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 5

Выполнил: ст. гр. ТКИ-142

Гаспарян Артём Артурович

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

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

Москва 2024

**Оглавление**

[1. ЗАДАНИЕ 2.1 3](#_Toc183906758)

[1.1 Формулировка задания 3](#_Toc183906759)

[1.2 Блок-схема алгоритма 3](#_Toc183906760)

[1.3 Текст программы на языке С 3](#_Toc183906761)

[1.4 Результат выполнения программы 3](#_Toc183906762)

[1.5 Выполнение тестовых примеров 3](#_Toc183906763)

[1.6 Отметка о выполнение задания 3](#_Toc183906764)

[2. ЗАДАНИЕ 2.2 3](#_Toc183906765)

[2.1 Формулировка задания 3](#_Toc183906766)

[2.2 Блок-схема алгоритма 3](#_Toc183906767)

[2.3 Текст программы на языке С 3](#_Toc183906768)

[2.4 Результат выполнения программы 3](#_Toc183906769)

[2.5 Выполнение тестовых примеров 3](#_Toc183906770)

[2.6 Отметка о выполнение задания 3](#_Toc183906771)

[3. ЗАДАНИЕ 2.3 3](#_Toc183906772)

[3.1 Формулировка задания 3](#_Toc183906773)

[3.2 Блок-схема алгоритма 3](#_Toc183906774)

[3.3 Текст программы на языке С 3](#_Toc183906775)

[3.4 Результат выполнения программы 4](#_Toc183906776)

[3.5 Выполнение тестовых примеров 4](#_Toc183906777)

[3.6 Отметка о выполнение задания 4](#_Toc183906778)

# ЗАДАНИЕ 2.1

* 1. Формулировка задания

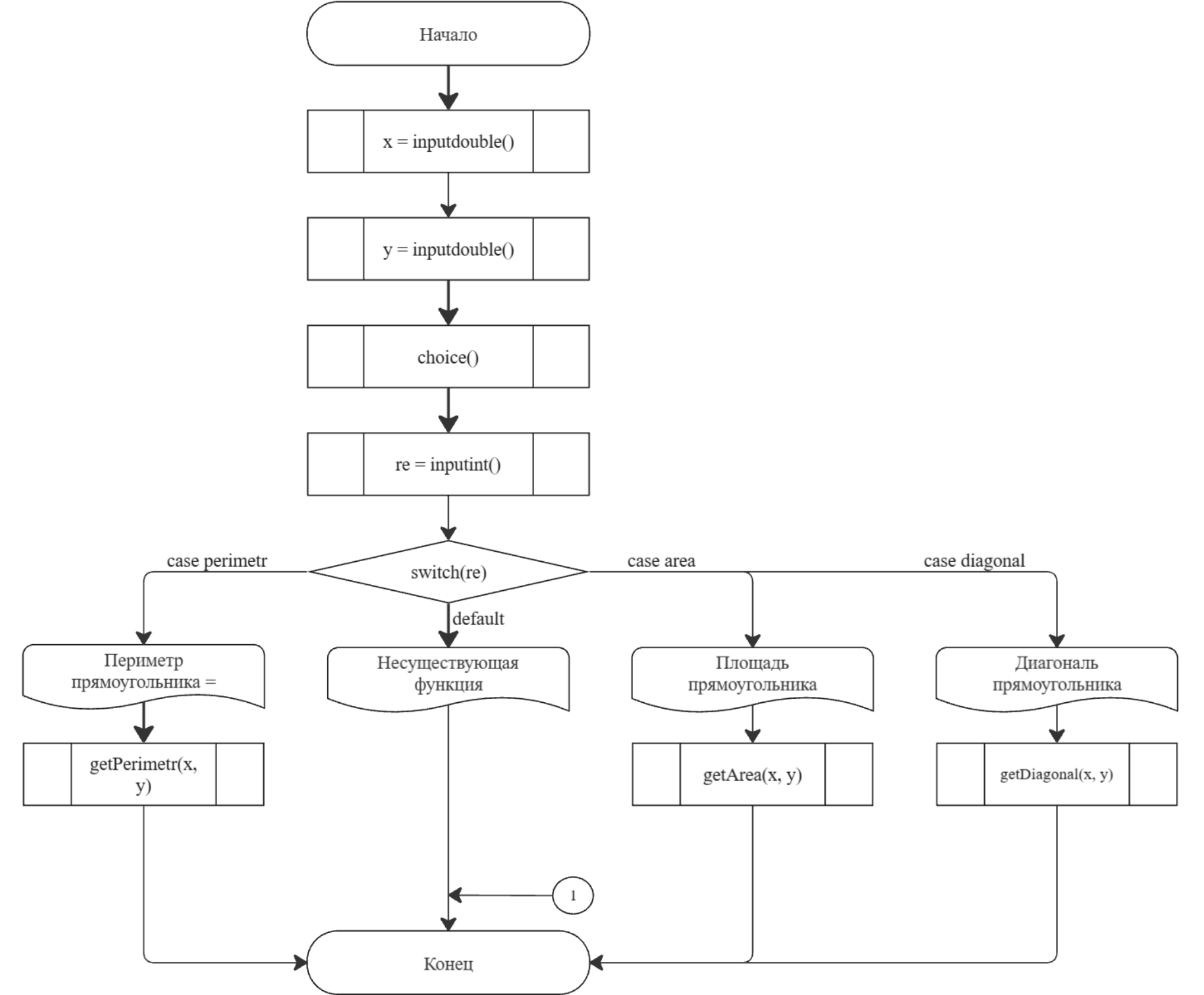
Создать консольное приложение для решения задачи, представленной в таблице (Таблица 1), с использованием перечислимого типа. Выбор формулы вычисления зависит от пользователя. Данные для решения задачи так же вводит пользователь. Ввод необходимо проверять на правильность. Все результаты вывести на экран.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

Таблица 1 – Исходные данные

|  |  |
| --- | --- |
| **Вариант** | **Задача** |
| **5** | Заданы стороны прямоугольника. Определить его:   * Периметр; * Площадь; * Длину диагонали. |

* 1. Блок-схема алгоритма

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

Рисунок 1 – Блок-схема основного алгоритма

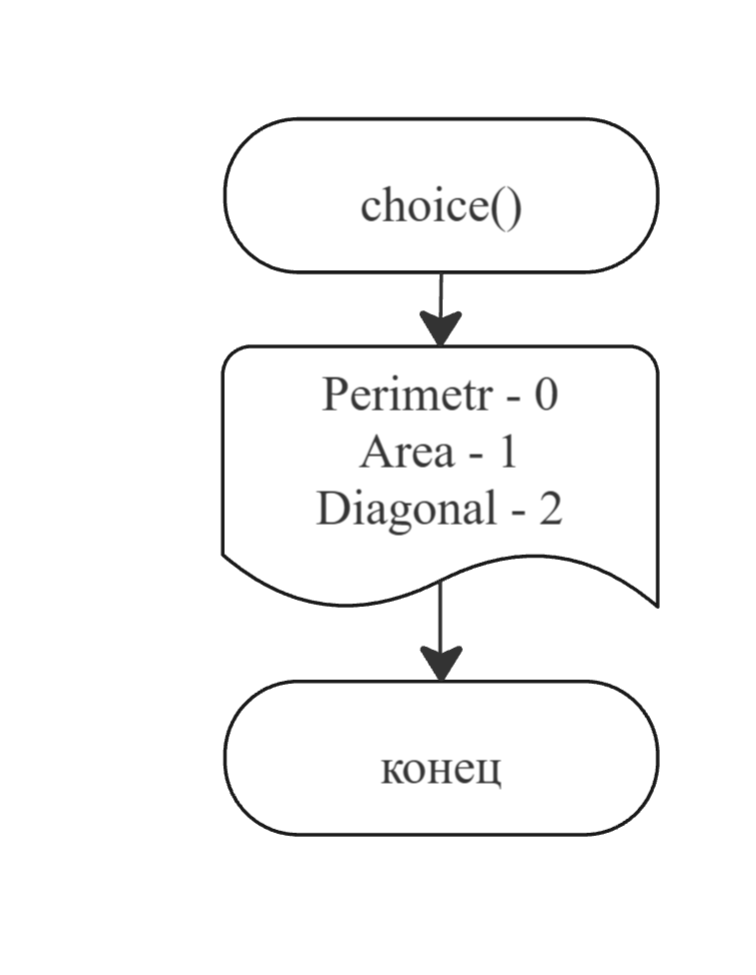


Рисунок 2 – Блок-схема функции choice()

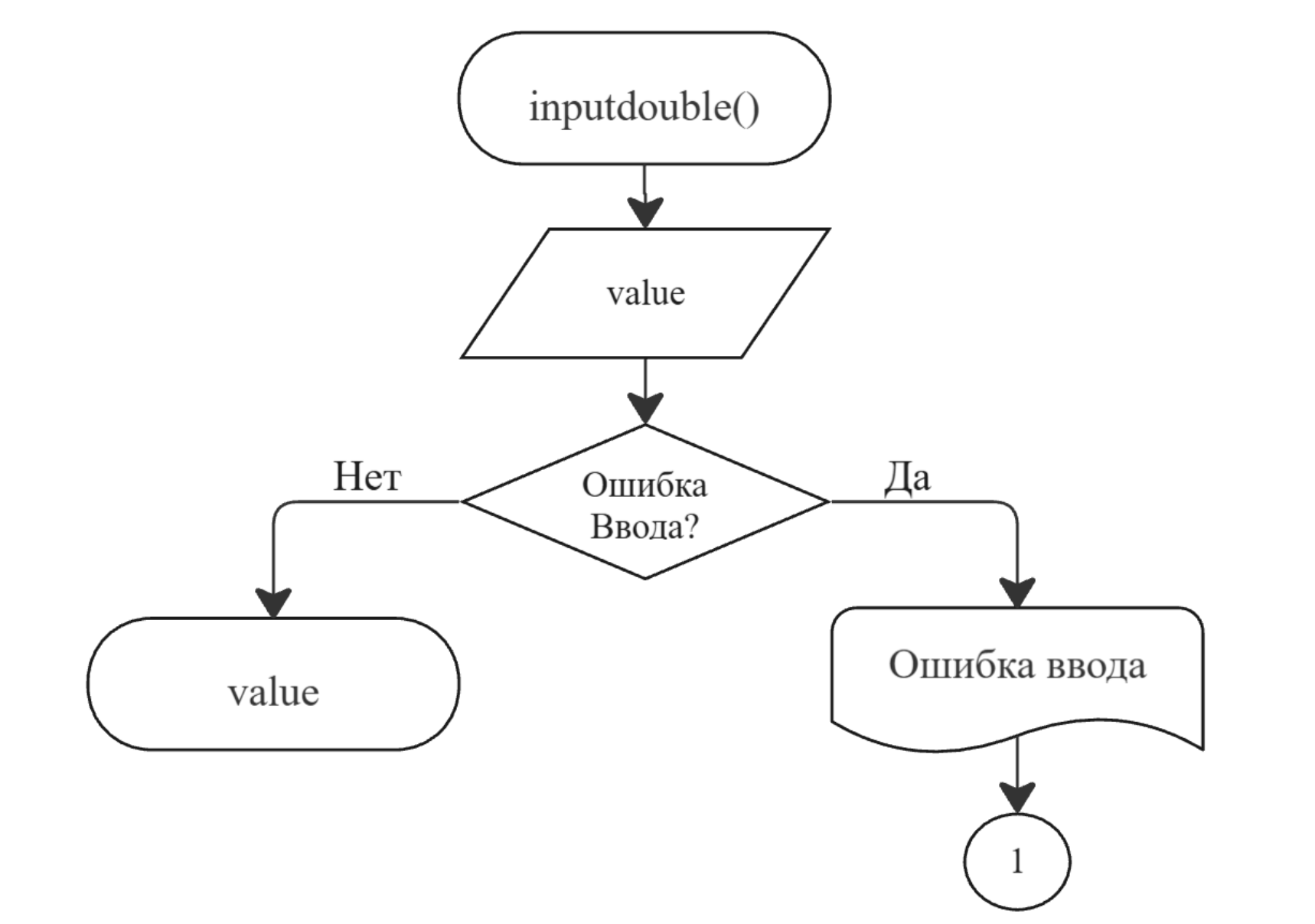


Рисунок 3 – Блок-схема функции inputdouble()

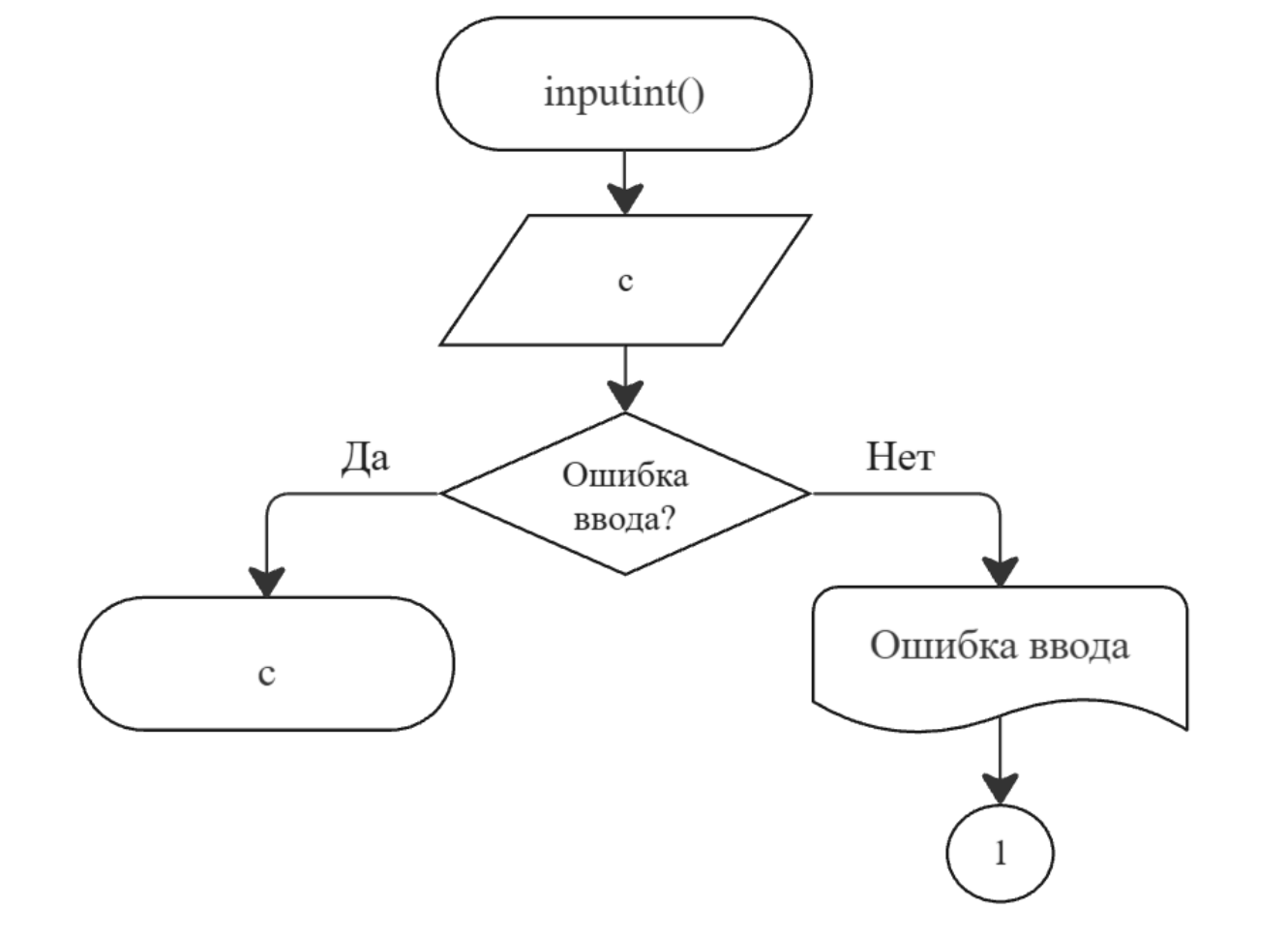


Рисунок 4 – Блок-схема функции inputint()

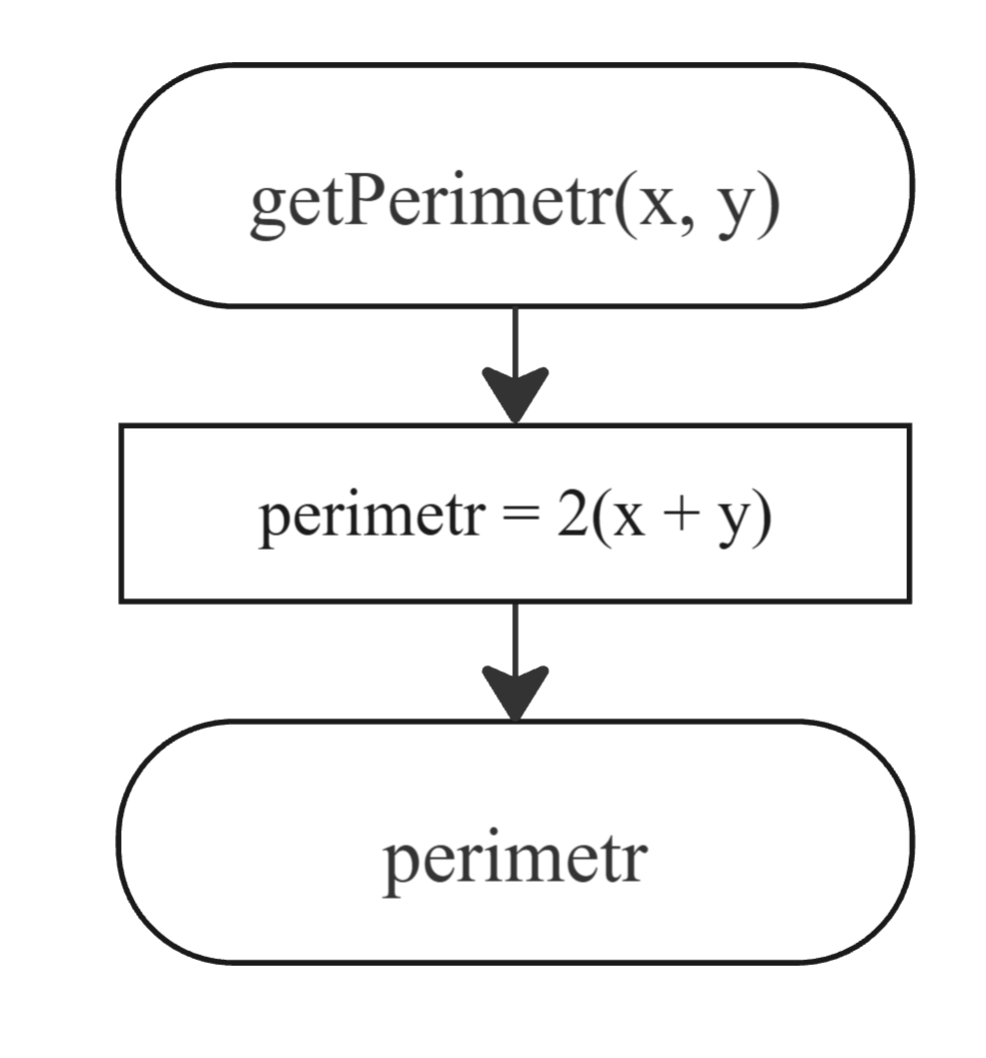


Рисунок 5 – Блок-схема функции getPerimetr()

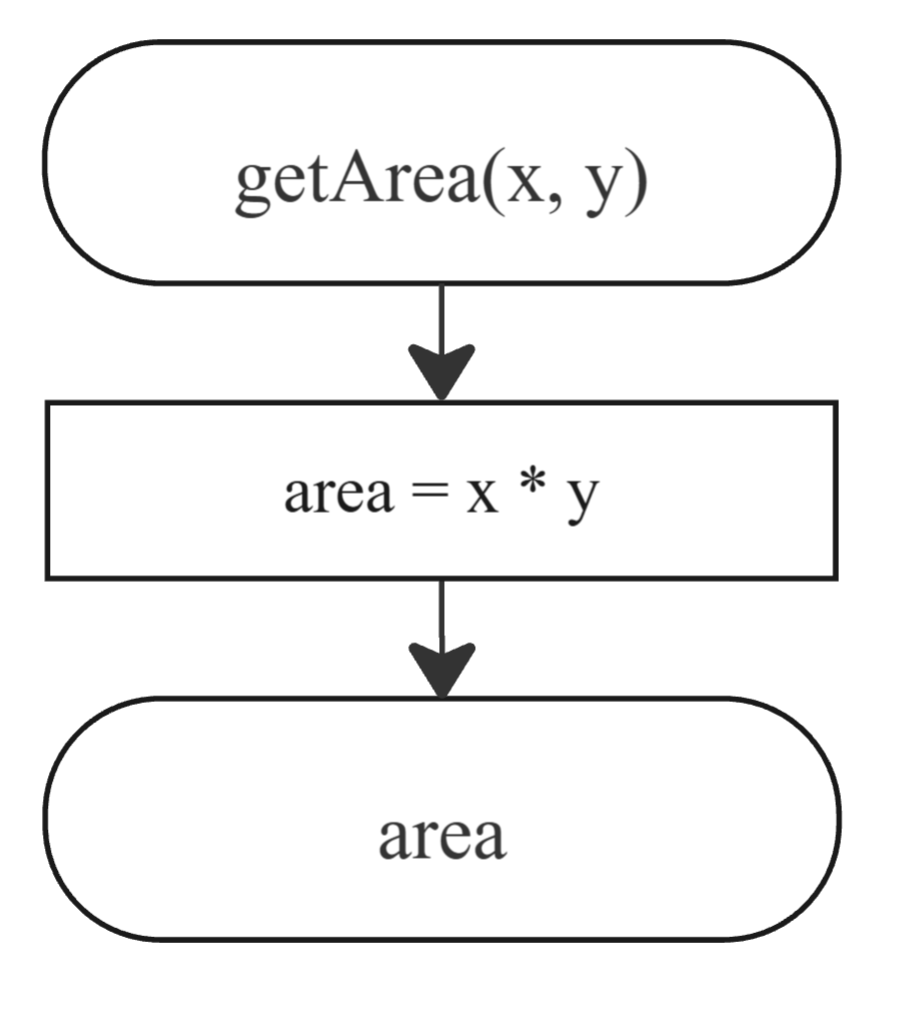


Рисунок 6 – Блок-схема функции getArea()

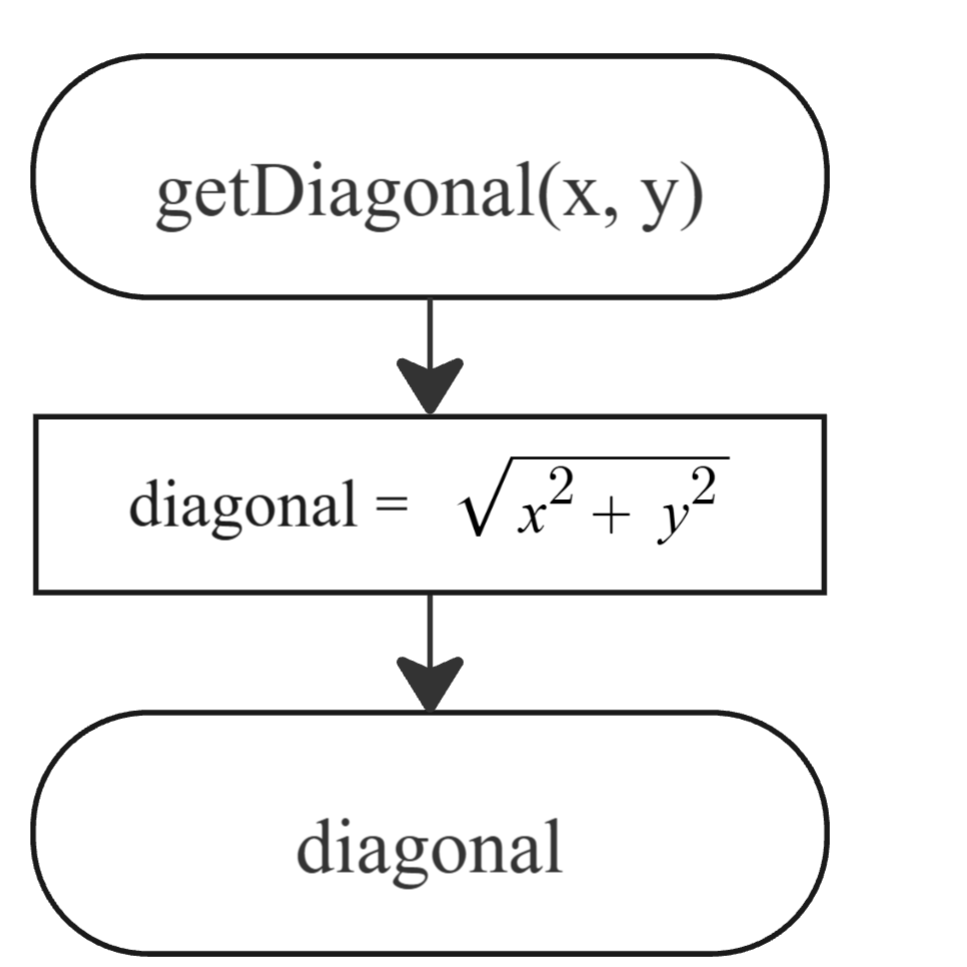


Рисунок 7 – Блок-схема функции getDiagonal()

* 1. Текст программы на языке С

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <errno.h>

/\*\*

\* @brief определяет наши константы

\* @return возвращает их значения

\*/

void determinant(void);

/\*\*

\* @brief проверка получаемого значения

\* @return возвращает проверенное число

\*/

double inputDouble(void);

/\*\*

\* @brief проверка полученного значения

\* @return возвращает полученное значение

\*/

int inputInt(void);

/\*\*

\* @brief рассчитывает периметр прямоугольника

\* @param x значение ширины

\* @param y значение длинны

\* @return рассчитанное значение периметра

\*/

double getPerimetr(double const x, double const y);

/\*\*

\* @brief рассчитывает площадь прямоугольника

\* @param x значение ширины

\* @param y значение длинны

\* @return рассчитанное значения площади прямоугольника

\*/

double getArea(double const x, double const y);

/\*\*

\* @brief рассчитывает диагональ прямоугольника

\* @param x значение ширины

\* @param y значение длинны

\* @return рассчитанное значение диагонали прямоугольника

\*/

double getDiagonal(double const x, double const y);

/\*\*

\* @param perimetr - периметр прямоугольника

\* @param area - площадь прямоугольника

\* @param diagonal - диагонал прямоугольника

\*/

enum request{perimetr, area, diagonal};

/\*\*

\* @brief точка входа в программу

\* @return 0 в случае успехa

\*/

int main(void)

{

double const x = inputDouble();

double const y = inputDouble();

choice();

int re = inputInt();

switch(re)

{

case perimetr:

printf("Периметр прямоугольника = %.2lf\n", getPerimetr(x, y));

break;

case area:

printf("Площадь прямоугольника = %.2lf\n", getArea(x, y));

break;

case diagonal:

printf("Диагональ прямоугольника = %.2lf\n", getDiagonal(x, y));

break;

default:

errno = ERANGE;

perror("Несуществующая функция\n");

exit(EXIT\_FAILURE);

}

return 0;

}

void choice(void)

{

printf("Perimetr - %d\n", perimetr);

printf("Area - %d\n", area);

printf("Diagonal - %d\n", diagonal);

}

double inputDouble(void)

{

double value = 0;

int result= scanf("%lf", &value);

if (result != 1)

{

errno = EIO;

perror("Ошибка ввода\n");

exit(EXIT\_FAILURE);

}

return value;

}

int inputInt(void)

{

int c = 0;

int check = scanf("%d", &c);

if (check != 1)

{

errno = EIO;

perror("Ошибка ввода");

exit(EXIT\_FAILURE);

}

return c;

}

double getPerimetr(double const x, double const y)

{

return 2 \* (x + y);

}

double getArea(double const x, double const y)

{

return x \* y;

}

double getDiagonal(double const x, double const y)

{

return sqrt(pow(x, 2) + pow(y, 2));

}

* 1. Результат выполнения программы

Результаты выполнения программы представлены ниже (Рисунок 8, Рисунок 9, Рисунок 10, Рисунок 11, Рисунок 12).

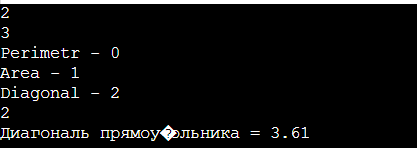


Рисунок 8 – выполнение расчета диагонали

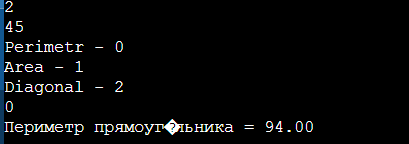


Рисунок 9 – выполнение расчета периметра

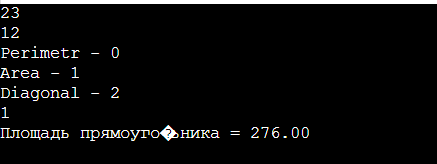


Рисунок 10 – выполнение расчета площади

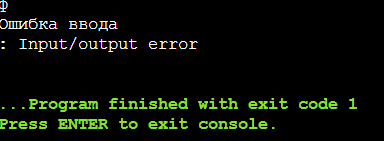


Рисунок 11 – случай, когда x или y введены некорректно

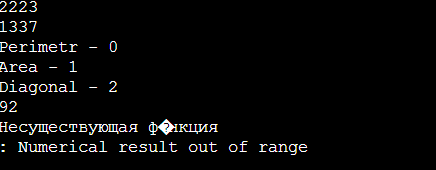


Рисунок 12 – случай, когда пользователь выбрал некорректную операцию

* 1. Выполнение тестовых примеров

В программе MS Excel были выполнены тестовые примеры. Результат их выполнения представлен ниже (Рисунок 13, Рисунок 14, Рисунок 15).

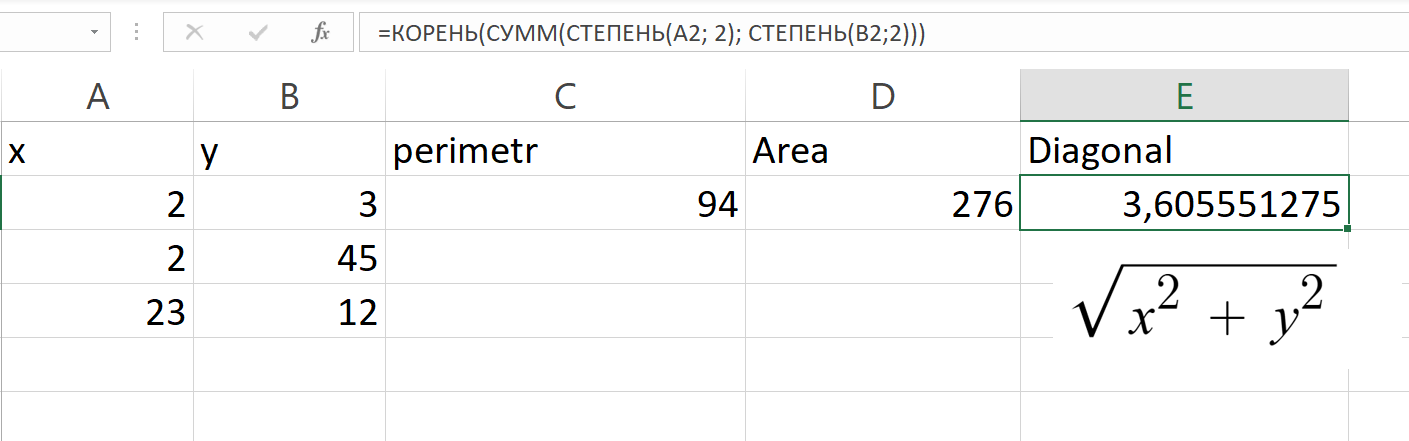


Рисунок 13 – расчет диагонали

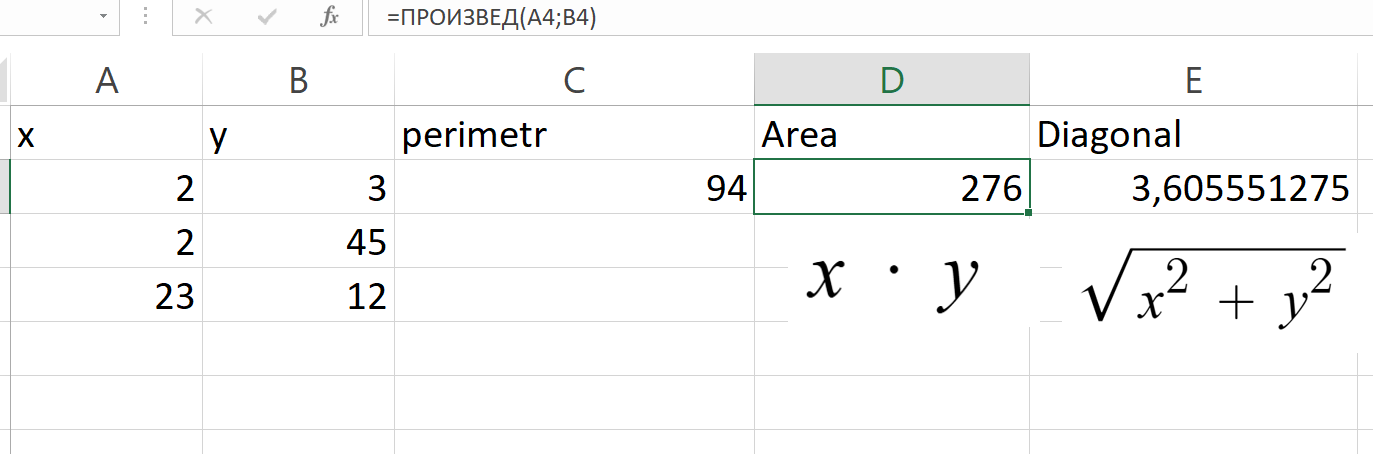


Рисунок 14 – Расчет площади

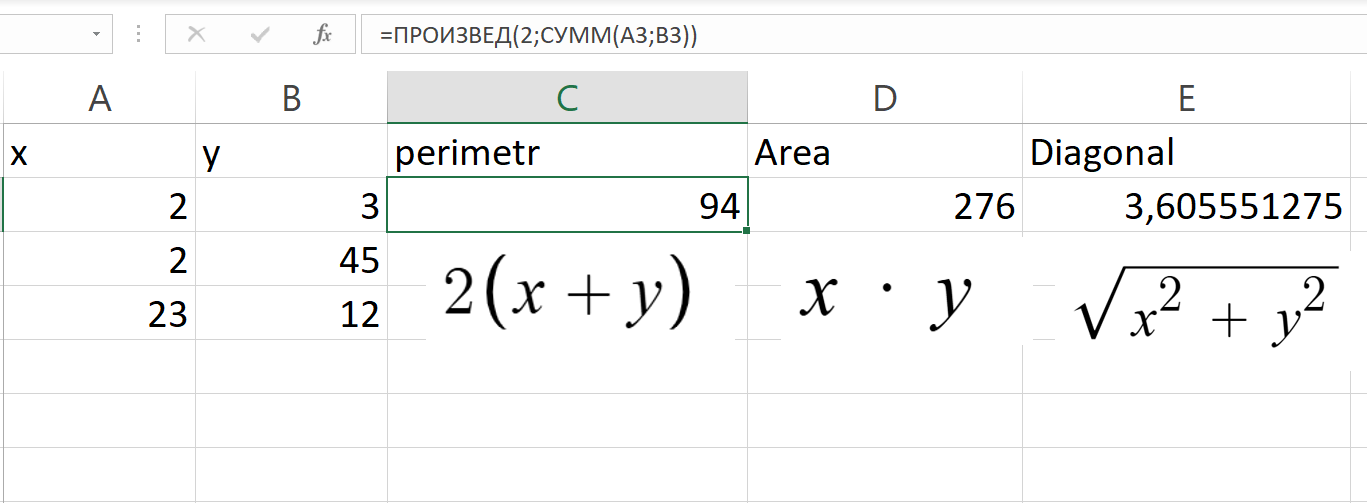


Рисунок 15 – расчет периметра

* 1. Отметка о выполнение задания

Отметка о выполнение задания в GitHub (Рисунок 16).

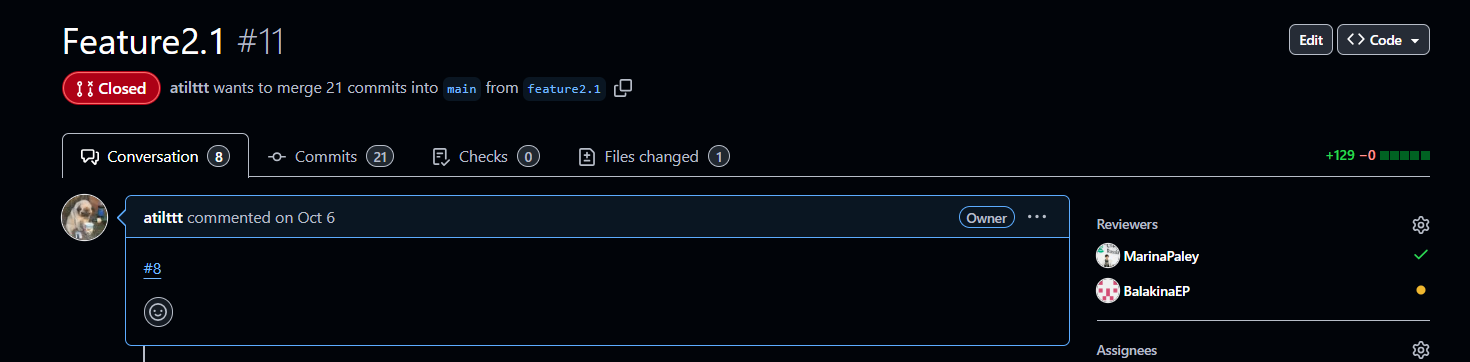


Рисунок 16 – Отметка о выполнение задания

1. задание 2.2
   1. Формулировка задания

Создать консольное приложение с организацией разветвлённого процесса для решения заданной в таблице функции (Таблица 2). Данные для решения вводит пользователь. Ввод необходимо проверять на правильность. Все результаты вывести на экран. Дополнить свой отчёт блок-схемой алгоритма.

Таблица 2 – Исходные данные

|  |  |  |
| --- | --- | --- |
| **Вариант** | **Функция** | **Константы** |
| **5** |  |  |

* 1. Блок-схема алгоритма

Блок-схема основного алгоритма представлена ниже (Рисунок 17). Блок-схемы функций расчета (Рисунок 18, Рисунок 19).

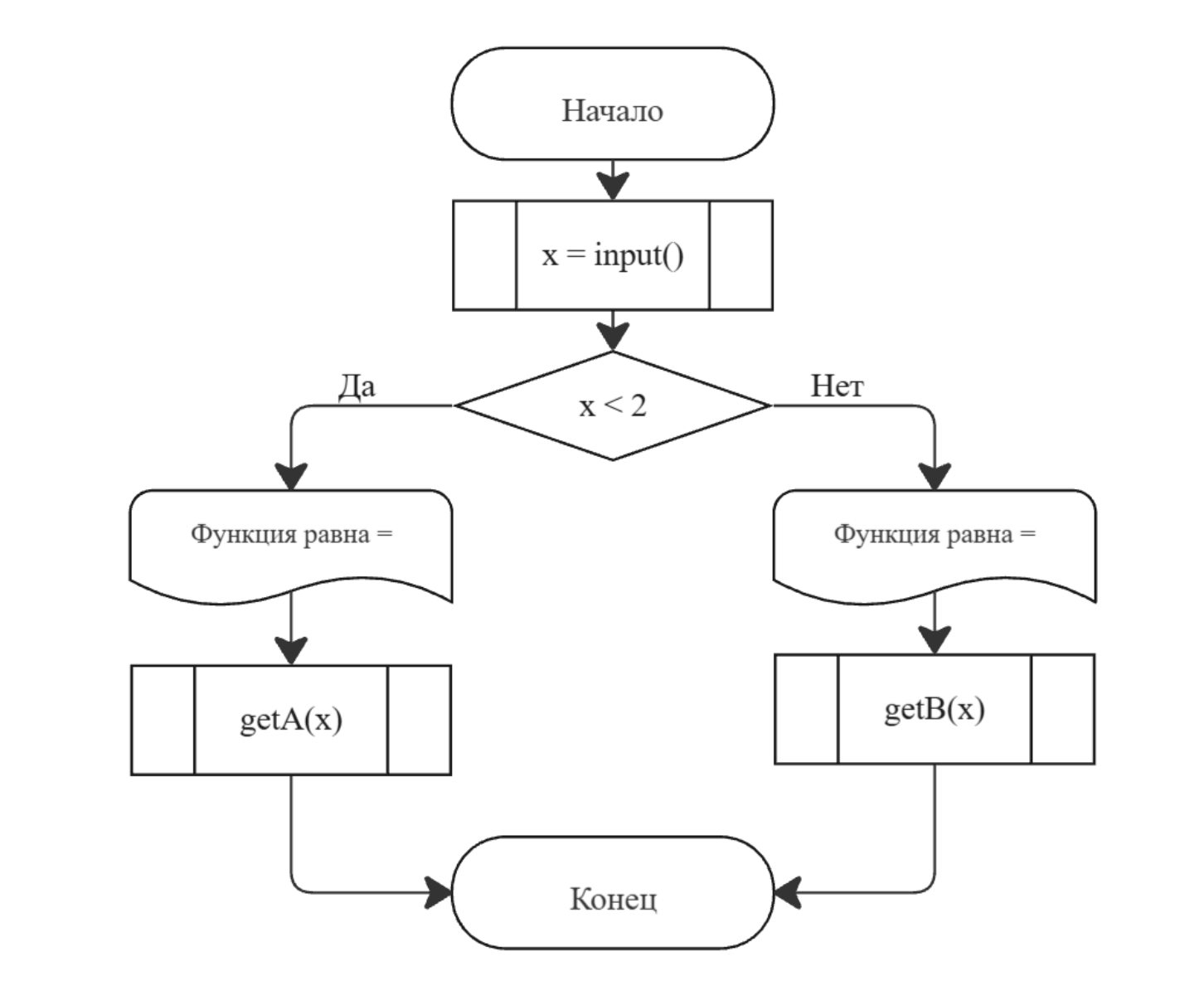
****

Рисунок 17 – Блок-схема основного алгоритма

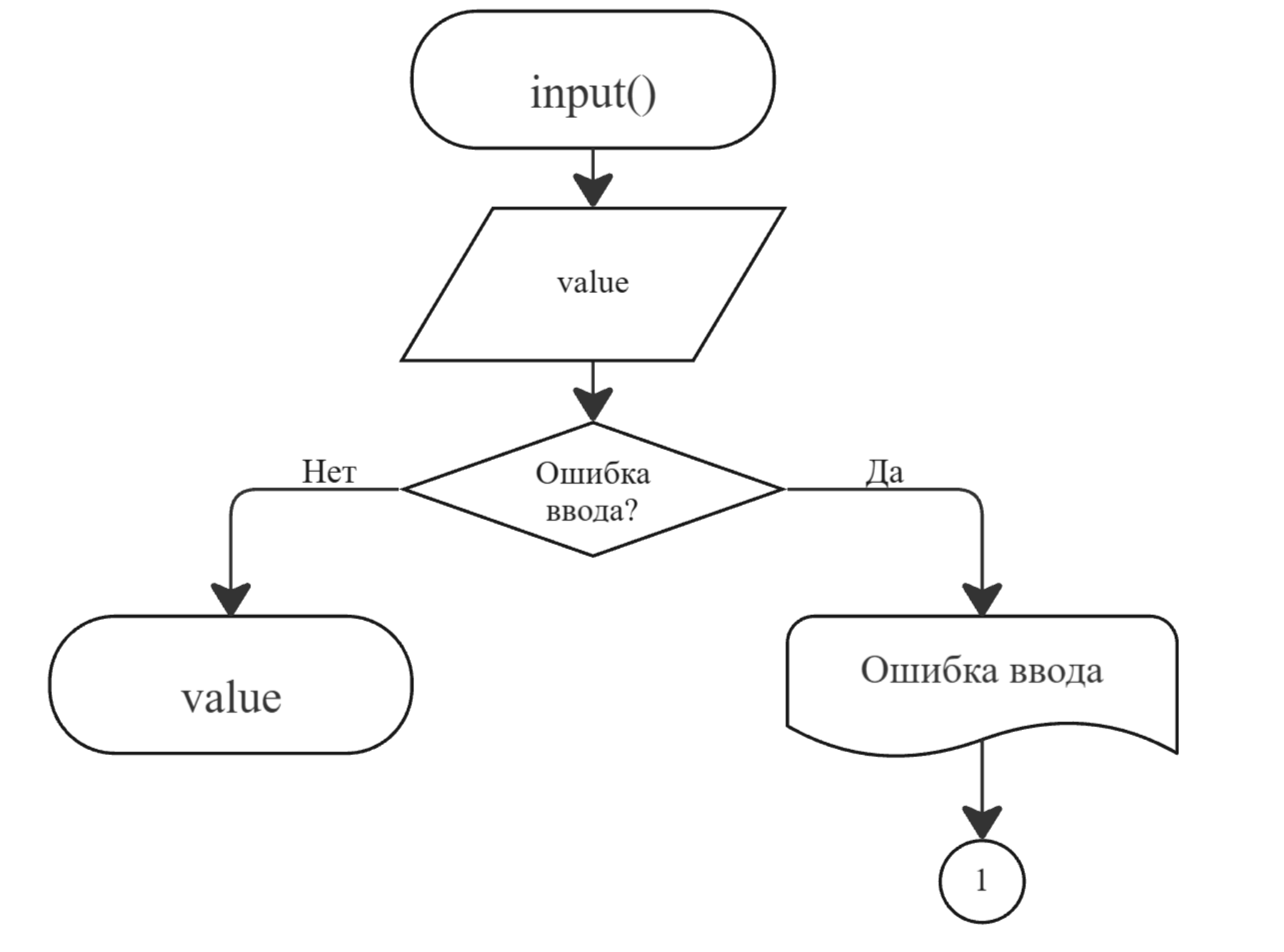


Рисунок 18 – Блок-схема функции input()

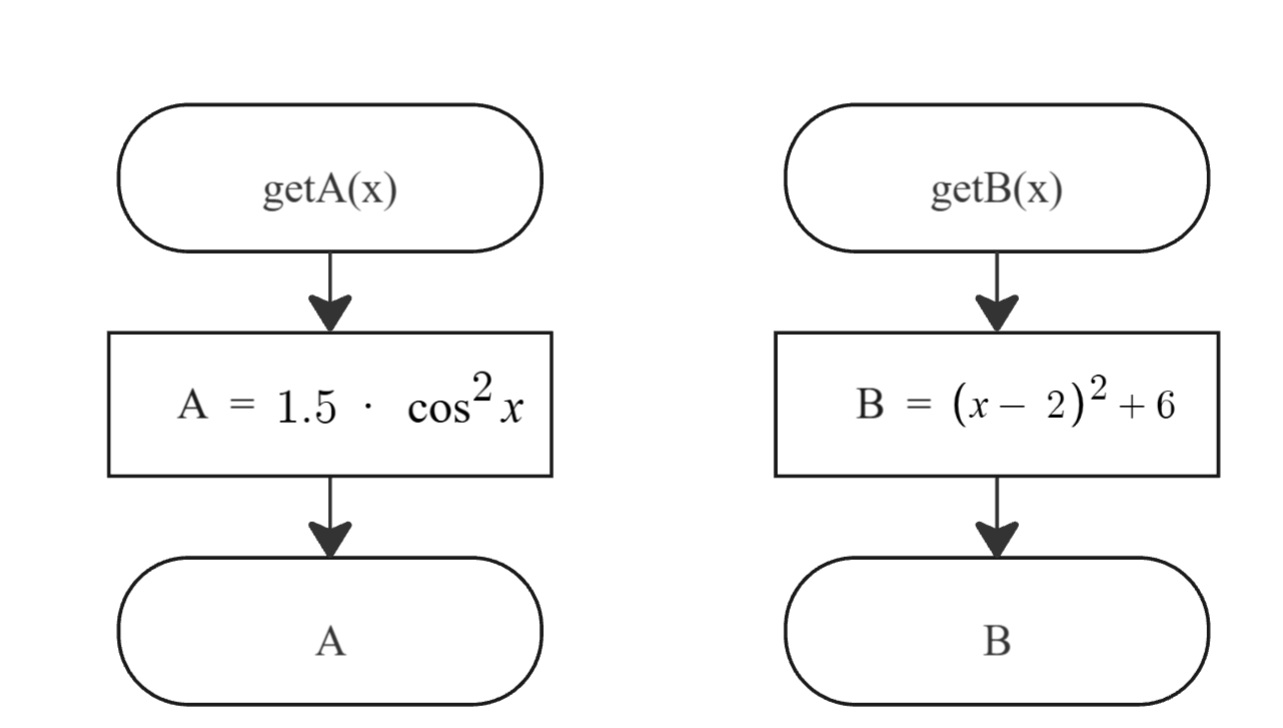


Рисунок 19 – Блок-схема функций getA(x), getB(x)

* 1. Текст программы на языке С

#include <stdio.h>

#include <math.h>

#include <stdlib.h>

#include <errno.h>

#include <float.h>

/\*\*

\* @brief проверка введенного значения

\* @return возвращает значение

\*/

double input(void);

/\*\*

\* @brief рассчитывает значение функции по формуле

\* @param x значение переменной x

\* @return возвращает рассчитанное значение функции, при x<2

\*/

double getA(const double x);

/\*\*

\* @brief рассчитывает значение функции по формуле

\* @param x значение переменной x

\* @return возвращает рассчитанное значение фукнции, при x>=2

\*/

double getB(const double x);

/\*\*

\* @brief точка входа в программу

\* @return 0 в случае успеха

\*/

int main(void)

{

const double x = input();

if (x < 2 - DBL\_EPSILON)

{

printf("функция равна = %.2lf", getA(x));

}

else

{

printf("функция равна = %.2lf", getB(x));

}

return 0;

}

double input(void)

{

double value;

int result = scanf("%lf", &value);

if (result != 1)

{

errno = EIO;

perror("Ошибка ввода\n");

exit(EXIT\_FAILURE);

}

return value;

}

double getA(const double x)

{

return 1.5 \* pow(cos(x), 2);

}

double getB(const double x)

{

return pow(x - 2, 2) + 6;

}

* 1. Результат выполнения программы

Результаты выполнения программы представлены ниже (Рисунок 20, Рисунок 21, Рисунок 22, Рисунок 23).



Рисунок 20 – результат выполнения программы



Рисунок 21 – результат выполнения программы

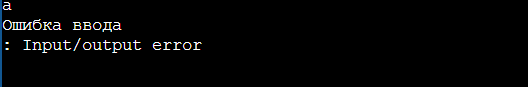


Рисунок 22 – проверка функции input()



Рисунок 23 – результат выполнения программы

* 1. Выполнение тестовых примеров

В программе MS Excel были выполнены тестовые примеры. Результаты их выполнения представлены ниже (Рисунок 24, Рисунок 25, Рисунок 26).

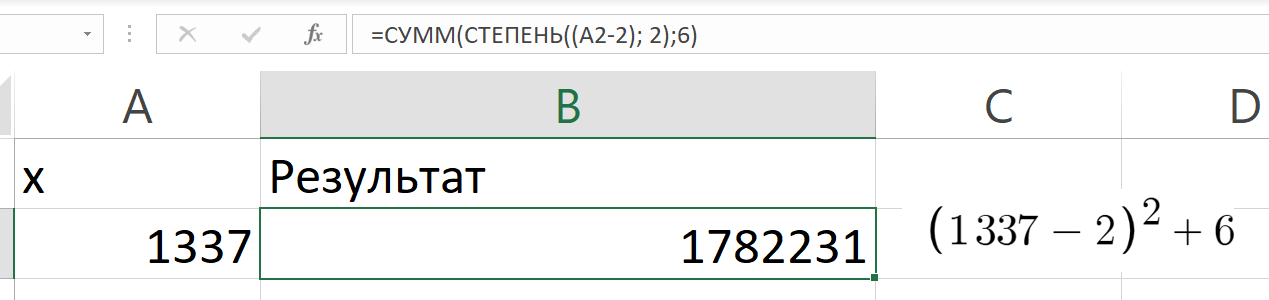


Рисунок 24 – Выполнение первого тестового примера

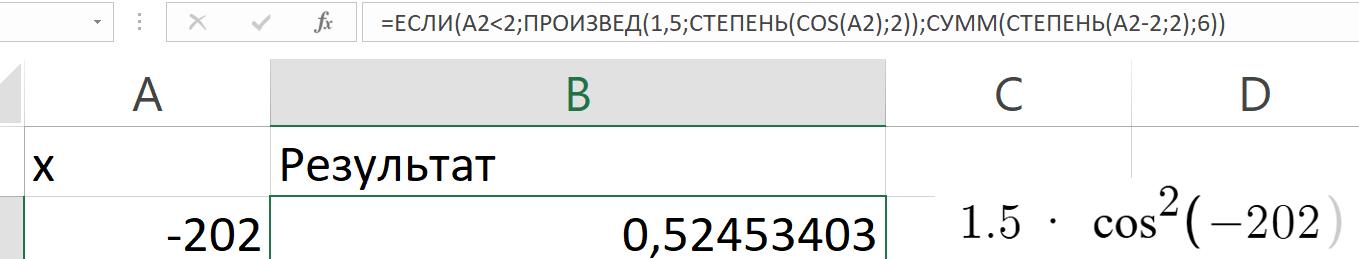


Рисунок 25 – Выполнение второго тестового примера

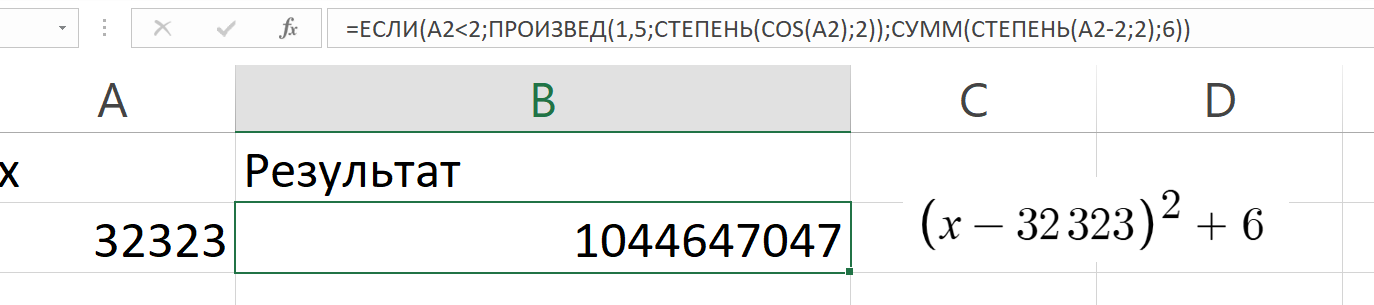


Рисунок 26 – Выполнение третьего тестового примера

2.6 Отметка о выполнение задания

Отметка о выполнение задания в GitHub

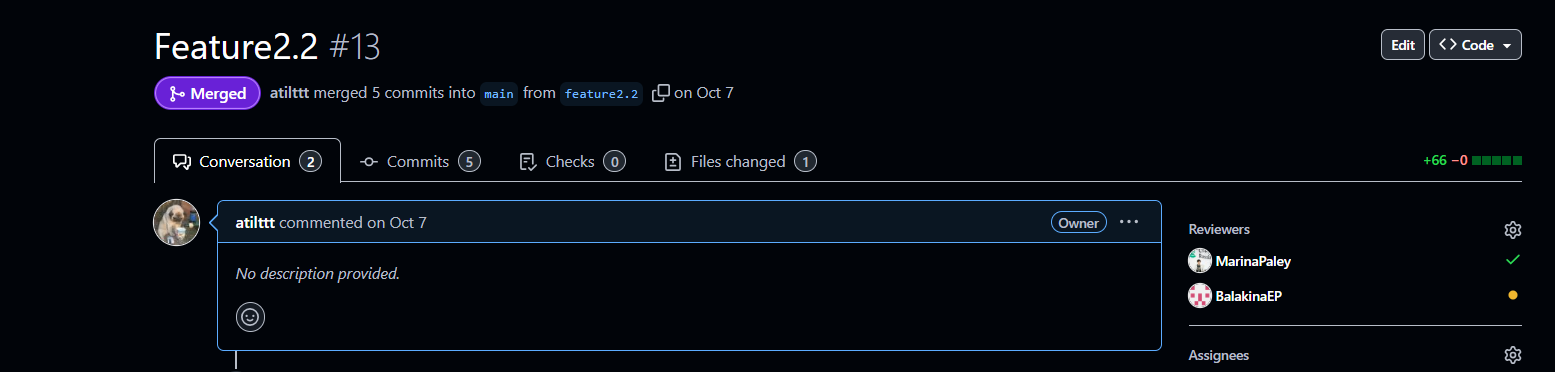


Рисунок 27 – Отметка о выполнении задания

1. задание 2.3
   1. Формулировка задания

Создать консольное приложение с организацией разветвлённого процесса для решения задачи из таблицы (Таблица 3). Данные для решения вводит пользователь. Ввод необходимо проверять на правильность. Все результаты вывести на экран. Дополнить свой отчёт блок-схемой алгоритма.

Таблица 3 – Исходные данные

|  |  |
| --- | --- |
| **Вариант** | **Задача** |
| **5** | Путник двигался *t1* часов со скоростью *v1*, а затем *t2* часов – со скоростью *v2* и *t3* часов – со скоростью *v3*. За какое время он одолел первую половину пути, после чего запланировал привал? |

* 1. Блок-схема алгоритма

Блок-схема основного алгоритма представлена ниже (Рисунок 28). Блок-схемы функций расчета (Рисунок 29, Рисунок 30).

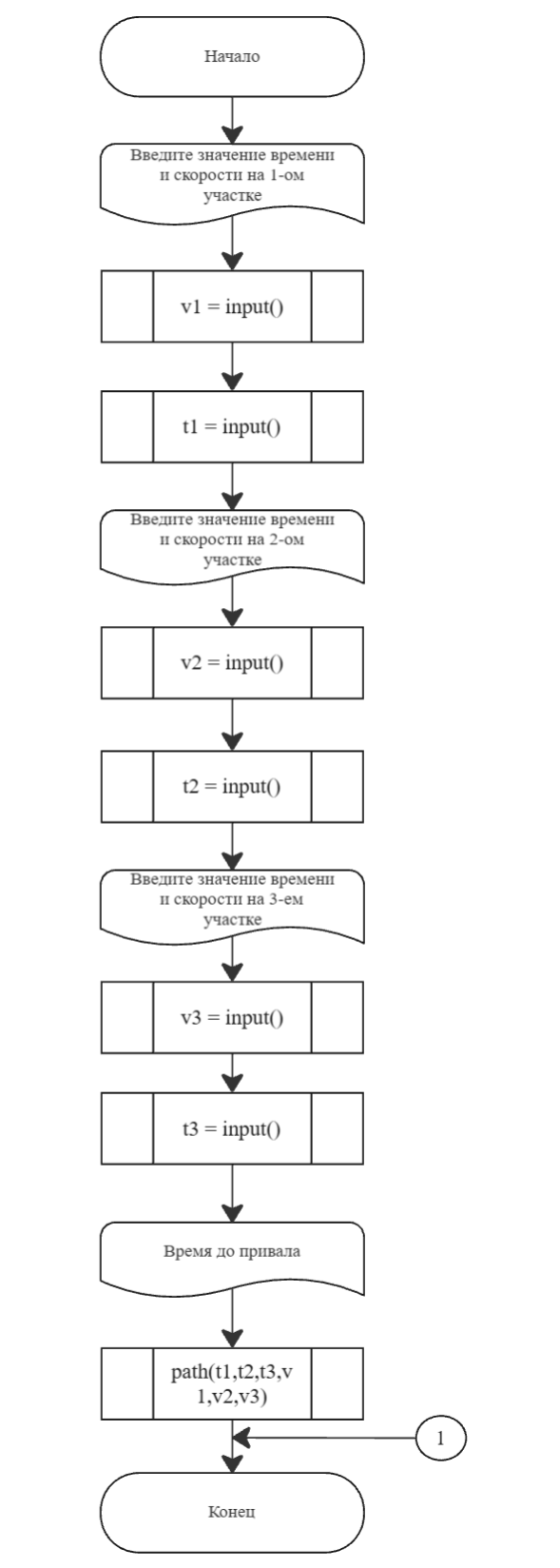


Рисунок 28 – Блок-схема основного алгоритма

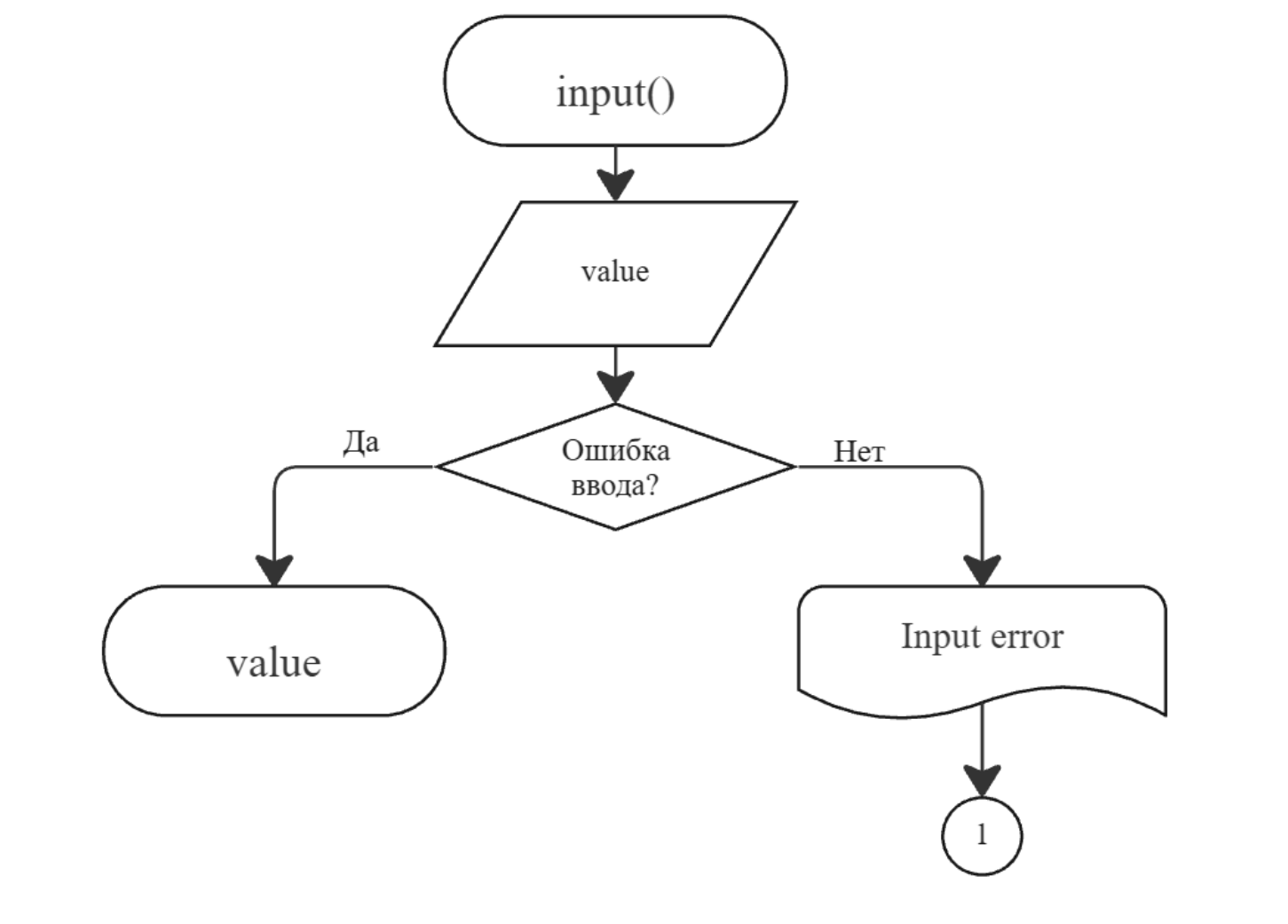


Рисунок 29 – Блок-схема функции input()

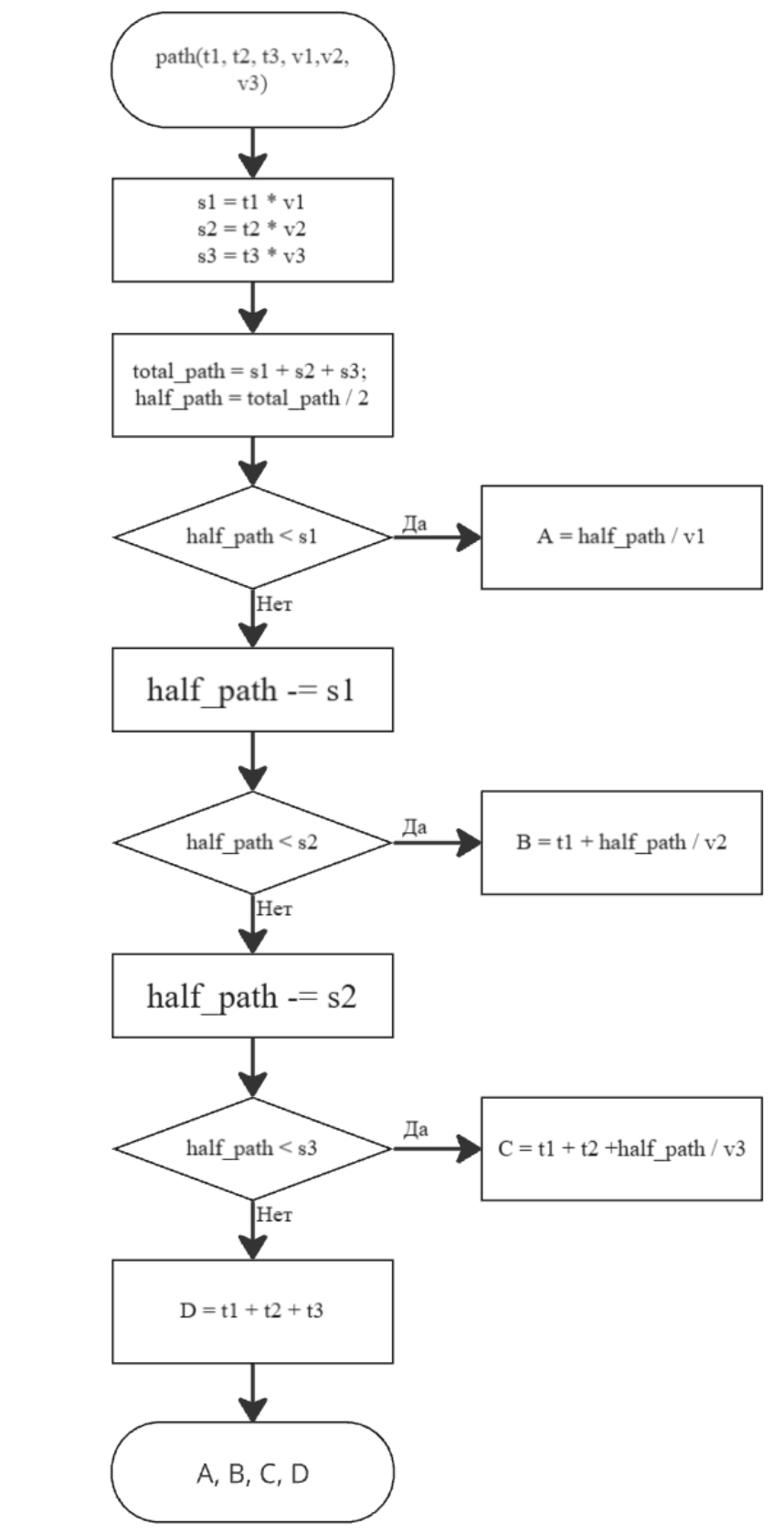


Рисунок 30 – Блок-схема функции path(t1, t2, t2, v1, v2, v3)

* 1. Текст программы на языке С

#include <stdio.h>

#include <math.h>

#include <stdlib.h>

#include <errno.h>

/\*\*

\* @brief получает на вход число из потока ввода

\* @return возвращает введенное число

\*/

double Input(void);

/\*\*

\* @brief рассчитывает время, за которое путник одолел первую половину пути, где сделал привал

\* @param t1 v1 - время и скорость на первом участке

\* @param t2 v2 - время и скорость на втором участке

\* @param t3 v2 - время и скорость на третьем участке

\* @return возвращает рассчитанное значение времени

\*/

double Path(const double t1, const double t2, const double t3, const double v1, const double v2, const double v3);

/\*\*

\* @brief точка входа в программу

\* @return 0 в случае успеха

\*/

int main()

{

printf("Введите значение времени и скорости на 1-ом участке\n");

const double v1 = Input();

const double t1 = Input();

printf("Введите значение времени и скорости на 2-ом участке\n");

const double v2 = Input();

const double t2 = Input();

printf("Введите значение времени и скорости на 3-eм участке\n");

const double v3 = Input();

const double t3 = Input();

printf("Время до привала %.3lf", Path(t1,t2,t3,v1,v2,v3));

return 0;

}

double Input(void)

{

double value = 0.0;

int result = scanf("%lf", &value);

if (result != 1 || value < 0)

{

errno = EIO;

perror("Input error");

exit(EXIT\_FAILURE);

}

return value;

}

double Path(const double t1, const double t2, const double t3, const double v1, const double v2, const double v3)

{

double s1 = t1 \* v1; //расстояние первого участка//

double s2 = t2 \* v2; //расстояние второго участка//

double s3 = t3 \* v3; //расстояние третьего участкого//

double total\_path = s1 + s2 + s3;

double half\_path = total\_path / 2;

if (half\_path < s1)

{

return half\_path / v1; //если путник находится на первом участке//

}

half\_path = half\_path - s1;

if (half\_path < s2)

{

return t1 + (half\_path / v2); //если путник находится на втором участке//

}

half\_path = half\_path - s2;

if (half\_path < s3)

{

return t1 + t2 + (half\_path / v3); //если путник находится на третьем участке//

}

return t1 + t2 + t3;

}

* 1. Результат выполнения программы

Ниже представлены результаты работы программы (Рисунок 31, Рисунок 32, Рисунок 33, Рисунок 34, Рисунок 35).

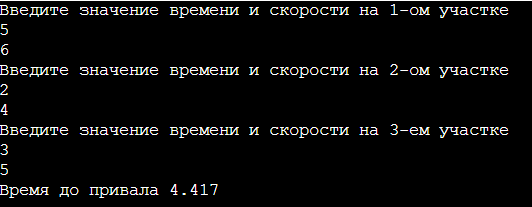


Рисунок 31 – Путник находится на первом участке

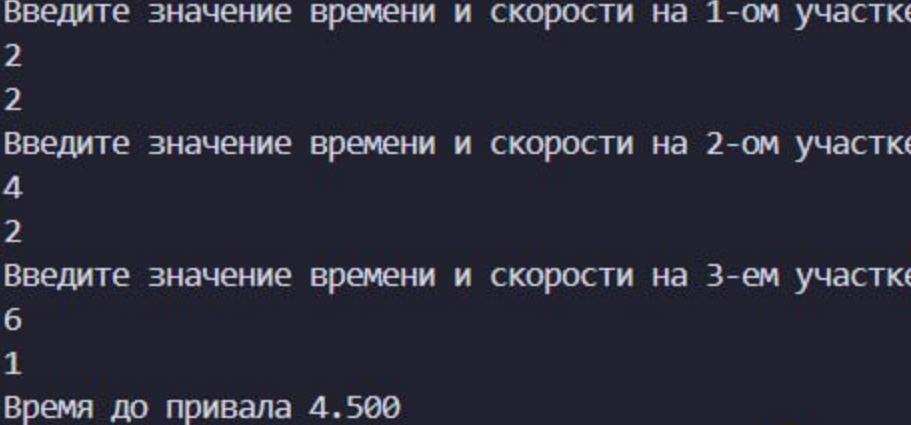


Рисунок 32 – Путник находится на втором участке

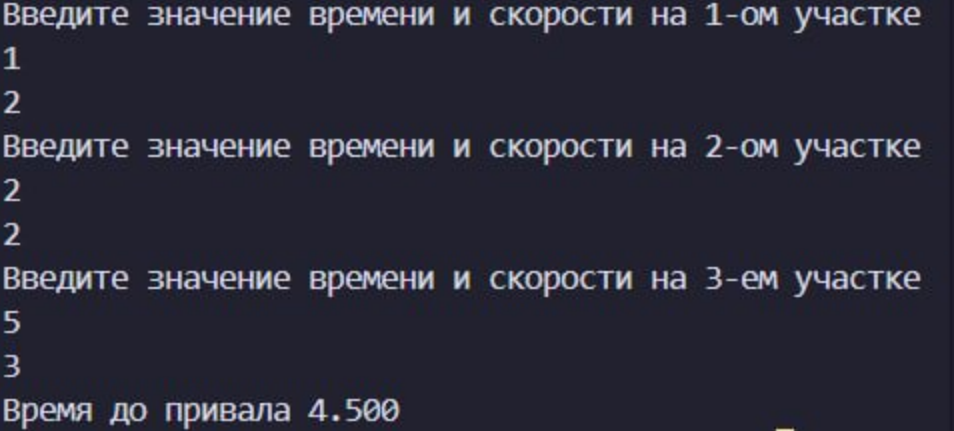


Рисунок 33 – Путник находится на третьем участке

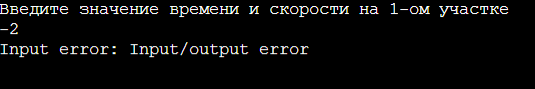


Рисунок 34 – Проверка функции input()

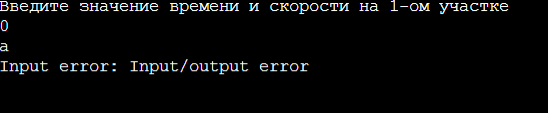


Рисунок 35 – Проверка функции input()

* 1. Выполнение тестовых примеров

В программе MS Excel были выполнены тестовые примеры. Результат их выполнения представлен ниже (Рисунок 36, Рисунок 37, Рисунок 38).

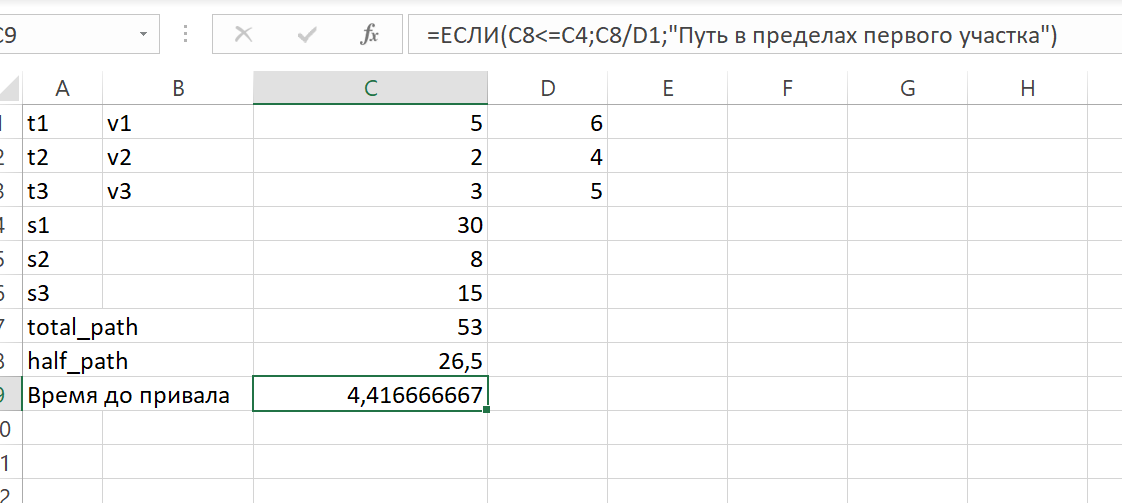


Рисунок 36 – Первый участок

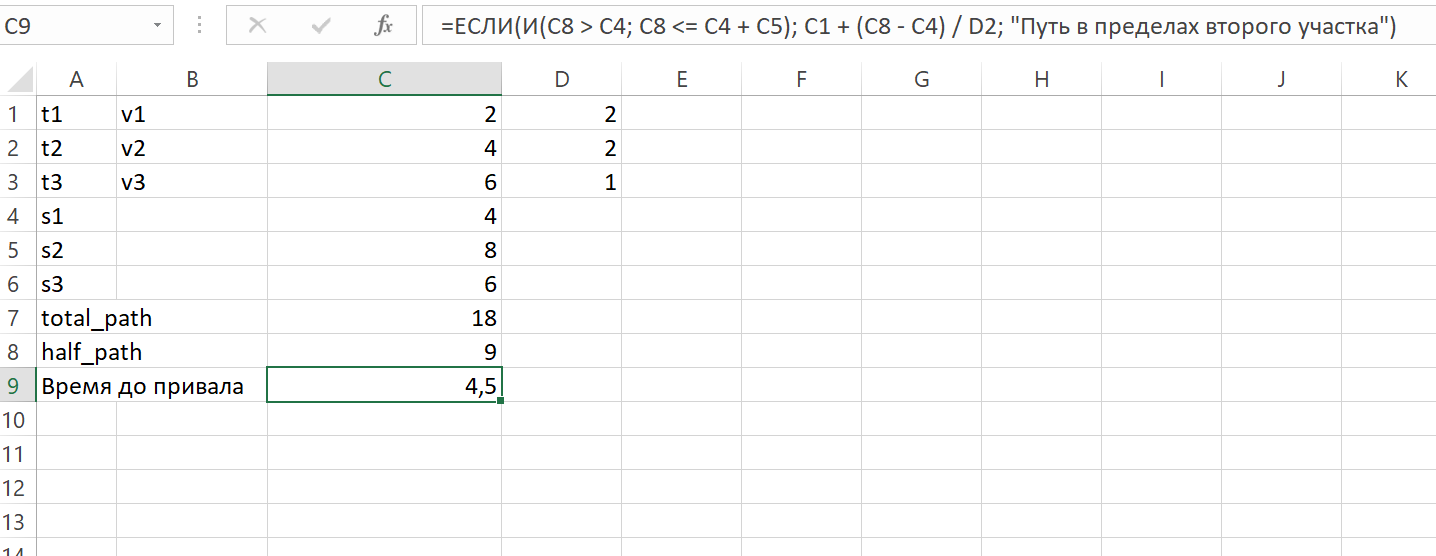


Рисунок 37 – Второй участок

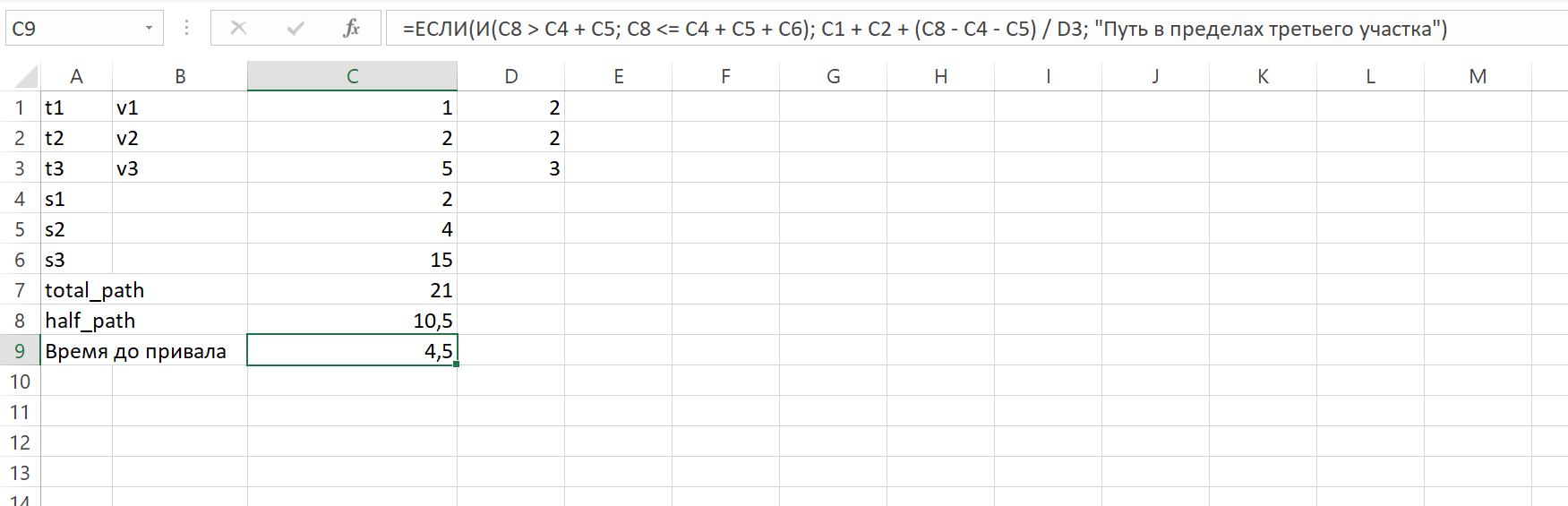


Рисунок 38 – Третий участок

* 1. Отметка о выполнение задания

Отметка о выполнение задания в GitHub (Рисунок 39).

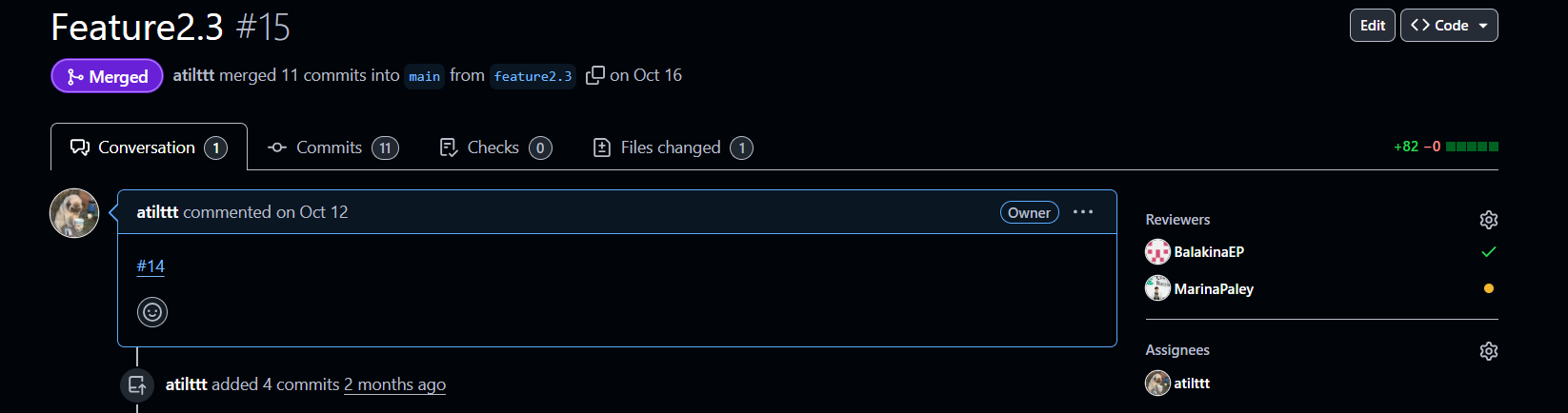


Рисунок 39 – Отметка о выполнение задания