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 triangles класса Triangle предназначен для ;
- функция main для Основная функция;
- Объект стандартного потока ввода/вывода cin/cout;
- Условный оператор if;
- Оператор со счетчиком.

Класс Triangle:

- свойства/поля:
    - поле Первая сторона:
        - наименование — a;
        - тип — Целочисленное;
        - модификатор доступа — private;
    - поле Вторая сторона:
        - наименование — b;
        - тип — Целочисленное;
        - модификатор доступа — private;
    - поле Третья сторона:
        - наименование — c;
        - тип — Целочисленное;
        - модификатор доступа — private;
- функционал:
    - метод print — Вывод результата;
    - метод Triangle — Конструктор;
    - метод operator+ — Метод сложения;
    - метод operator - — Метод разности.

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 print класса Triangle

Функционал: Вывод результата.

Параметры: нет.

Возвращаемое значение: Отсутствует.

Алгоритм метода представлен в таблице 1.

Таблица 1 – Алгоритм метода print класса Triangle

№	Предикат	Действия	№ перехода
1		Вывод результата	∅

3.2 Алгоритм конструктора класса Triangle

Функционал: Конструктор.

Параметры: нет.

Алгоритм конструктора представлен в таблице 2.

Таблица 2 – Алгоритм конструктора класса Triangle

№	Предикат	Действия	№ перехода
1		Конструктор	∅

3.3 Алгоритм метода operator+ класса Triangle

Функционал: Метод сложения.

Параметры: нет.

Возвращаемое значение: Отсутствует.

Алгоритм метода представлен в таблице 3.

Таблица 3 – Алгоритм метода operator+ класса Triangle

№	Предикат	Действия	№ перехода
1		Выполнение операции сложения одного треугольника с другим	∅

3.4 Алгоритм метода operator - класса Triangle

Функционал: Метод разности.

Параметры: нет.

Возвращаемое значение: Отсутствует.

Алгоритм метода представлен в таблице 4.

Таблица 4 – Алгоритм метода operator - класса Triangle

№	Предикат	Действия	№ перехода
1		Выполнение операции разности одного треугольника с другим	∅

3.5 Алгоритм функции main

Функционал: Основная функция.

Параметры: нет.

Возвращаемое значение: Целочисленное.

Алгоритм функции представлен в таблице 5.

Таблица 5 – Алгоритм функции main

№	Предикат	Действия	№ перехода
1		Ввод количества треугольников n	2
2		В цикле для каждого треугольника вводятся исходные длины сторон.	3

№	Предикат	Действия	№ перехода
		Далее создается объект, в конструктор которого передаются значения длин сторон. Каждый объект треугольника получает свой номер от 1 до n.	
3		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.	4
4		После каждого ввода выполняется операция, результат присваивается первому аргументу (объекту треугольника).	5
5		Цикл завершается по завершению данных.	6
6		Выводится результат последней операции.	∅

4 БЛОК-СХЕМЫ АЛГОРИТМОВ

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

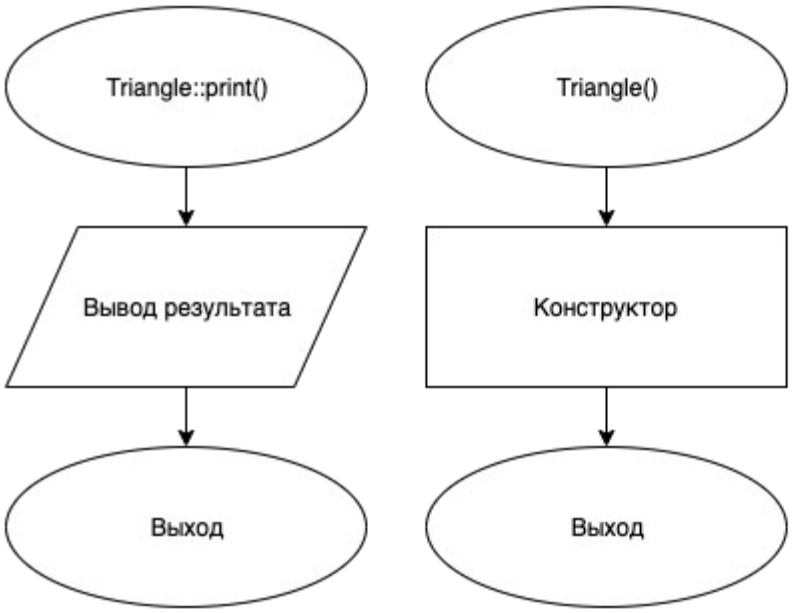

Рисунок 1 – Блок-схема алгоритма

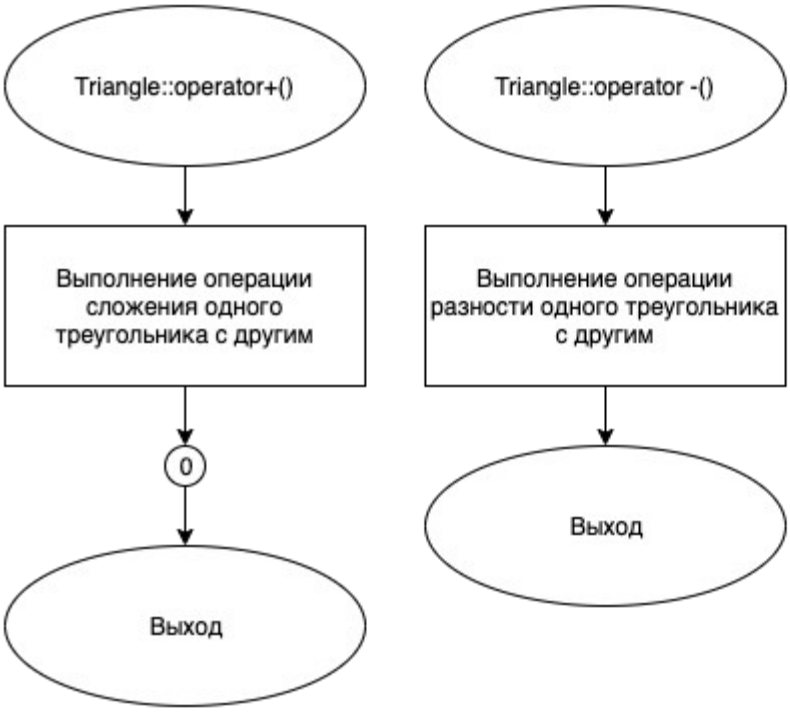

Рисунок 2 – Блок-схема алгоритма

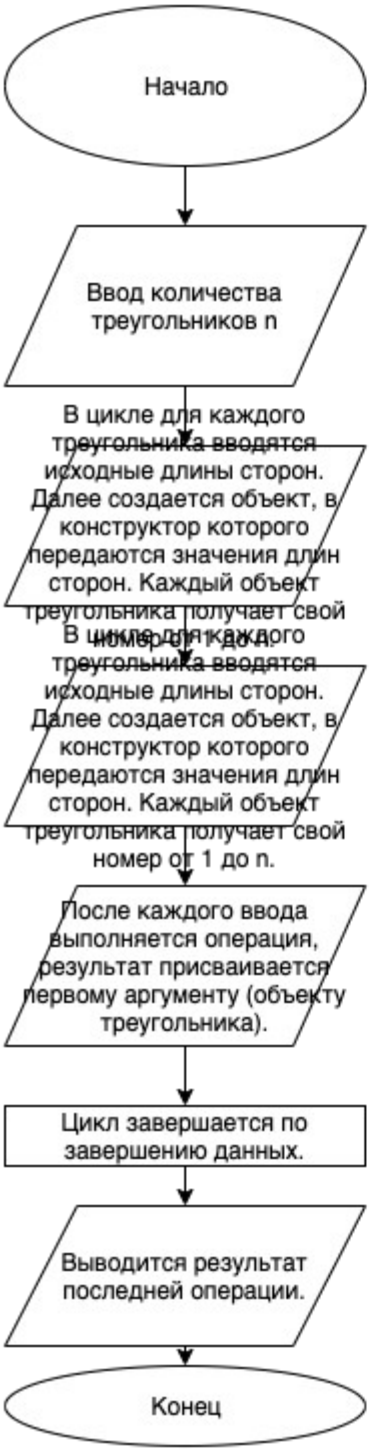

Рисунок 3 – Блок-схема алгоритма

5 КОД ПРОГРАММЫ

Программная реализация алгоритмов для решения задачи представлена ниже.

5.1 Файл main.cpp

Листинг 1 – main.cpp

```
#include <stdlib.h>
#include <stdio.h>
#include <iostream>
#include <vector>
#include "Triangle.h"
int main()
{
   char znak;
   int head, sub, amount = 0;
   vector<Triangle> triangles;
   while(amount <= 0)
   {
      cin >> amount;
   }
   for(; amount > 0; amount--)
   {
      int side[3] = {0, 0, 0};
      for(int i = 0; i < 3; i++)
      {
         while(side[i] <= 0)
         {
            cin >> side[i];
         }
      }
      triangles.push_back(Triangle(side[0], side[1], side[2]));
   }
   while(cin >> head)
   {
      cin >> znak >> sub;
      if(head > 0 && head <= triangles.size() && sub > 0 && sub <=
triangles.size())
      {
         if(znak == '+')
            triangles[head - 1] += triangles[sub - 1];
         else if(znak == '-')
            triangles[head - 1] -= triangles[sub - 1];
      }
   }
   if(head > 0 && head <= triangles.size())
```

```
   {
      cout  <<  "a  =  "<<  triangles.at(head  -  1).a  <<  ";  b  =  "  <<
triangles.at(head - 1).b << "; c = " << triangles.at(head - 1).c <<".";
   }
   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;
}

Triangle Triangle::operator+(const Triangle& other)
{

   if(other.a + other.b > other.c && other.a + other.c > other.b && other.b +
other.c > other.a)
   {
      a+= other.a;
      b+= other.b;
      c+= other.c;
   }
   return *this;
}

Triangle Triangle::operator+=(const Triangle& other)
{
   return operator+(other);
}

Triangle Triangle::operator-(const Triangle& other)
{
   if(a - other.a > 0 && b - other.b > 0 && c - other.c > 0)
   {
      if((a - other.a) + (b - other.b) > c - other.c && (a - other.a) + (c -
other.c) > b - other.b && (c - other.c) + (b - other.b) > a - other.a)
      {
         a-= other.a;
         b-= other.b;
         c-= other.c;
      }
   }
   return *this;
```

```
}

Triangle Triangle::operator-=(const Triangle& other)
{
   return operator-(other);
}
```

5.3 Файл Triangle.h

Листинг 3 – Triangle.h

```
#ifndef __TRIANGLE__H
#define __TRIANGLE__H
#include <iostream>
#include <cmath>

using namespace std;

class Triangle
{
public:
   int a,b,c;
   Triangle(int, int, int);
   Triangle operator+(const Triangle& other);
   Triangle operator-(const Triangle& other);
   Triangle operator+=(const Triangle& other);
   Triangle operator-=(const Triangle& other);
};

#endif
```

6 ТЕСТИРОВАНИЕ

Результат тестирования программы представлен в таблице 6.

Таблица 6 – Результат тестирования программы

Входные данные	Ожидаемые выходные данные	Фактические выходные данные
2 3 4 5 5 12 13 1+2	a = 8; b = 16; c = 18.	a = 8; b = 16; c = 18.
1 3 5 3 1+2	a = 3; b = 5; c = 3.	a = 3; b = 5; 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).