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:

- объект Triangle класса Triangle предназначен для массив для хранения сторон треугольника;
- функция main для главная функция программы;
- библиотека istream;
- объект cin стандартного потока ввода с клавиатуры;
- объект cout стандартного потока вывода на экран;
- условная конструкция if;
- цикл со счётчиком for;
- цикл с предусловием while.

Класс Triangle:

- свойства/поля:
    - поле первая сторона треугольника:
        - наименование — side_a;
        - тип — int;
        - модификатор доступа — public;
    - поле вторая сторона треугольника:
        - наименование — side_b;
        - тип — int;
        - модификатор доступа — public;
    - поле третья сторона треугольника:
        - наименование — side_c;
        - тип — int;
        - модификатор доступа — public;
- функционал:

- метод Triangle — конструктор по умолчанию;
- метод Triangle — параметризированный конструктор для установки значения сторон очередного треугольника;
- метод operator+ — перегрузка оператора сложения двух свойств объекта;
- метод operator- — перегрузка оператора разности двух свойств объекта.

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 конструктора класса Triangle

Функционал: параметризированный конструктор для установки значения сторон очередного треугольника.

Параметры: нет.

Алгоритм конструктора представлен в таблице 1.

Таблица 1 – Алгоритм конструктора класса Triangle

№	Предикат	Действия	№ перехода
1		Присвоение полю side_a значения переданного аргумента side_a	2
2		Присвоение полю side_b значения переданного аргумента side_b	3
3		Присвоение полю side_c значения переданного аргумента side_c	∅

3.2 Алгоритм метода operator+ класса Triangle

Функционал: перегрузка оператора сложения двух свойств объекта.

Параметры: ссылка на текущий объект класса Triangle.

Возвращаемое значение: Отсутствует.

Алгоритм метода представлен в таблице 2.

Таблица 2 – Алгоритм метода operator+ класса Triangle

№	Предикат	Действия	№ перехода
1	Треугольник существует?	Свойству side_a первого объекта присвоить сумму свойств side_a первого объекта и side_a второго	2

№	Предикат	Действия	№ перехода
		объекта	
			∅
2		Свойству side_b первого объекта присвоить сумму свойств side_b первого объекта и side_b второго объекта	3
3		Свойству side_c первого объекта присвоить сумму свойств side_c первого объекта и side_c второго объекта	∅

3.3 Алгоритм метода operator- класса Triangle

Функционал: перегрузка оператора разности двух свойств объекта.

Параметры: ссылка на текущий объект класса Triangle.

Возвращаемое значение: Отсутствует.

Алгоритм метода представлен в таблице 3.

Таблица 3 – Алгоритм метода operator- класса Triangle

№	Предикат	Действия	№ перехода
1	Треугольник существует?	Свойству side_a первого объекта присвоить разность свойств side_a первого объекта и side_a второго объекта	2
			∅
2		Свойству side_b первого объекта присвоить разность свойств side_b первого объекта и side_b второго объекта	3
3		Свойству side_c первого объекта присвоить разность свойств side_c первого объекта и side_c второго объекта	∅

3.4 Алгоритм функции main

Функционал: главная функция программы.

Параметры: нет.

Возвращаемое значение: целое, индикация корректности работы программы.

Алгоритм функции представлен в таблице 4.

Таблица 4 – Алгоритм функции main

№	Предикат	Действия	№ перехода
1		Объявление целочисленных переменных n, a, b и c	2
2		Ввод с клавиатуры значения n	3
3		Инициализация указателя array на объект класса Triangle, хранящего динамический массив размера n	4
4		Инициализация целочсиленной переменной счётчика i значением 0	5
5	Значение переменной i меньше значения переменной n	Ввод значений сторон a, b и c очередного треугольника	6
	Значение переменной i не меньше значения переменной n		8
6		Присвоение i-ому элементу массива объекта класса Triangle значений введённых сторон очередного треугольника, при передаче параметризированному конструктору введённых значений a, b и c	7
7		Инкремент переменной i	5
8		Объявление переменной строкового типа sign	9
9		Ввод значений переменных a (номер треугольника), sign (арифметическая операция (+	10

№	Предикат	Действия	№ перехода
		или -)) и b (номер треугольника)	
10	Ввод отсутствует (пустой)	Отстановка (break)	14
	Ввод не отсутствует (не пустой)		11
11	Значение переменной sign равняется "-"	Замена операции вычитания действием перегрузки оператора вычитания operator+	12
	Значение переменной sign не равняется "-"		12
12	Значение переменной sign равняется "+"	Замена операции сложения действием перегрузки оператора сложения operator+	13
	Значение переменной sign не равняется "+"		13
13		Присвоение переменной n значения переменной a	9
14		Вывод:" a = ",<<значение свойства side_a последней операции>>,"; b = ",<<значение свойства side_b последней операции>>,"; c = ",<<значение свойства side_c последней операции>>"."	∅

4 БЛОК-СХЕМЫ АЛГОРИТМОВ

Представим описание алгоритмов в графическом виде на рисунках 1-3.

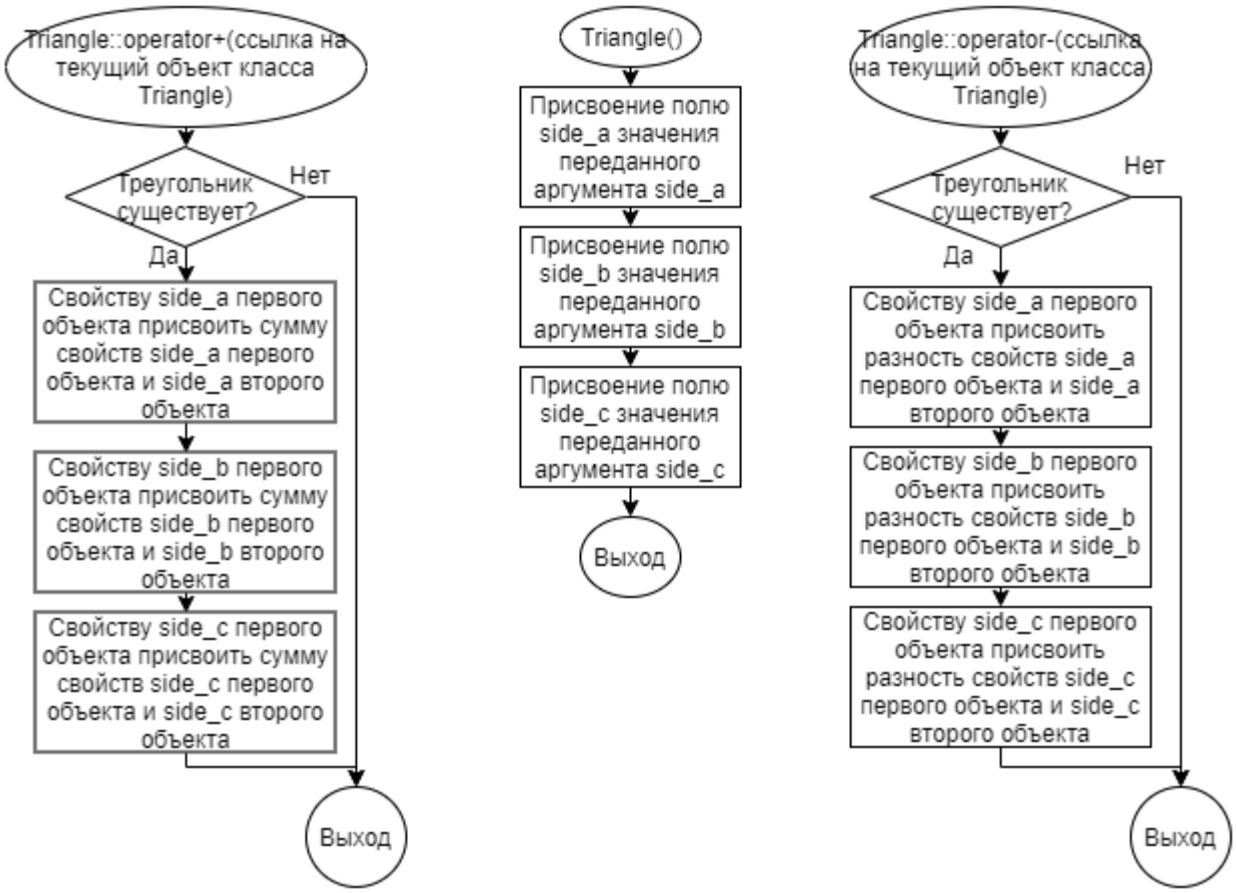

Рисунок 1 – Блок-схема алгоритма

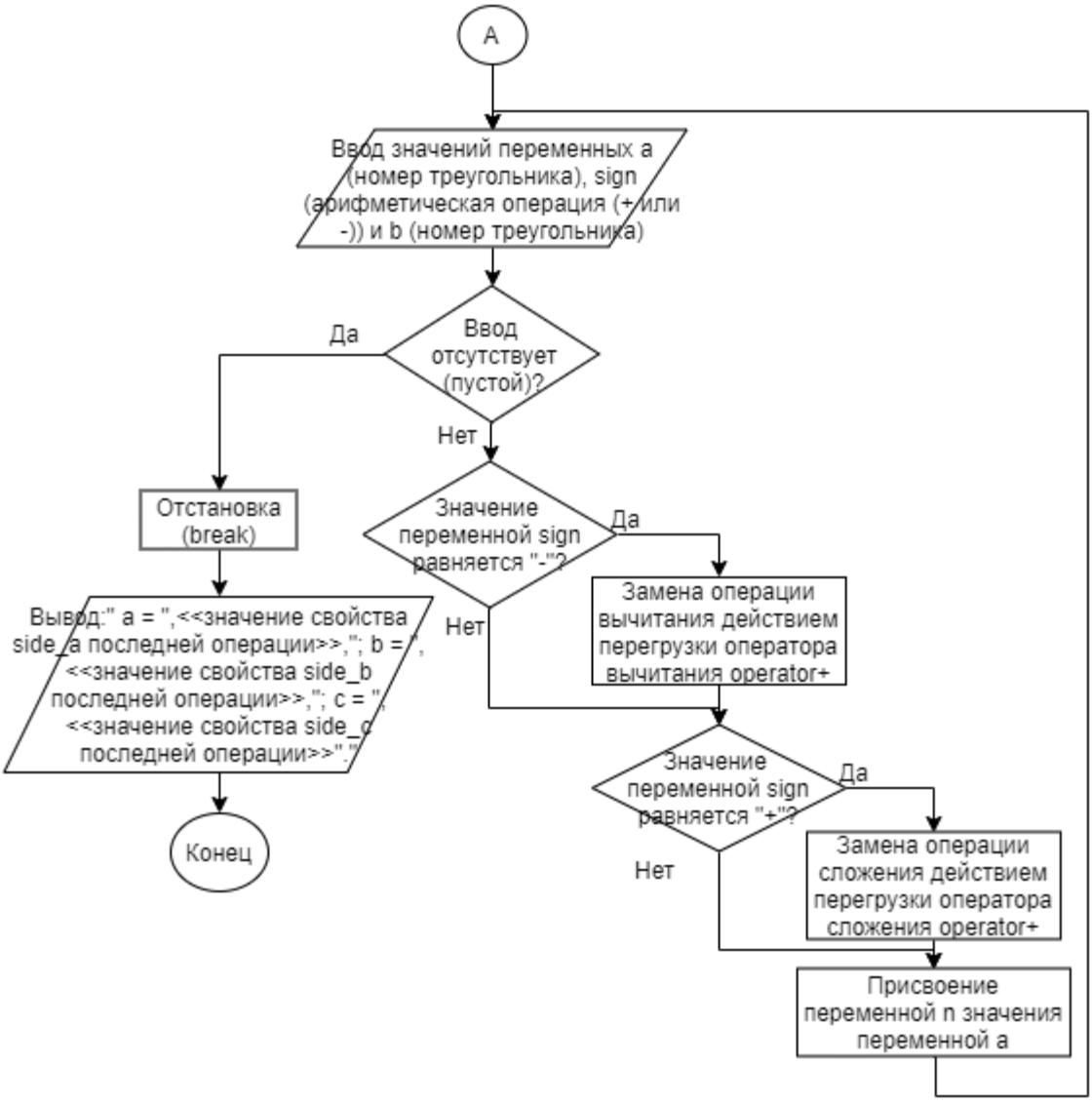

Рисунок 2 – Блок-схема алгоритма

Рисунок 3 – Блок-схема алгоритма

5 КОД ПРОГРАММЫ

Программная реализация алгоритмов для решения задачи представлена ниже.

5.1 Файл main.cpp

Листинг 1 – main.cpp

```
#include "Triangle.h"
#include <iostream>

using namespace std;

int main()
{
   int n, a, b, c;
   cin >> n;
   Triangle* array = new Triangle[n];
   for(int i = 0; i < n; i++)
   {
      cin >> a >> b >> c;
      array[i] = Triangle(a, b, c);
   }
   string sign;
   while (true)
   {
      cin >> a >> sign >> c;
      if (cin.fail()) {break;}
      if (sign == "-") {array[a-1] - array[c-1];}
      if (sign == "+") {array[a-1] + array[c-1];}
      n = a;
   }
   cout << "a = " << array[n - 1].side_a << "; b = " << array[n - 1].side_b
<< "; c = " << array[n - 1].side_c<< ".";
   return 0;
}
```

5.2 Файл Triangle.cpp

Листинг 2 – Triangle.cpp

```
#include "Triangle.h"
```

```
Triangle :: Triangle(int side_a, int side_b, int side_c)
{
   this -> side_a = side_a;
   this -> side_b = side_b;
   this -> side_c = side_c;
}

Triangle :: Triangle(){}

/*
float Triangle :: Area()
{
   float half_perimetr = (side_a + side_b + side_c)/2.0;
   float area = sqrt(half_perimetr*(half_perimetr - side_a)*(half_perimetr -
side_b)*(half_perimetr - side_c));
   return area;
}
int Triangle :: Perimetr()
{
   int perimetr = side_a + side_b + side_c;
   return perimetr;
}
*/

void Triangle::operator+(Triangle& Object)
{
   if ((Object.side_a + this -> side_a + Object.side_b + this -> side_b >
Object.side_c + this -> side_c) && (Object.side_a + this -> side_a +
Object.side_c + this -> side_c > Object.side_b + this -> side_b) &&
(Object.side_c + this -> side_c + Object.side_b + this -> side_b >
Object.side_a + this -> side_a))
   {
      this -> side_a += Object.side_a;
      this -> side_b += Object.side_b;
      this -> side_c += Object.side_c;
   }
}

void Triangle :: operator-(Triangle& Object)
{
   if(this -> side_a - Object.side_a > 0 && this -> side_b - Object.side_b >
0 && this -> side_c - Object.side_c > 0 && this -> side_a - Object.side_a +
this -> side_b - Object.side_b > this -> side_c - Object.side_c && this ->
side_a - Object.side_a + this -> side_c - Object.side_c > this -> side_b -
Object.side_b && this -> side_c - Object.side_c + this -> side_b -
Object.side_b > this -> side_a - Object.side_a)
   {
      this -> side_a = this -> side_a - Object.side_a;
      this -> side_b = this -> side_b - Object.side_b;
      this -> side_c = this -> side_c - Object.side_c;
   }
}
```

5.3 Файл Triangle.h

Листинг 3 – Triangle.h

```
#ifndef __TRIANGLE__H
#define __TRIANGLE__H
class Triangle
{
   public:

   /*2_1_2c
   Triangle(int side_a, int side_b, int side_c);
   float Area();
   int Perimetr();
   */

   Triangle(int side_a, int side_b, int side_c);
   Triangle();
   void operator+(Triangle& Object);
   void operator-(Triangle& Object);
   int side_a, side_b, side_c;
};


#endif
```

6 ТЕСТИРОВАНИЕ

Результат тестирования программы представлен в таблице 5.

Таблица 5 – Результат тестирования программы

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

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).