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 ПРАКТИЧЕСКОЙ РАБОТЕ №2

По дисциплине «Введение в языки программирования»

Выполнил: ст. гр. ТКИ – 111

Насонова А.Н.

Проверил: к.т.н., доц.

Васильева М.А.

Москва 2021

**Оглавление**

[**Задание 2-1** 3](#_Toc93680093)

[**Формулировка задания** 3](#_Toc93680094)

[**Блок-схема алгоритма** 4](#_Toc93680095)

[**Программа на языкe C++** 5](#_Toc93680096)

[**Решение тестовых примеров на C++** 8](#_Toc93680097)

[**Решение тестовых примеров Excel** 9](#_Toc93680098)

[**Поясняющие формулы** 10](#_Toc93680099)

[**Задание 2-2** 11](#_Toc93680100)

[**Формулировка задания** 11](#_Toc93680101)

[**Блок-схема алгоритма** 12](#_Toc93680102)

[**Программа на языке С++** 13](#_Toc93680103)

[**Решение тестовых примеров C++** 15](#_Toc93680104)

[**Решение тестовых примеров Excel** 16](#_Toc93680105)

[**Задание 2-3** 17](#_Toc93680106)

[**Формулировка задания** 17](#_Toc93680107)

[**Блок-схема алгоритма** 18](#_Toc93680108)

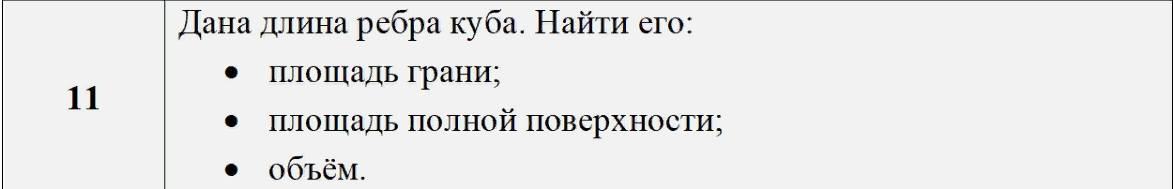
[**Программа на языке С++** 19](#_Toc93680109)

[**Решение тестовых примеров C++** 20](#_Toc93680110)

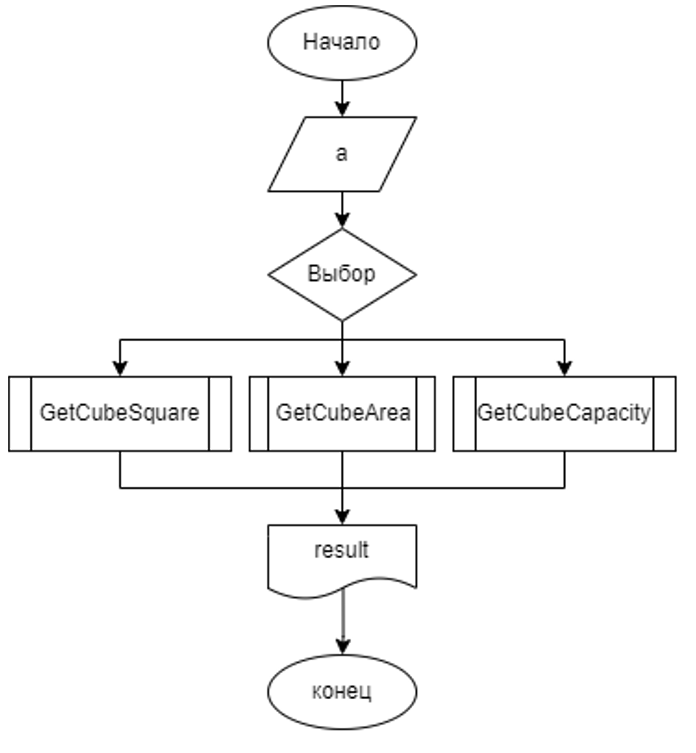
# **Задание 2-1**

# **Формулировка задания**

Создать консольное приложение для решения задачи, представленной в таблице, 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.

****

# **Блок-схема алгоритма**

****

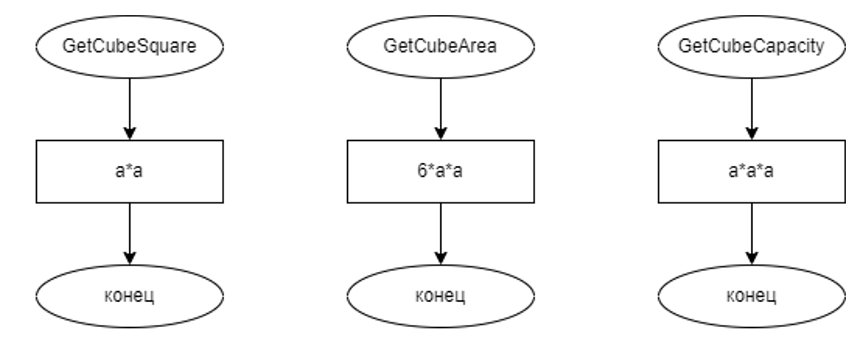
****

Рисунок 1 – Блок-схема алгоритма

# **Программа на языкe C++**

#include <iostream>

#include <string>

using namespace std;

/\*\*

\* \brief Перечисляемый тип вычеслений, для которых считаем площадь и объем.

\* \*/

enum class Cube

{

/\*\*

\* \brief Неправильно сформированный выбор.

\*/

none,

/\*\*

\* \brief Выбранный расчет -- площадь грани.

\*/

square,

/\*\*

\* \brief Выбранный расчет -- площадь полной поверхности.

\*/

area,

/\*\*

\* \brief Выбранный расчет -- объем.

\*/

capacity,

};

/\*\*

\* \brief Функция расчета площади грани куба.

\* \param a сторона куба.

\* \return Площадь гарни куба.

\*/

double GetCubeSquare(const double a);

/\*\*

\* \brief Функция расчета площади полной поверхности куба.

\* \param a сторона куба.

\* \return Площадь полной поверхности куба.

\*/

double GetCubeArea(const double a);

/\*\*

\* \brief Функция расчета объема куба.

\* \param a сторона куба.

\* \return Объем куба.

\*/

double GetCubeCapacity(const double a);

/\*\*

\* \brief Ввод стороны куба.

\* \param message Разъясняющая надпись.

\* \param out Произвольный поток вывода.

\* \param in Произвольный поток ввода.

\* \return Сторону куба.

\*/

double ReadSide(const string& message = "", ostream& out = cout, istream& in = cin);

/\*\*

\* \brief

\* \param message Сообщение для пользователя.

\* \param out Произвольный поток вывода.

\* \param in Произвольный поток ввода.

\* \return Выбор пользователя.

\*/

Cube ReadUserChoice(const string& message = "", ostream& out = cout, istream& in = cin);

/\*\*

\* \brief Точка вход в программу

\* \return Код ошибки, если 0 - успешное выполнение

\*/

int main()

{

setlocale(LC\_ALL, "Russian");

const auto message = "Выберите, что посчитать: "

+ to\_string(static\_cast<int>(Cube::square)) + " - площадь грани,"

+ to\_string(static\_cast<int>(Cube::area)) + " - площадь полной поверхности,"

+ to\_string(static\_cast<int>(Cube::capacity)) + " - объем";

const auto cube = ReadUserChoice(message);

switch (cube)

{

case Cube::square:

{

const auto side = ReadSide("Введите сторону куба = ");

const auto squareCube = GetCubeSquare(side);

cout << "\nПлощадь грани куба " << squareCube << endl;

break;

}

case Cube::area:

{

const auto side = ReadSide("Введите сторону куба = ");

const auto areaCube = GetCubeArea(side);

cout << "\nПлощадь полной поверхности " << areaCube << endl;

break;

}

case Cube::capacity:

{

const auto side = ReadSide("Введите сторону куба = ");

const auto capacityCube = GetCubeCapacity(side);

cout << "\nОбъем " << capacityCube << endl;

break;

}

default:

cout << "ошибка!";

}

system("pause");

return 0;

}

double GetCubeSquare(const double a)

{

return a \* a;

}

double GetCubeArea(const double a)

{

return 6 \* a \* a;

}

double GetCubeCapacity(const double a)

{

return a \* a \* a;

}

double ReadSide(const string& message, ostream& out, istream& in) {

out << message;

double side;

in >> side;

return side;

}

Cube ReadUserChoice(const string& message, ostream& out, istream& in)

{

out << message;

int userInput;

in >> userInput;

return static\_cast<Cube>(userInput);

}

# **Решение тестовых примеров на C++**

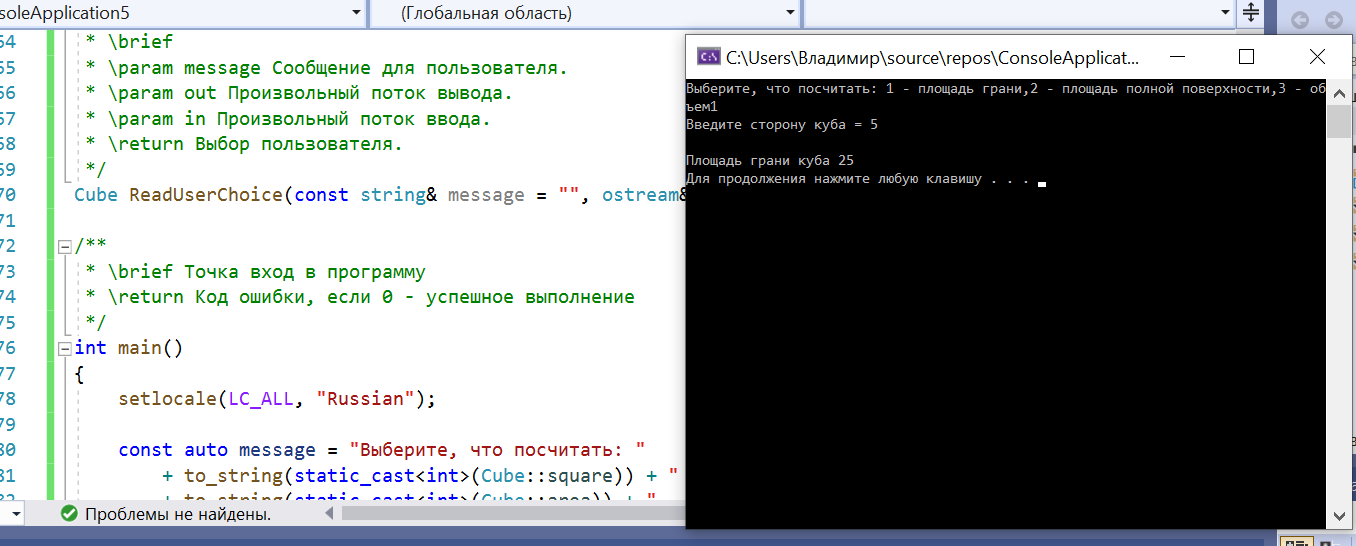
****

Рисунок 2 – Решение тестового примера на C++

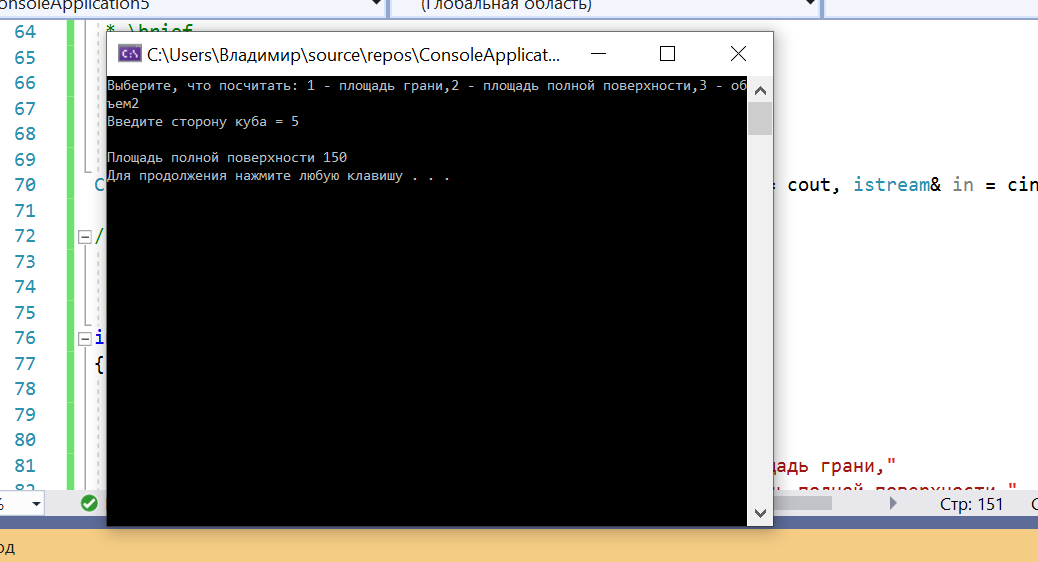
****

Рисунок 3 – Решение тестового примера на C++

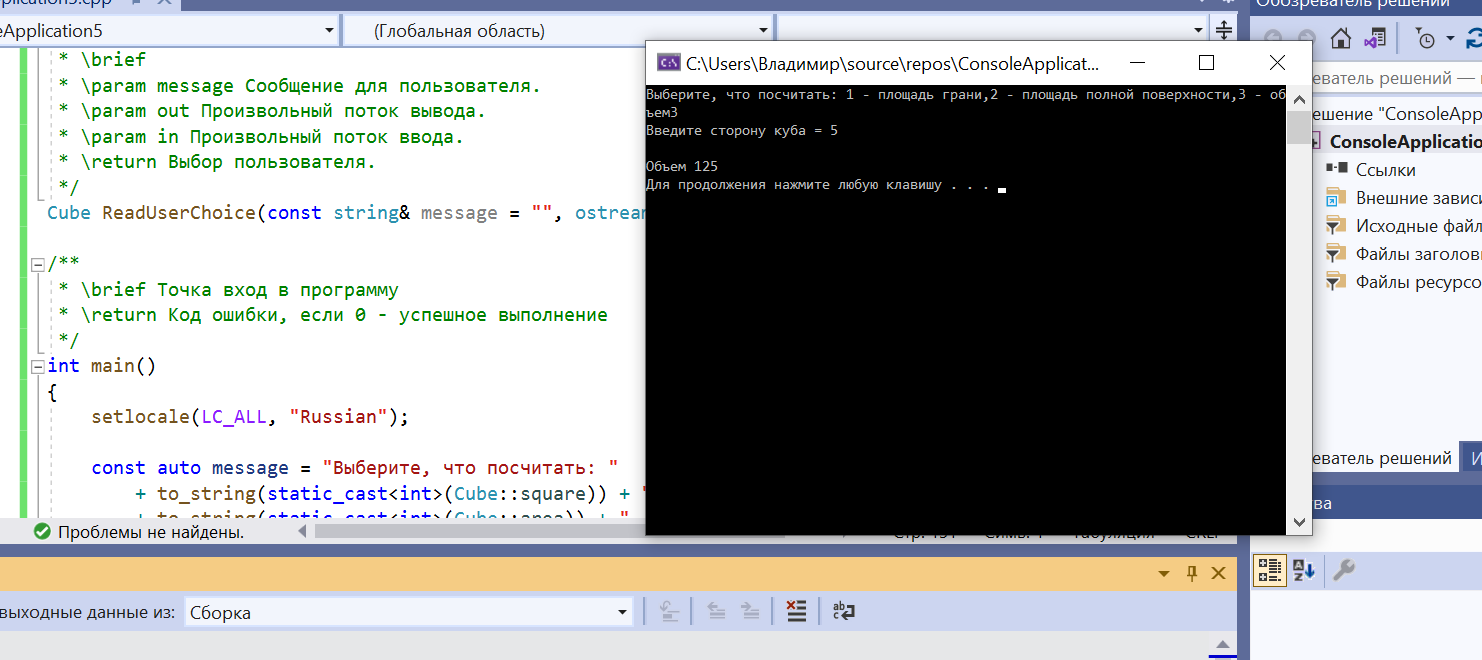
****

Рисунок 4 – Решение тестового примера на C++

# **Решение тестовых примеров Excel**

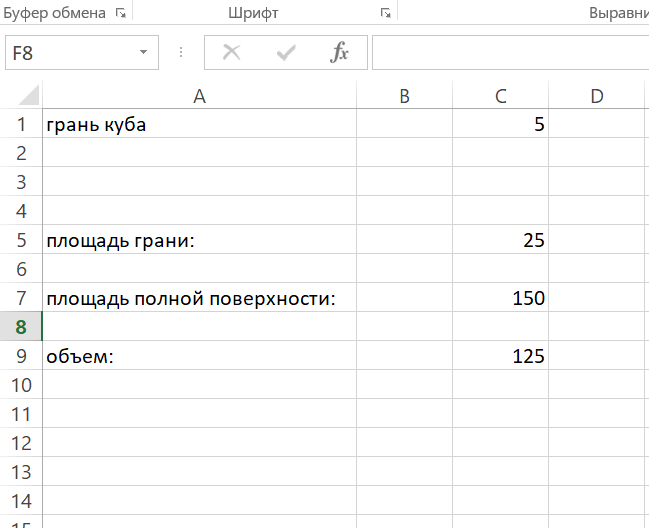
****

Рисунок 5 – Решение тестового примера в Excel

# **Поясняющие формулы**

Площадь грани

a \* a

Площадь полной поверхности куба

6 \* a \* a

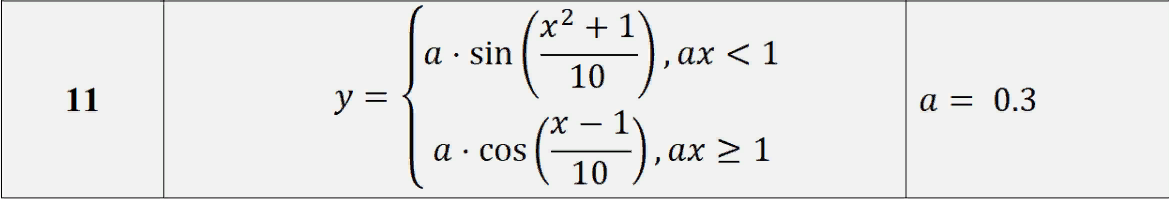
Объем куба

a \* a \* a

# **Задание 2-2**

# **Формулировка задания**

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

****

# **Блок-схема алгоритма**

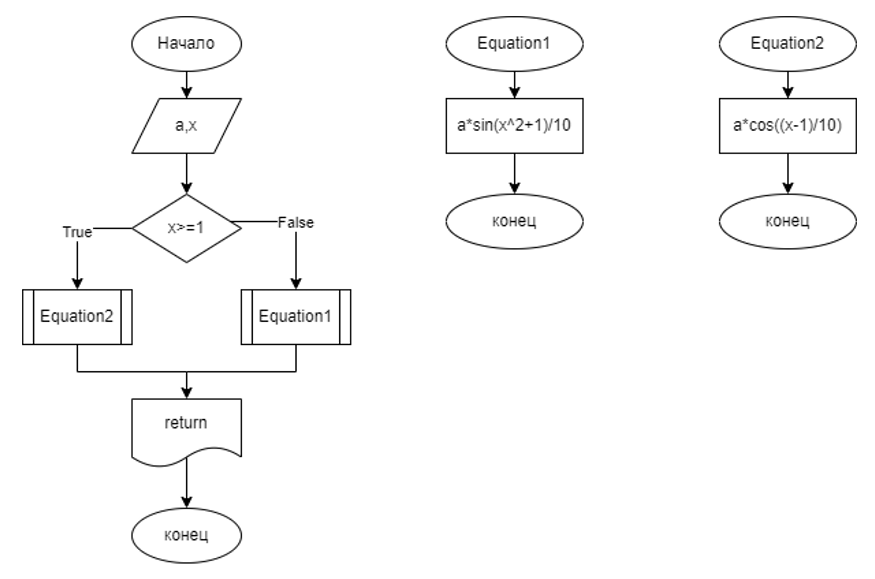
****

Рисунок 6 – Блок-схема алгоритма

# **Программа на языке С++**

#define \_USE\_MATH\_DEFINES // for C++

#include <iostream>

#include <cmath>

using namespace std;

double RamifiedProcess(double const x, double const a);

double Equation1(double const x, double const a);

/\*\*

\*\brief функция выполняет вычисление при x\*a<1

\*\param const a - константа

\*\param x - число, вводимое пользователем

\*\return возвращает значение функции

\*\*/

double Equation2(double const x, double const a);

/\*\*

\*\brief функция выполняет вычисление при a\*x>=1

\*\param const a - константа

\*\param x - число, вводимое пользователем

\*\return возвращает значение функции

\*\*/

double InputValue();

/\*\*

\* brief точка входа в программу

\* \return возвращает 0 ы случае успешного выполнения

\*\*/

int main()

{

setlocale(0, "");

const double x = InputValue();

const double a = 0.3;

cout << RamifiedProcess(x, a) << endl;

system("pause");

return 0;

}

double InputValue() {

double value;

cout << "Введите число: ";

cin >> value;

return value;

}

double RamifiedProcess(double const x, double const a)

{

if (a \* x < 1)

return Equation1(x, a);

else

return Equation2(x, a);

}

double Equation1(double const x, double const a)

{

return a \* sin((pow(x, 2) + 1) / 10);

}

double Equation2(double const x, double const a)

{

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

}

# **Решение тестовых примеров C++**

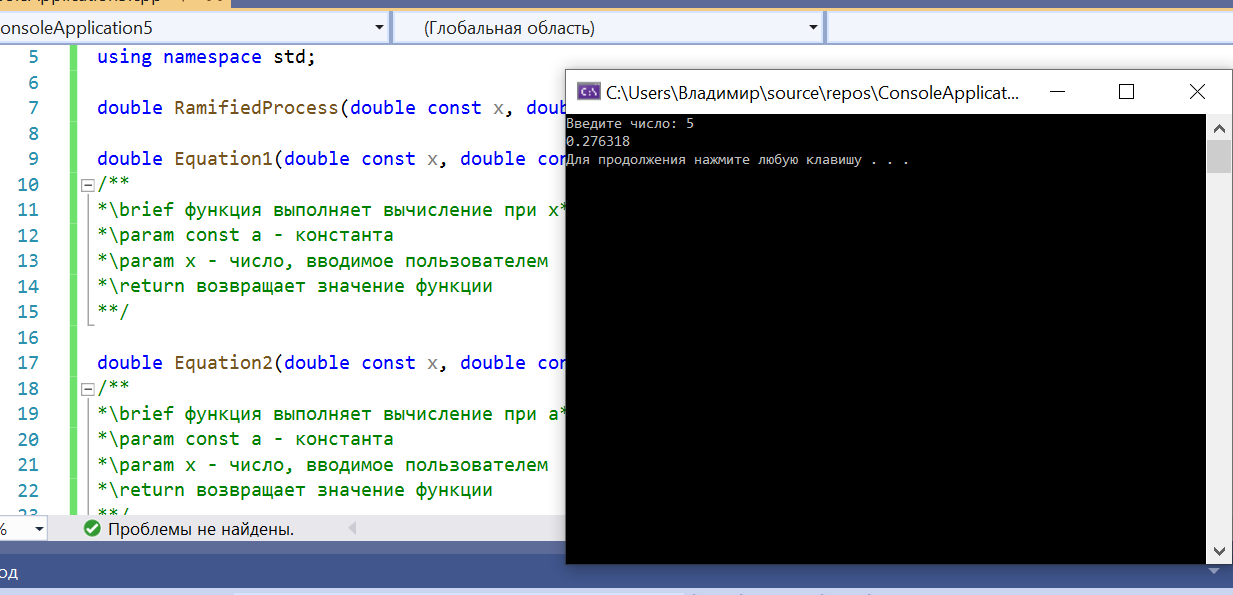
****

Рисунок 7 – Решение тестового примера на C++

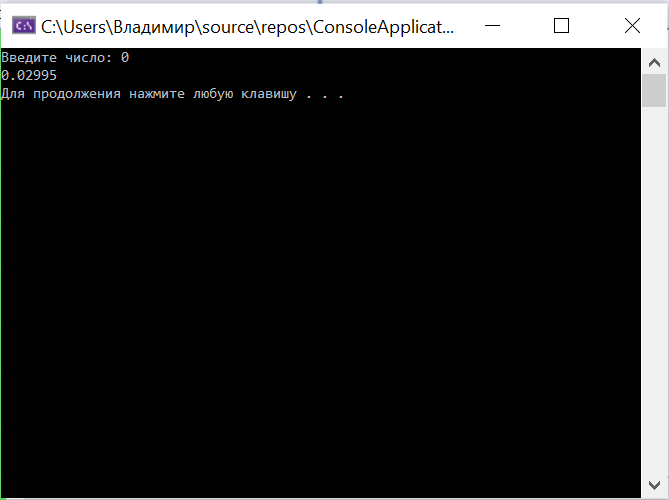
****

Рисунок 8 – Решение тестового примера на C++

# **Решение тестовых примеров Excel**

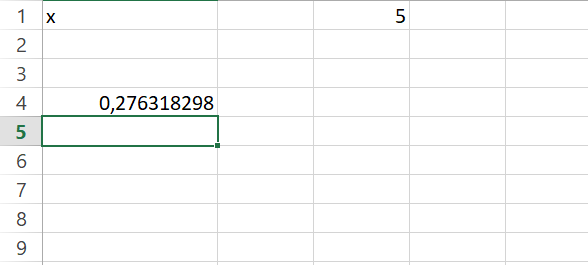
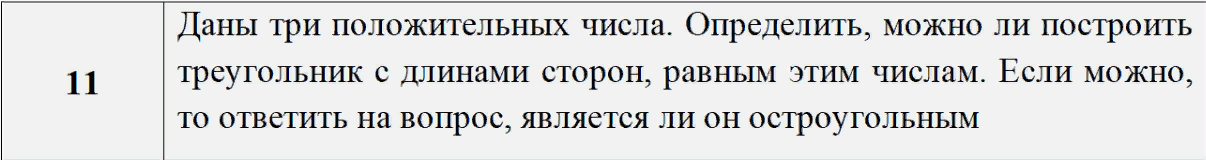
****

Рисунок 9 – Решение тестового примера в Excel

# **Задание 2-3**

# **Формулировка задания**

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

****

# **Блок-схема алгоритма**

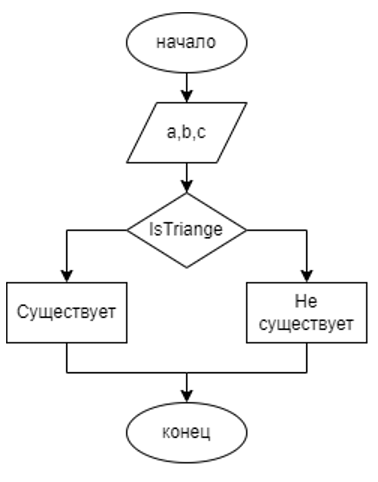
****

Рисунок 10 – Блок-схема алгоритма

# **Программа на языке С++**

#include <iostream>

#include <cmath>

using namespace std;

double InputSide();

bool IsTriangle(double const A, double const B, double const C);

bool IsAcuteAngle(double const A, double const B, double const C);

int main() {

setlocale(0, "");

const auto SideA = InputSide();

const auto SideB = InputSide();

const auto SideC = InputSide();

if (IsTriangle(SideA, SideB, SideC)) {

cout << "Треугольник с данными сторонами существует" << endl;

if (IsAcuteAngle(SideA, SideB, SideC))

cout << "Треугольник остроугольный" << endl;

else

cout << "Треугольник не остроугольный" << endl;

}

else

cout << "Не существует треугольника с данными сторонами" << endl;

system("pause");

return 0;

}

double InputSide() {

double side;

cout << "Введите сторону треугольника: ";

cin >> side;

return side;

}

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

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

}

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

return (pow(b, 2) + pow(c, 2) - pow(a, 2) > 0) && (pow(a, 2) + pow(b, 2) - pow(c, 2) > 0) && (pow(a, 2) + pow(c, 2) - pow(b, 2) > 0);

}

# **Решение тестовых примеров C++**

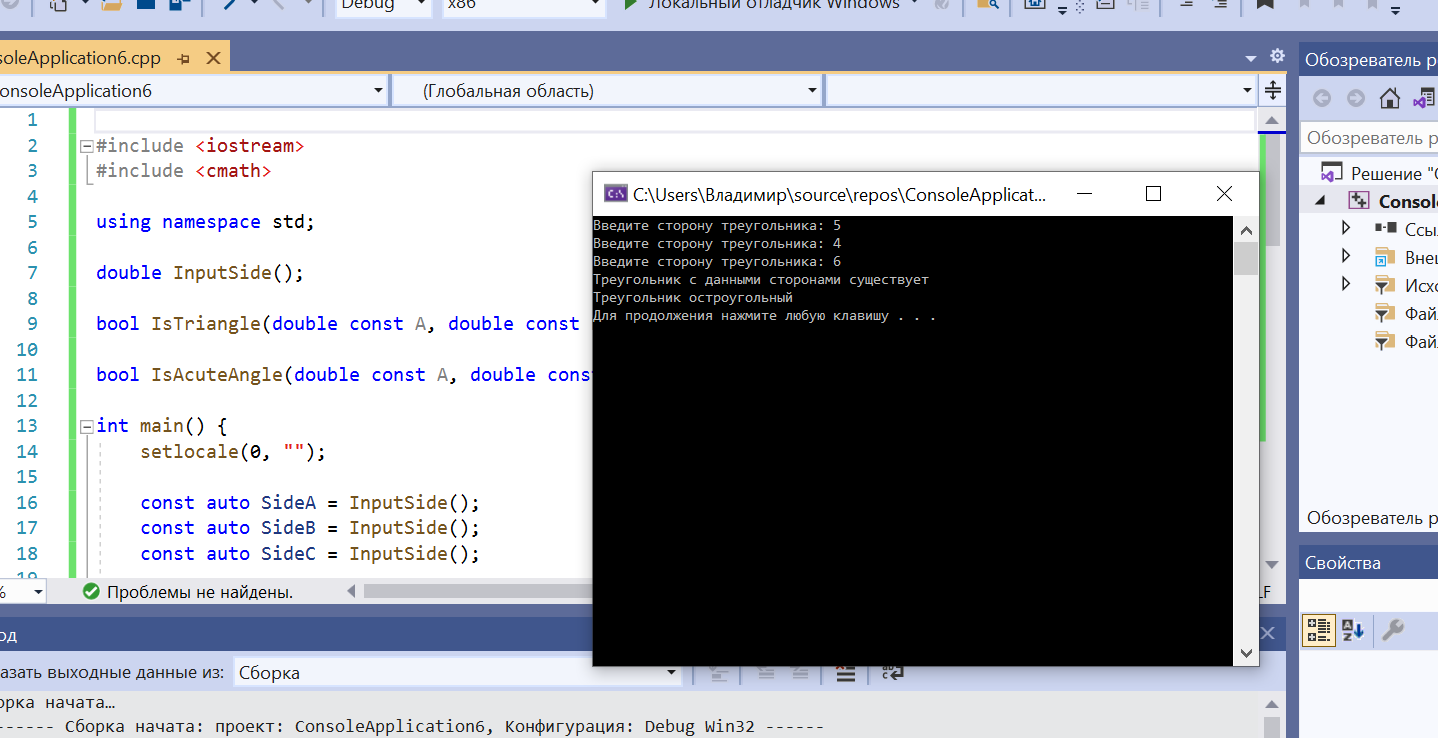
****

Рисунок 11 – Решение тестового примера на C++