ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА»  
(РУТ (МИИТ))

Институт транспортной техники и систем управления

Кафедра «Управление и защита информации»

ОТЧЁТ  
О ЛАБОРАТОРНОЙ РАБОТЕ №2

По дисциплине «Языки программирования»

ВАРИАНТ 11

Выполнил: ст. гр. ТКИ-142

Луканкин Вячеслав Сергеевич

Проверил: к.т.н., доц. Васильева М. А.

(Проверил: к.т.н, доц. Балакина Е. П.)

Москва 2023

Оглавление

**Задание 2.11**

1.1 ФОРМУЛИРОВКА ЗАДАНИЯ 1

1.2 БЛОК-СХЕМА АЛГОРИТМА2-5

1.3 ТЕКСТ ПРОГРАММЫ НА ЯЗЫКЕ C6-9

1.4 РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ ПРОГРАММЫ10

1.5 ВЫПОЛНЕНИЕ ТЕСТОВЫХ ПРИМЕРОВ11

1.6 ОТМЕТКА О ВЫПОЛНЕНИИ ЗАДАНИЯ В ВЕБ-ХОСТИНГЕ СИСТЕМЫ КОНТРОЛЯ ВЕРСИЙ12

**Задание 2.2 13**

1.1 ФОРМУЛИРОВКА ЗАДАНИЯ 13

1.2 БЛОК-СХЕМА АЛГОРИТМА14-16

1.3 ТЕКСТ ПРОГРАММЫ НА ЯЗЫКЕ C17-18

1.4 РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ ПРОГРАММЫ19

1.5 ВЫПОЛНЕНИЕ ТЕСТОВЫХ ПРИМЕРОВ20

1.6 ОТМЕТКА О ВЫПОЛНЕНИИ ЗАДАНИЯ В ВЕБ-ХОСТИНГЕ СИСТЕМЫ КОНТРОЛЯ ВЕРСИЙ21

**Задание 2.3 22**

1.1 ФОРМУЛИРОВКА ЗАДАНИЯ 22

1.2 БЛОК-СХЕМА АЛГОРИТМА23-25

1.3 ТЕКСТ ПРОГРАММЫ НА ЯЗЫКЕ C26-29

1.4 РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ ПРОГРАММЫ30

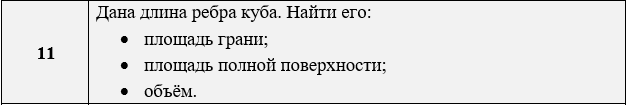
1.5 ВЫПОЛНЕНИЕ ТЕСТОВЫХ ПРИМЕРОВ31

1.6 ОТМЕТКА О ВЫПОЛНЕНИИ ЗАДАНИЯ В ВЕБ-ХОСТИНГЕ СИСТЕМЫ КОНТРОЛЯ ВЕРСИЙ32

1. зАДАНИЕ 2.1
   1. ФОРМУЛИРОВКА ЗАДАНИЯ

Создать консольное приложение для решения задачи, представленной в таблице (Таблица 1), с использованием перечислимого типа. Выбор формулы вычисления зависит от пользователя. Данные для решения задачи так же вводит пользователь. Ввод необходимо проверять на правильность. Все результаты вывести на экран. Дополнить свой отчёт блок-схемой алгоритма.

Таблица  – Исходные данные



* 1. Блок-схема алгоритма

Блок-схема основного алгоритма представлена ниже (Рисунок 1, Рисунок 2, Рисунок 3). Блок-схемы вспомогательных функций представлены ниже (Рисунок 4, Рисунок 5, Рисунок 6, Рисунок 7)

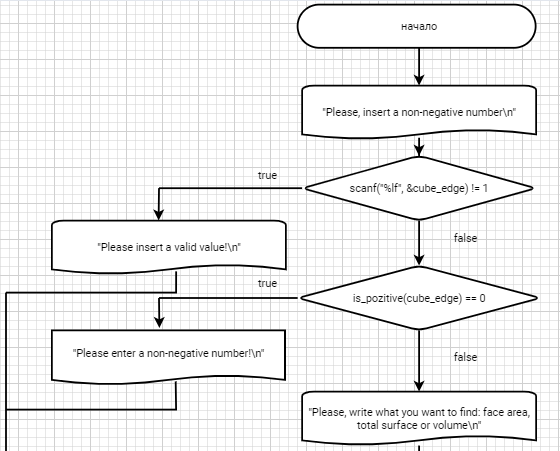


Рисунок  ­ Блок-схема основного алгоритма часть 1

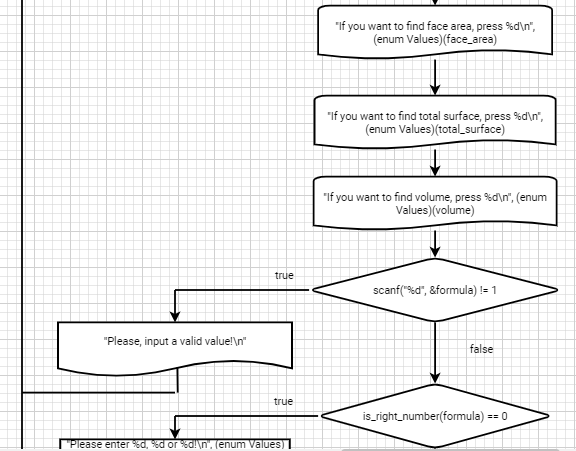


Рисунок  – Блок-схема основного алгоритма часть 2

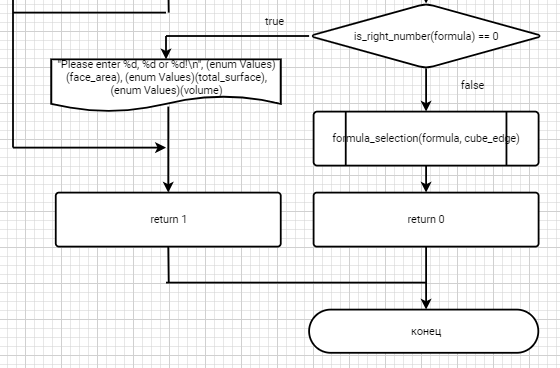


Рисунок  – Блок-схема основного алгоритма часть 3

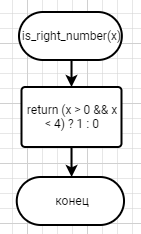


Рисунок  – Блок-схема функции is\_right\_number(x)

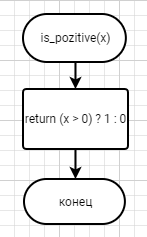


Рисунок  – Блок-схема функции is\_pozitive(x)

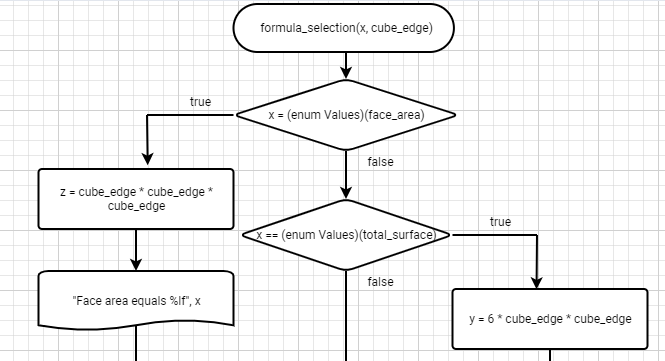


Рисунок 6 – Блок-схема функции formula\_selection(x, cube\_edge) часть 1.

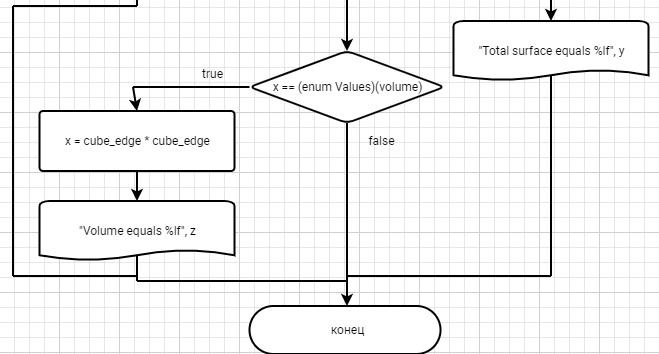


Рисунок 7 – Блок-схема функции formula\_selection(x, cube\_edge) часть 2.

* 1. Текст программы на языке C

#include <stdio.h>

#include <math.h>

#include <stdlib.h>

#include <stdbool.h>

/\*\*

\*@brief Функция проверяет, положительное ли число?

\*@param x проверяемое число

\*@return 1 - число положительное и отличное от нуля, 0 - число отрицательное или нуль

\*/

int is\_positive(const double x);

/\*\*

\*@brief Функция вычисляет нужную пользователю искомую величину

\*@param x порядковый номер формулы

\*/

double formula\_selection(int x, double cube\_edge);

/\*\*

\*@brief Функция сверяет введенное значение из переменной formula

\*@param x значение переменной formula

\*@return 1 - введенное число принадлежит диапазону [1;3]

\*/

int is\_right\_number(const int x);

enum Values

{

face\_area = 1,

total\_surface = 2,

volume = 3,

};

/\*\*

\*@brief Точка входа в программу

\*@return 0 программа работает правильно

\*@return 1 в программе есть ошибка

\*/

int main(){

double cube\_edge;

int formula;

puts("Please, insert a non-negative number\n");

if(scanf("%lf", &cube\_edge) != 1){

puts("Please insert a valid value\n");

return 1;}

if (is\_positive(cube\_edge) == 0){

puts("Please, enter a non-negative number!\n");

return 1;

}

puts("Please, write what you want to find: face area, total surface area or volume\n");

printf("If you want to find face area, press %d\n", (enum Values)(face\_area));

printf("If you want to find total surface, press %d\n", (enum Values)(total\_surface));

printf("If you want to find volume, press %d\n", (enum Values)(volume));

if(scanf("%d", &formula) != 1){

puts("Please input a valid value!\n");

return 1;}

if(is\_right\_number(formula) == 0){

printf("Please, enter %d, %d or %d!\n", (enum Values)(face\_area), (enum Values)(total\_surface), (enum Values)(volume));

return 1;

}

else{

formula\_selection(formula, cube\_edge);

}

return 0;

}

int is\_right\_number(const int x){

return (x > 0 && x < 4) ? 1 : 0;

}

int is\_positive(const double x){ // Зачем мне тут использовать bool?

return (x > 0) ? 1 : 0;

}

double formula\_selection(int x, double cube\_edge){

if(x == (enum Values)(face\_area)){

double x = cube\_edge\*cube\_edge;

printf("Face area equals: %lf", x);

}

else if(x == (enum Values)(total\_surface)){

double y = 6 \* cube\_edge \* cube\_edge;

printf("Total surface equals: %lf", y);

}

else if(x == (enum Values)(volume)){

double z = cube\_edge\*cube\_edge\*cube\_edge;

printf("Volume equals: %lf", z);

}

}

* 1. Результаты выполнения программы

Результаты выполнения программы представлены ниже (Рисунок 8).

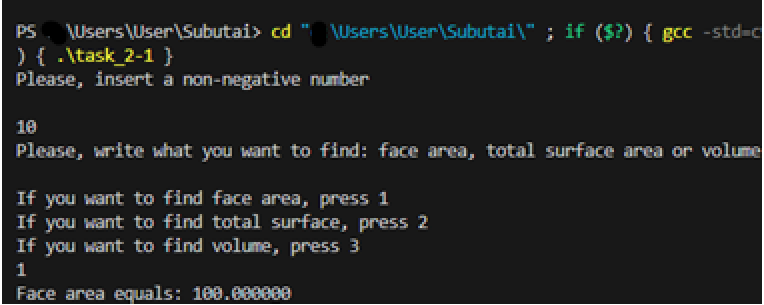


Рисунок  – Результаты выполнения программы

* 1. Выполнение тестовых примеров

В программе MS Excel выполнены тестовые примеры. Результаты их выполнения представлены ниже (Рисунок 9, Рисунок 10, Рисунок 11).

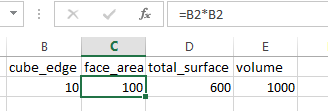


Рисунок  – Результат расчета переменной face\_area

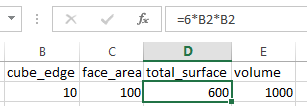


Рисунок  – Результат расчета переменной total\_surface

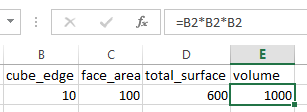
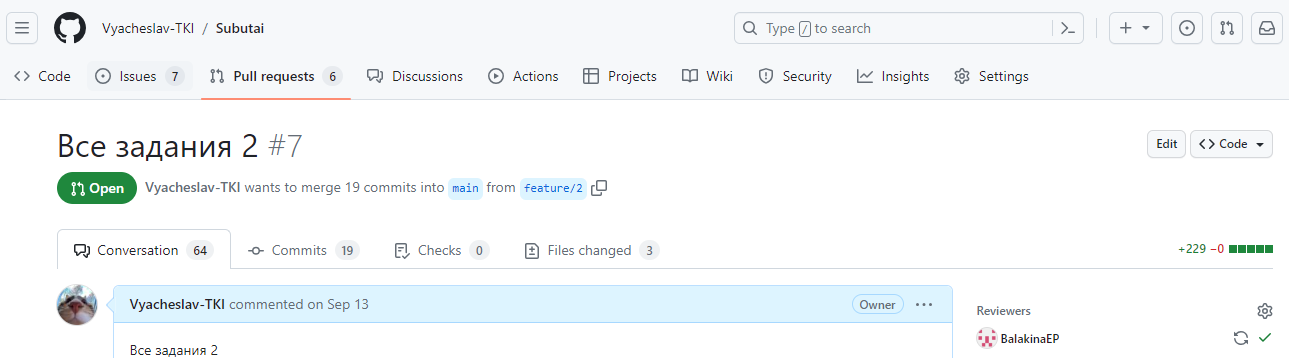


Рисунок  – Результат расчета переменной volume

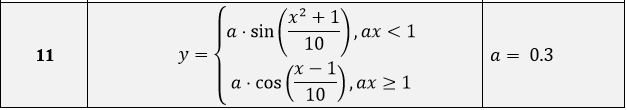
* 1. Отметка о выполнении задания в веб-хостинге системы контроля версий



1. зАДАНИЕ 2.2
   1. Формулировка задания

Создать консольное приложение с организацией разветвлённого процесса для решения заданной в таблице (Таблица 1) функции. Данные для решения вводит пользователь. Ввод необходимо проверять на правильность. Все результаты вывести на экран. Дополнить свой отчёт блок-схемой алгоритма

Таблица  – Исходные данные



2.2 Блок-схема алгоритма

Блок-схема основного алгоритма представлена ниже (Рисунок 12, Рисунок 13). Блок-схемы функций расчета представлены ниже (Рисунок 14, Рисунок 15).

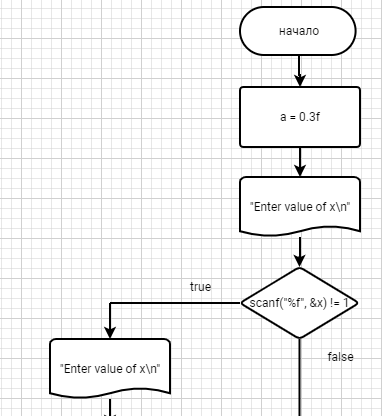


Рисунок  ­ Блок-схема основного алгоритма часть 1

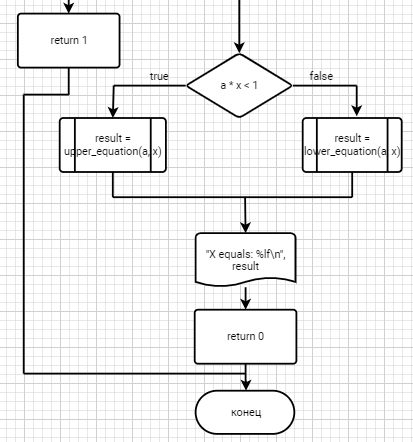


Рисунок  – Блок-схема основного алгоритма часть 2

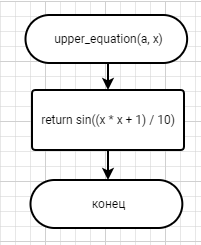


Рисунок 14 – Блок-схема функции upper\_equation(a, x)

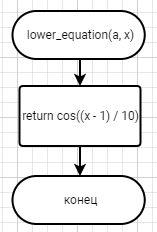


Рисунок 15 – Блок-схема функции lower\_equation(a, x)

* 1. Текст программы на языке C

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

/\*\*

\* @brief Функция вычисляет первое(верхнее уравнение)

\* @param a Заданная константа

\* @param x Заданная пользователем переменная

\*/

double upper\_equation(double a, double x);

/\*\*

\* @brief Функция вычисляет второе(нижнее) уравнение

\* @param a Заданная константа

\* @param x Заданная пользователем переменная

\*/

double lower\_equation(double a, double x);

/\*\*

\*@brief Точка входа в программу

\*@return 0 Программа исправна

\*@return 1 Программа неисправна

\*/

int main(){

double a = 0.3f;

double x;

double y;

double result;

puts("Enter value of x\n");

if(scanf("%f", &x) != 1){

puts("Please insert a valid value\n");

return 1;}

if(a\*x < 1){

result = upper\_equation(a, x);

}

else{

result = lower\_equation(a, x);

}

printf("X equals: %lf", result);

return 0;

}

double upper\_equation(double a, double x){

return a \* sin((x\*x+1)/10);

}

double lower\_equation(double a, double x){

return a \* cos((x-1)/10);

}

* 1. Результаты выполнения программы

Результаты выполнения программы представлены ниже (Рисунок 17).

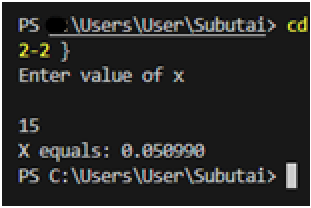


Рисунок  – Результаты выполнения программы

* 1. Выполнение тестовых примеров

В программе MS Excel выполнены тестовые примеры. Результаты их выполнения представлены ниже (Рисунок 18).

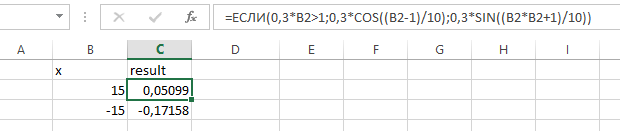
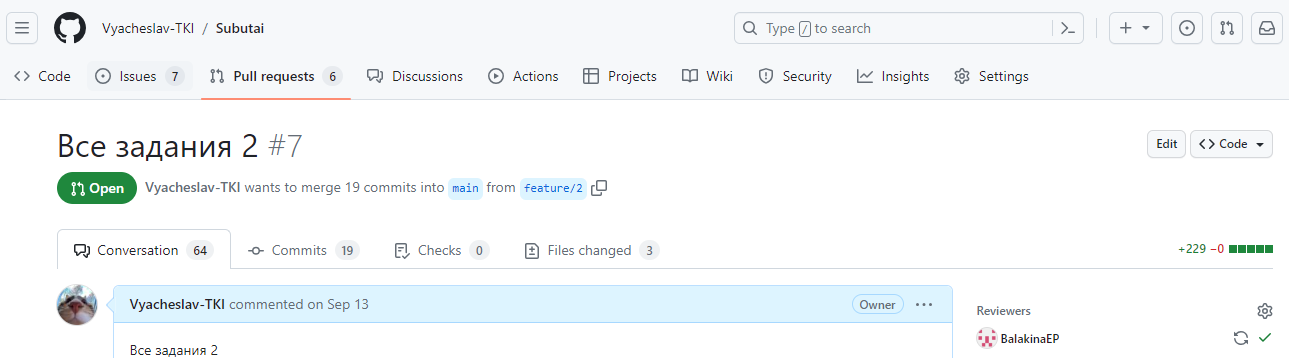


Рисунок  – Результат расчета переменной x

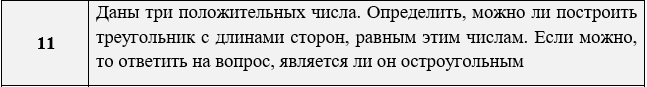
* 1. Отметка о выполнении задания в веб-хостинге системы контроля версий



1. зАДАНИЕ 2.3
   1. Формулировка задания

Создать консольное приложение с организацией разветвлённого процесса для решения задачи из таблицы (Таблица 1). Данные для решения вводит пользователь. Ввод необходимо проверять на правильность. Все результаты вывести на экран. Дополнить свой отчёт блок-схемой алгоритма.

Таблица  – Исходные данные



* 1. Блок-схема алгоритма

Блок-схема основного алгоритма представлена ниже (Рисунок 18, Рисунок 19). Блок-схема вспомогательных функций представлена ниже (Рисунок 20, Рисунок 21, Рисунок 22, Рисунок 23).

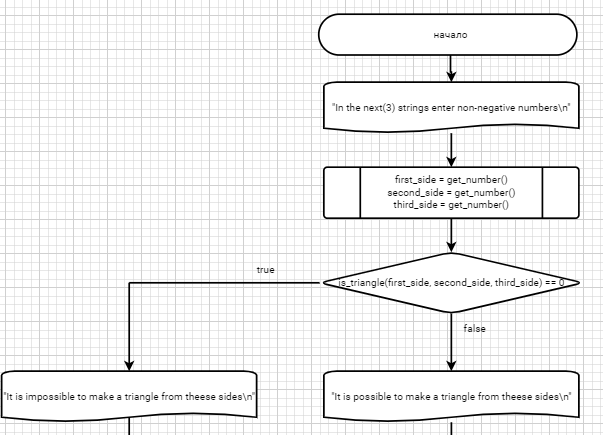


Рисунок  ­ Блок-схема основного алгоритма часть 1

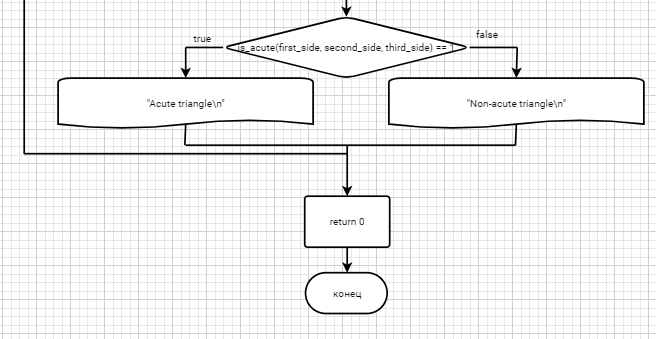


Рисунок  – Блок-схема основного алгоритма часть 2

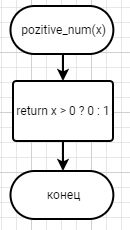


Рисунок  – Блок-схема функции positive\_num(x)

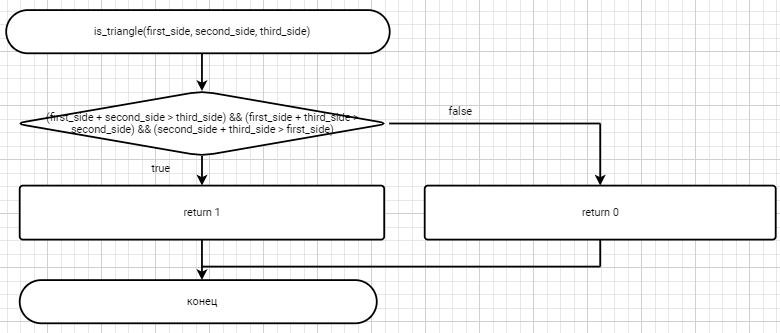


Рисунок 21 – Блок-схема is\_triangle(first\_side, second\_side, third\_side)

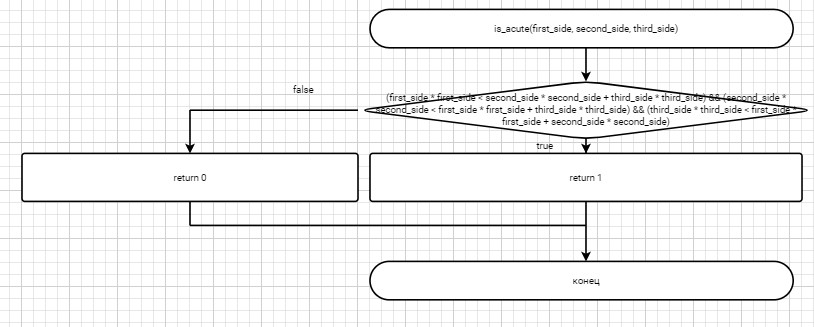


Рисунок 22 – Блок-схема функции is\_acute(first\_side, second\_side, third\_side)

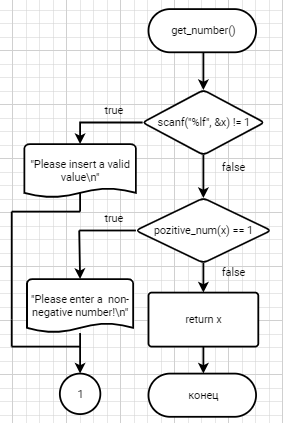


Рисунок  – Блок-схема функции get\_number()

* 1. Текст программы на языке C

#include <stdio.h>

#include <math.h>

#include <stdlib.h>

/\*\*

\*@brief Функция проверяет положительность числа

\*@param x проверяемое число

\*@return 1 - число отрицательное или равное нулю, 0 - число положительное

\*/

int pozitive\_num(double x);

/\*\*

\*@brief Функция проверяет принадлежность задданного значения к положительным числам

\*@return x введенное правильное значение

\*/

double get\_number();

/\*\*

\*@brief Функция проверяет, можно ли из этих сторон построить треугольник

\*@param first\_side длина отрезка

\*@param second\_side длина отрезка

\*@param third\_side длина отрезка

\*@return 1 можно построить треугольник

\*@return 0 нельзя построить треугольник

\*/

int is\_triangle(double first\_side, double second\_side, double third\_side);

/\*\*

\*@brief Функция проверяет треугольник на остроугольность по его сторонам

\*@param first\_side длина стороны

\*@param second\_side длина стороны

\*@param third\_side длина стороны

\*@return 1 Треугольник остроугольный

\*@return 0 Треугольник не остроугольный

\*/

int is\_acute(double first\_side, double second\_side, double third\_side);

/\*\*

\*@brief Точка входа в программу

\*/

int main(){

puts("In the next(3) strings enter non-negative numbers\n");

double first\_side = get\_number(), second\_side = get\_number(), third\_side = get\_number();

if (is\_triangle(first\_side, second\_side, third\_side) == 0){

puts("It is impossible to make a triangle from these sides\n");

}

else{

puts("It is possible to make a triangle from these sides\n");

if (is\_acute(first\_side, second\_side, third\_side) == 1){

puts("Acute triangle\n");

}

else{

puts("Non-acute triangle\n");

}

}

return 0;

}

int pozitive\_num(double x){

return x > 0 ? 0 : 1;

}

double get\_number(){

double x;

if (scanf("%lf", &x) != 1){

puts("Please insert a valid value\n");

abort();

}

if (pozitive\_num(x) == 1){

puts("Please, enter non-negative number\n");

abort();

}

return x;

}

int is\_triangle(double first\_side, double second\_side, double third\_side){

if ((first\_side + second\_side > third\_side) && (first\_side + third\_side > second\_side) && (second\_side + third\_side > first\_side)){

return 1;

}

return 0;

}

int is\_acute(double first\_side, double second\_side, double third\_side){

if ((first\_side \* first\_side < second\_side \* second\_side + third\_side \* third\_side) && (second\_side \* second\_side < first\_side \* first\_side+third\_side \* third\_side) && (third\_side\*third\_side < first\_side \* first\_side + second\_side \* second\_side)){

return 1;

}

return 0;

}

* 1. Результаты выполнения программы

Результаты выполнения программы представлены ниже (Рисунок 24).

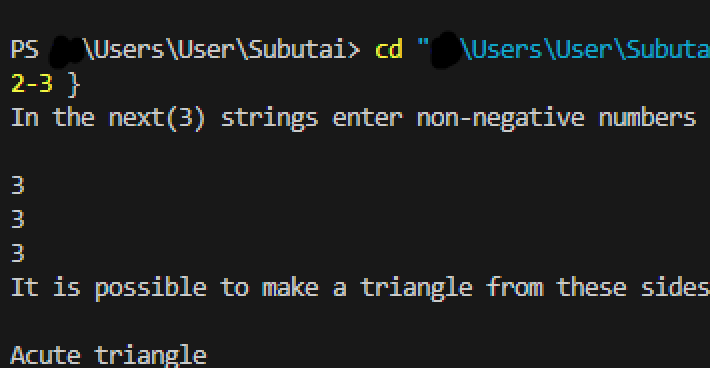


Рисунок  – Результат выполнения программы

* 1. Выполнение тестовых примеров

В программе MS Excel выполнены тестовые примеры. Результаты их выполнения представлены ниже (Рисунок 25, Рисунок 26).

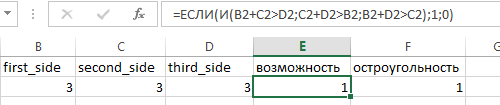


Рисунок  – Результат проверки возможности построения треугольника

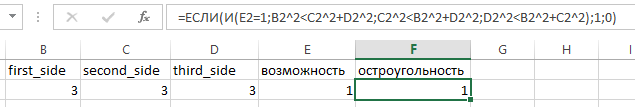


Рисунок  – Блок-схема проверки треугольника на остроугольность

* 1. Отметка о выполнении задания в веб-хостинге системы контроля версий

