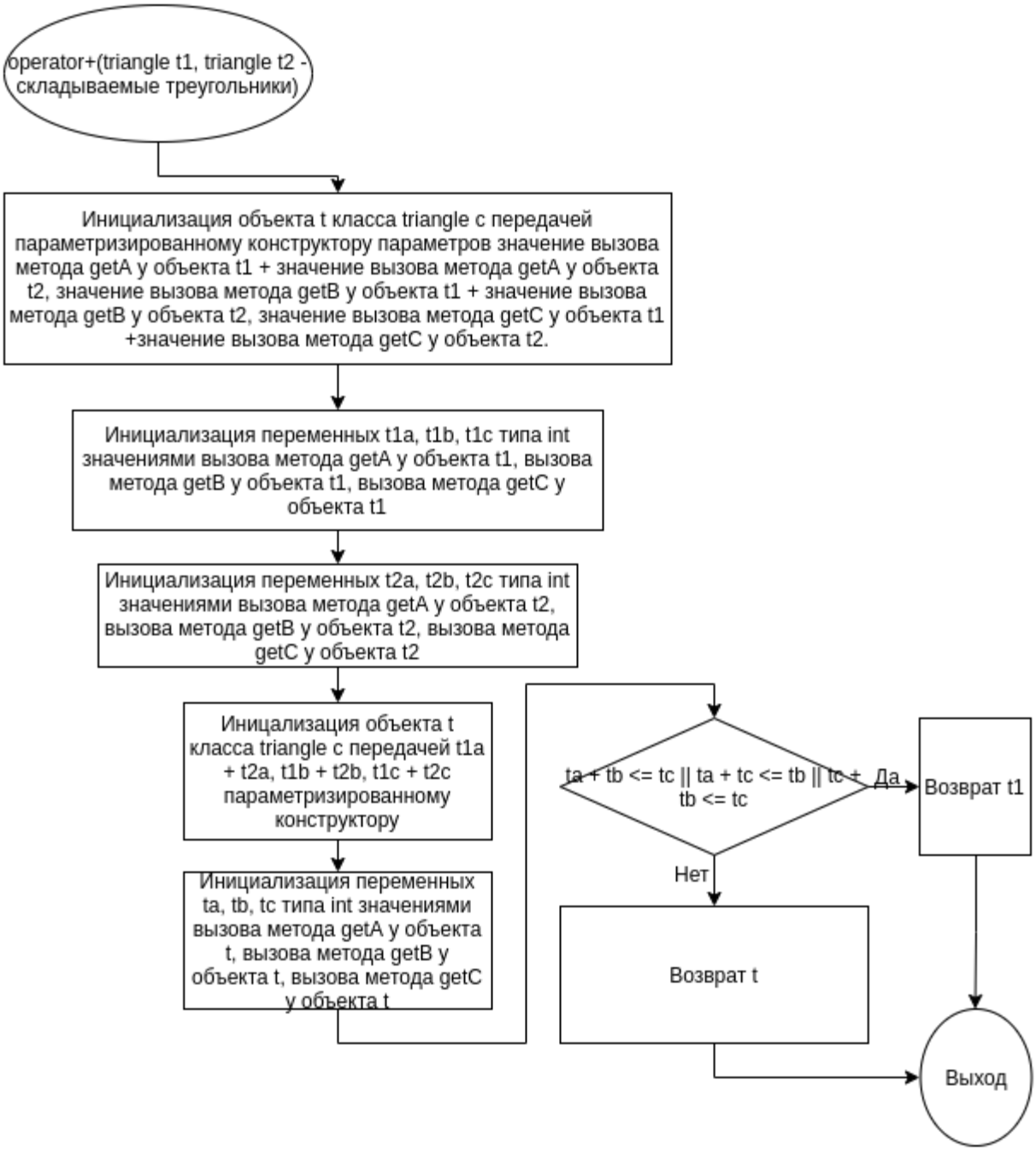

Рисунок 3 – Блок-схема алгоритма

Рисунок 4 – Блок-схема алгоритма

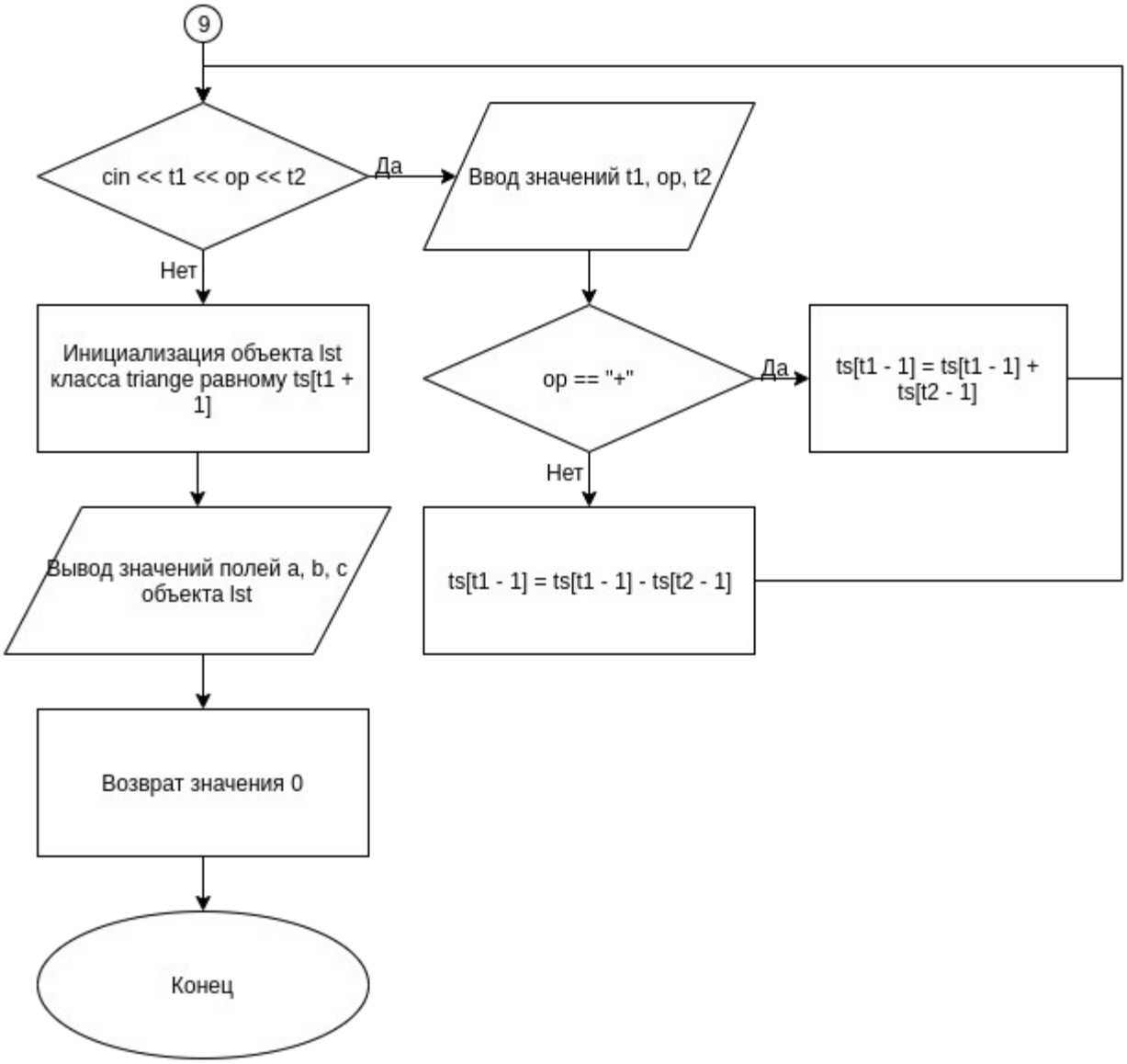

Рисунок 5 – Блок-схема алгоритма

5 КОД ПРОГРАММЫ

Программная реализация алгоритмов для решения задачи представлена ниже.

5.1 Файл main.cpp

Листинг 1 – main.cpp

```
#include <stdlib.h>
#include <stdio.h>
#include "triangle.h"
#include <vector>



triangle operator+(triangle t1, triangle t2) {
   int t1a = t1.getA(), t1b = t1.getB(), t1c = t1.getC();
   int t2a = t2.getA(), t2b = t2.getB(), t2c = t2.getC();

   triangle t(t1a + t2a, t1b + t2b, t1c + t2c);
   int ta = t.getA(), tb = t.getB(), tc = t.getC();

   if(ta +  tb <= tc || ta +  tc <= tb || tc +  tb <= tc)
      return t1;

   return t;
}

triangle operator-(triangle t1, triangle t2) {
   int t1a = t1.getA(), t1b = t1.getB(), t1c = t1.getC();
   int t2a = t2.getA(), t2b = t2.getB(), t2c = t2.getC();

   triangle t(t1a - t2a, t1b - t2b, t1c - t2c);
   int ta = t.getA(), tb = t.getB(), tc = t.getC();

   if(ta + tb <= tc || ta + tc <= tb || tc + tb <= tc || ta <= 0 || tb <= 0
|| tc <= 0)
      return t1;
   return t;
}

int main()
{

   int n, a, b, c;

   cin >> n;
```

```
   vector<triangle> ts;

   for(int i = 0; i < n; i++) {
      cin >> a >> b >> c;
      triangle t(a, b, c);
      ts.push_back(t);
   }
   int t1, t2;
   string op;
   while(cin >> t1 >> op >> t2) {
      if(op == "+")
         ts[t1 - 1] = ts[t1 - 1] + ts[t2 - 1];
      else
         ts[t1 - 1] = ts[t1 - 1] - ts[t2 - 1];
   }

   triangle lst = ts[t1 - 1];
   cout << "a = " <<  lst.getA() << "; b = " << lst.getB() << "; c = " <<
lst.getC() << ".";
   return(0);
}
```

5.2 Файл triangle.cpp

Листинг 2 – triangle.cpp

```
#include "triangle.h"



triangle::triangle(int a, int b, int c) {
   this->a = a;
   this->b = b;
   this->c = c;
}

float triangle::S() {
   float p = P() * 0.5;
   return sqrt(p * (p - a) * (p - b) * (p - c));
}

float triangle::P() {
   return a + b + c;
}

int triangle::getA() {
   return a;
}
```

```
int triangle::getB() {
   return b;
}

int triangle::getC() {
   return c;
}
```

5.3 Файл triangle.h

Листинг 3 – triangle.h

```
#ifndef __TRIANGLE__H
#define __TRIANGLE__H

#include <cmath>
#include <iostream>
using namespace std;

class triangle
{
private:
   int a, b, c;
public:
   triangle(int, int, int);
   float P();
   float S();

   int getA();
   int getB();
   int getC();
};


#endif
```

6 ТЕСТИРОВАНИЕ

Результат тестирования программы представлен в таблице 7.

Таблица 7 – Результат тестирования программы

Входные данные	Ожидаемые выходные данные	Фактические выходные данные
2 3 4 5 1 2 3 1 + 2	a = 4; b = 6; c = 8.	a = 4; b = 6; c = 8.
2 3 4 5 1 2 3 1 + 2 1 - 2	a = 3; b = 4; c = 5.	a = 3; b = 4; c = 5.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ГОСТ 19 Единая система программной документации.

2. Методическое пособие студента для выполнения практических заданий, контрольных и курсовых работ по дисциплине «Объектно-ориентированное программирование» [Электронный ресурс] – URL: https://mirea.aco-avrora.ru/student/files/methodichescoe_posobie_dlya_laboratornyh_rabot_3.pdf (дата обращения 05.05.2021).

3. Приложение к методическому пособию студента по выполнению заданий в рамках курса «Объектно-ориентированное программирование» [Электронный ресурс]. URL: https://mirea.aco-avrora.ru/student/files/Prilozheniye_k_methodichke.pdf (дата обращения 05.05.2021).

4. Шилдт Г. C++: базовый курс. 3-е изд. Пер. с англ.. — М.: Вильямс, 2019. — 624 с.

5. Видео лекции по курсу «Объектно-ориентированное программирование» [Электронный ресурс]. АСО «Аврора».

6. Антик М.И. Дискретная математика [Электронный ресурс]: Учебное пособие /Антик М.И., Казанцева Л.В. — М.: МИРЭА — Российский технологический университет, 2018 — 1 электрон. опт. диск (CD-ROM).