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 14

Выполнил: ст. гр. ТКИ - 142

Кузнецов Денис Константинович

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

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

Москва 2024

Оглавление

[1. Задание 2.1 4](#_Toc185256759)

[1.1. Формулировка задания 4](#_Toc185256760)

[1.2. Блок-схема алгоритма 5](#_Toc185256761)

[1.3. Код задания 2.1 8](#_Toc185256762)

[1.4. Результаты выполнения программы 10](#_Toc185256763)

[1.5. Решение тестовых примеров 12](#_Toc185256764)

[12](#_Toc185256765)

[1.6. Зачёт задания в GitHub 12](#_Toc185256766)

[2. Задание 2.2 13](#_Toc185256767)

[2.1. Формулировка задания 13](#_Toc185256768)

[2.2. Блок-схема алгоритма 14](#_Toc185256769)

[2.3. Код задания 2.2 16](#_Toc185256770)

[2.4. Решение тестовых примеров 17](#_Toc185256771)

[2.5. Решение тестовых примеров в Excel 17](#_Toc185256772)

[2.6. Зачёт задания в GitHub 18](#_Toc185256773)

[3. Задание 2.3 19](#_Toc185256774)

[3.1. Формулировка задания 19](#_Toc185256775)

[3.2. Блок-схема алгоритма 20](#_Toc185256776)

[3.3. Код задания 2.3 24](#_Toc185256777)

[3.4. Решение тестового примера 26](#_Toc185256778)

[3.5. Решение тестовых примеров на Python 27](#_Toc185256779)

[27](#_Toc185256780)

[3.6. Зачёт задания в GitHub 28](#_Toc185256781)

# Задание 2.1

* 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.

Таблица  – Формулировка задания 2.1

|  |  |
| --- | --- |
| Вариант | Формулы |
| 14 | Даны числа *a*, *b*, *c*. Найти:   * количество отрицательных чисел; * абсолютное значение суммы этих чисел.. |

* 1. Блок-схема алгоритма

Блок-схема алгоритма представлена на рисунках (Рисунок 1, Рисунок *2*, Рисунок *3*, Рисунок *4*)

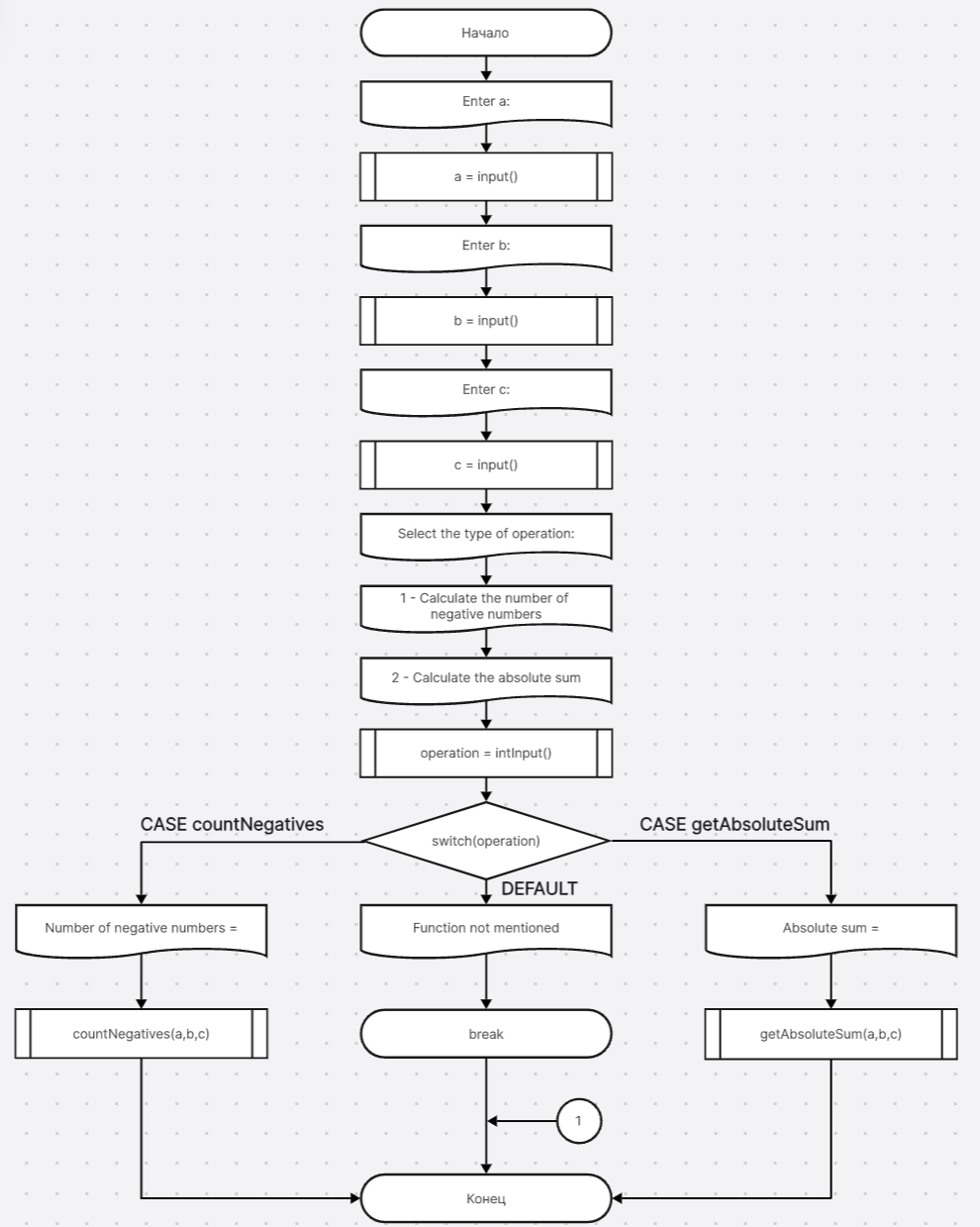


Рисунок 1 – Блок-схема основного алгоритма

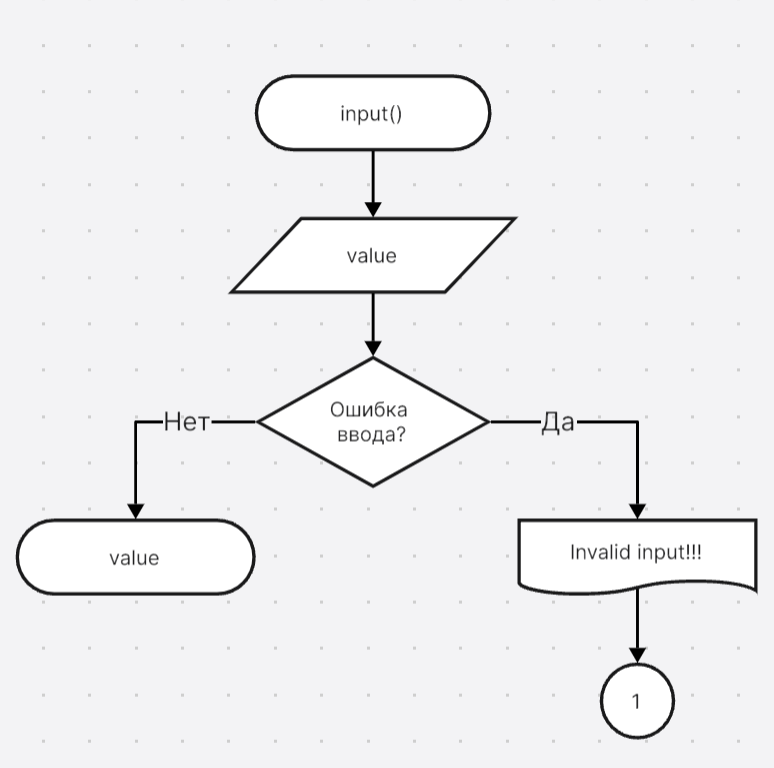


Рисунок 2 – Блок-схема функции input()

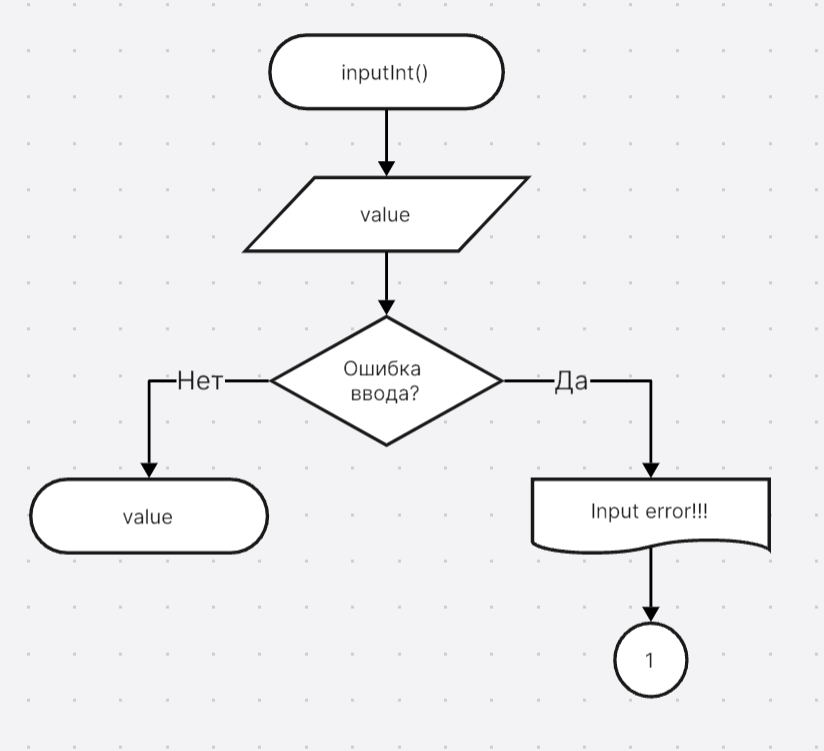


Рисунок 3 – Блок-схема функции inputInt()

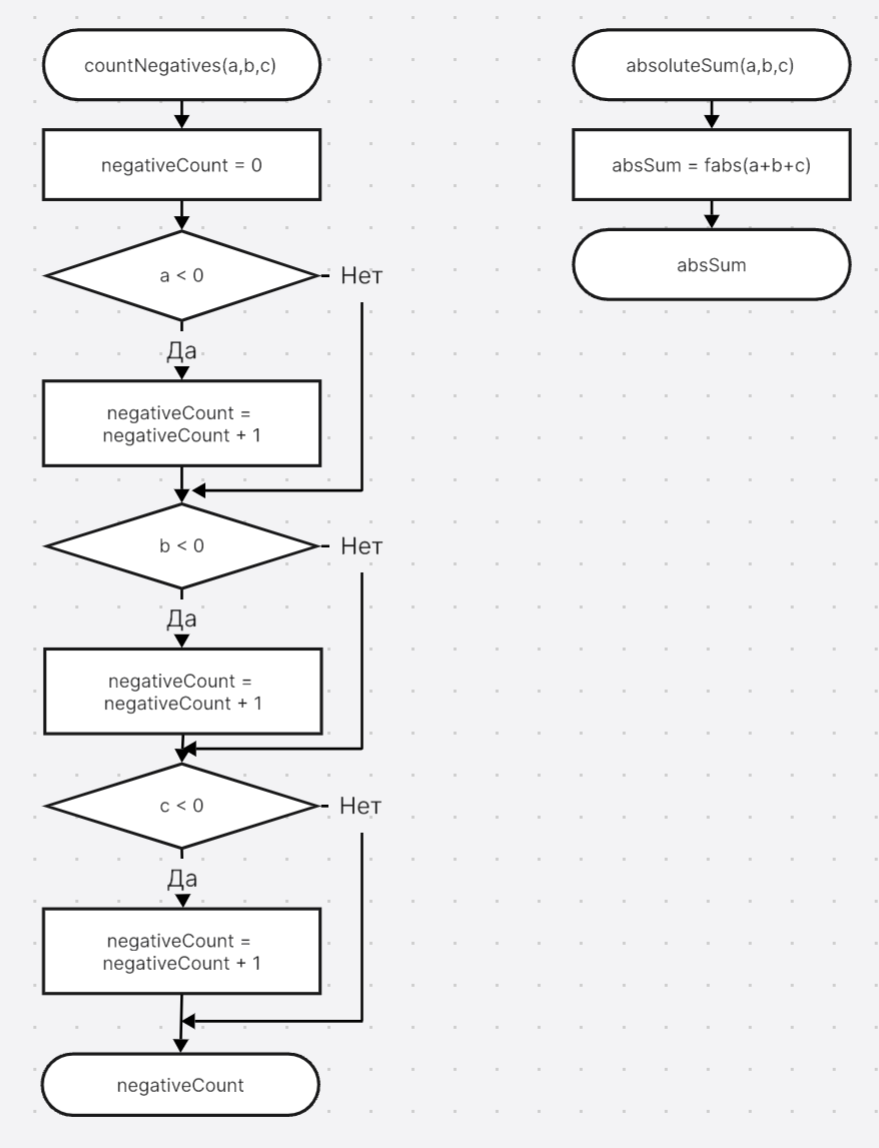


Рисунок 4 – Блок-схема функций countNegatives и absoluteSum

* 1. Код задания 2.1

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <errno.h>

/\*\*

\* @param getNegatives - выводит количество отрицательных чисел

\* @param getAbsoluteSum - выводит значение абсолютной суммы

\*/

typedef enum {

getNegatives = 1,

getAbsoluteSum = 2

};

/\*\*

\* @brief считает число отрицательных чисел

\* @param a значение параметра a

\* @param b значение параметра b

\* @param c значение параметра c

\* @return количество отрицательных чисел

\*/

int countNegatives(double const a, double const b, double const c);

/\*\*

\* @brief считает абсолютную сумму чисел

\* @param a значение параметра a

\* @param b значение параметра b

\* @param c значение параметра c

\* @return абсолютную сумму чисел, округленную до целого

\*/

int absoluteSum(double const a, double const b, double const c);

/\*\*

\* @brief Считывает вещественное число

\* @return возвращает вещественное число

\*/

double input(void);

/\*\*

\* @brief Ввод полученного целого значения

\* @return Возвращает полученное целое значение

\*/

int inputInt(void);

/\*\*

\* @brief точка входа в программу

\* @return 0 в случае успеха

\*/

int main(void) {

printf("Enter a: ");

const double a = input();

printf("Enter b: ");

const double b = input();

printf("Enter c: ");

const double c = input();

printf("Select the type of operation: \n%d - Calculate the number of negative numbers\n%d - Calculate the absolute sum\n", (int)getNegatives, (int)getAbsoluteSum);

int operation = inputInt();

switch (operation) {

case getNegatives: {

printf("Number of negative numbers = %d\n", countNegatives(a, b, c));

break;

}

case getAbsoluteSum: {

printf("Absolute sum = %d\n", absoluteSum(a, b, c));

break;

}

default: {

errno = ERANGE;

perror("Function not mentioned");

exit(EXIT\_FAILURE);

}

}

return 0;

}

double input() {

double value = 0.0;

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

if (result != 1) {

errno = EIO;

perror("Invalid input!!!");

exit(EXIT\_FAILURE);

}

return value;

}

int countNegatives(double const a, double const b, double const c) {

int negativeCount = 0;

if (a < 0) negativeCount++;

if (b < 0) negativeCount++;

if (c < 0) negativeCount++;

return negativeCount;

}

int absoluteSum(double const a, double const b, double const c) {

return fabs(a + b + c);

}

int inputInt(void) {

int i = 0;

int check = scanf\_s("%d", &i);

if (check != 1) {

errno = EIO;

perror("Input error!!!\n");

exit(EXIT\_FAILURE);

}

return i;

}

* 1. Результаты выполнения программы

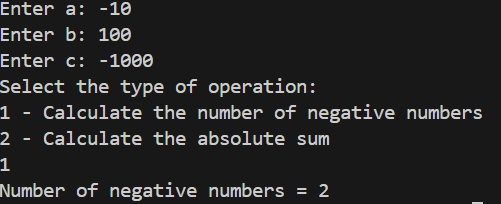


Рисунок 5 – Вывод количества отрицательных чисел

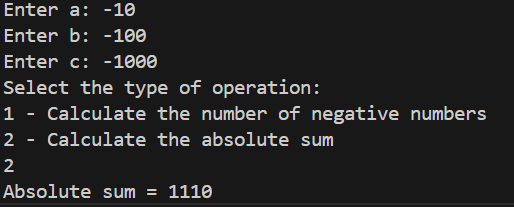


Рисунок 6 – Вывод абсолютной суммы

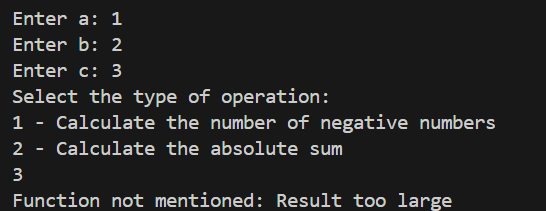


Рисунок 7 – Вывод, когда функция не определена

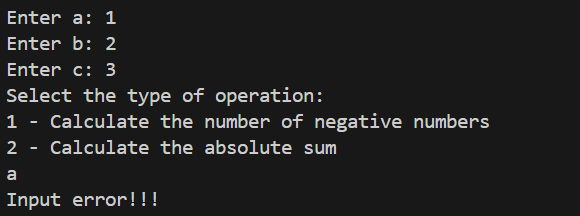


Рисунок 8 – Вывод, когда первый выбор – буква



Рисунок 9 – Вывод, когда число а является буквой

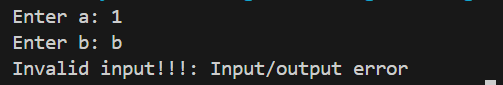


Рисунок 10 – Вывод, когда число b является буквой

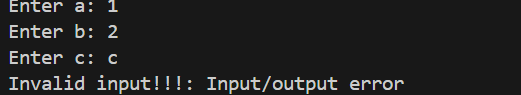


Рисунок 11 – Вывод, когда число b является буквой

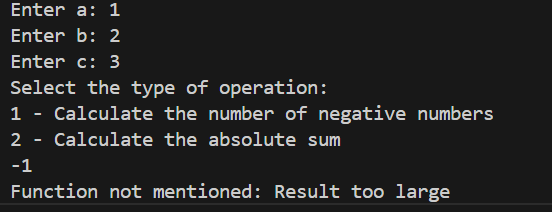


Рисунок 12 – Вывод, когда номер функции отрицательный

* 1. Решение тестовых примеров

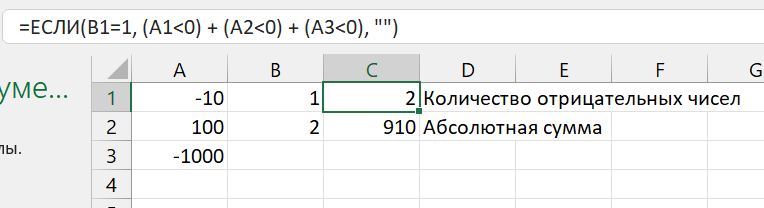


Рисунок 13 –Тестовый пример количества отрицательных чисел

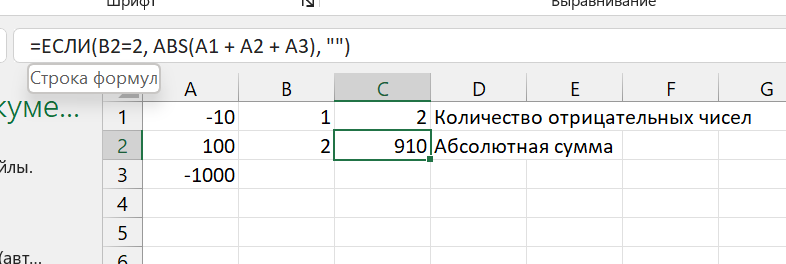


Рисунок 14 –Тестовый пример абсолютной суммы

* 1. Зачёт задания в GitHub

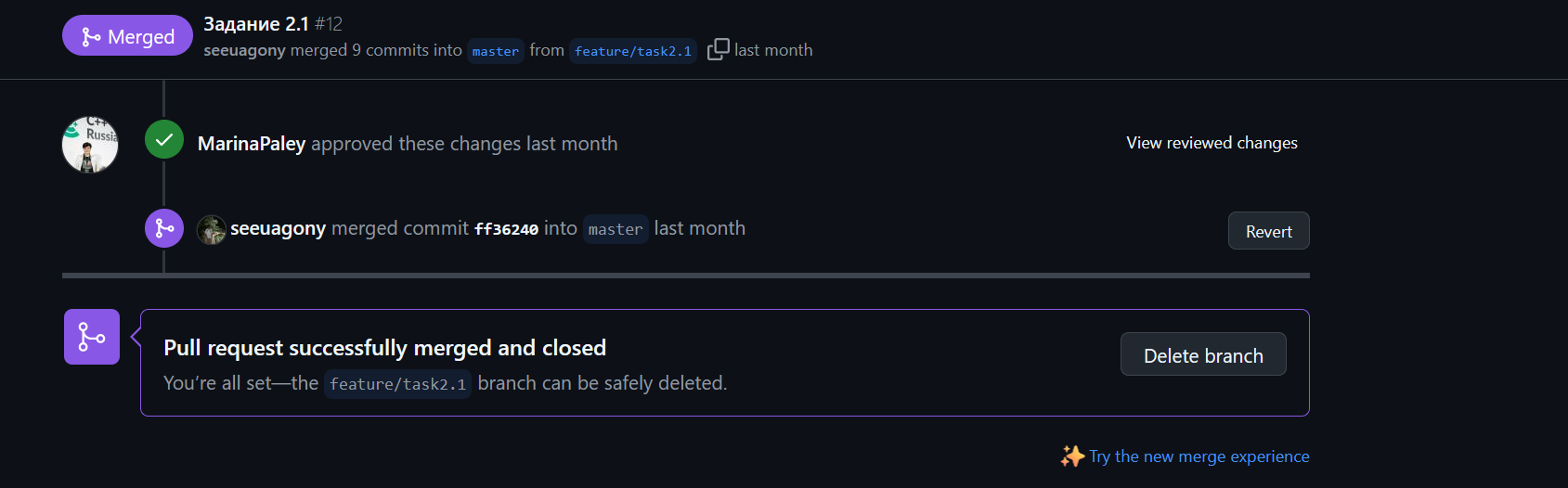


Рисунок 15 – Зачёт задания в GitHub

# Задание 2.2

* 1. 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.

|  |  |  |
| --- | --- | --- |
| Вариант | Функция | Константы |
| 14 |  |  |

Таблица 2 – Формулировка задания 2.2

* 1. Блок-схема алгоритма

Блок-схема алгоритма представлена на рисунках (Рисунок 16, Рисунок 17, Рисунок 18)

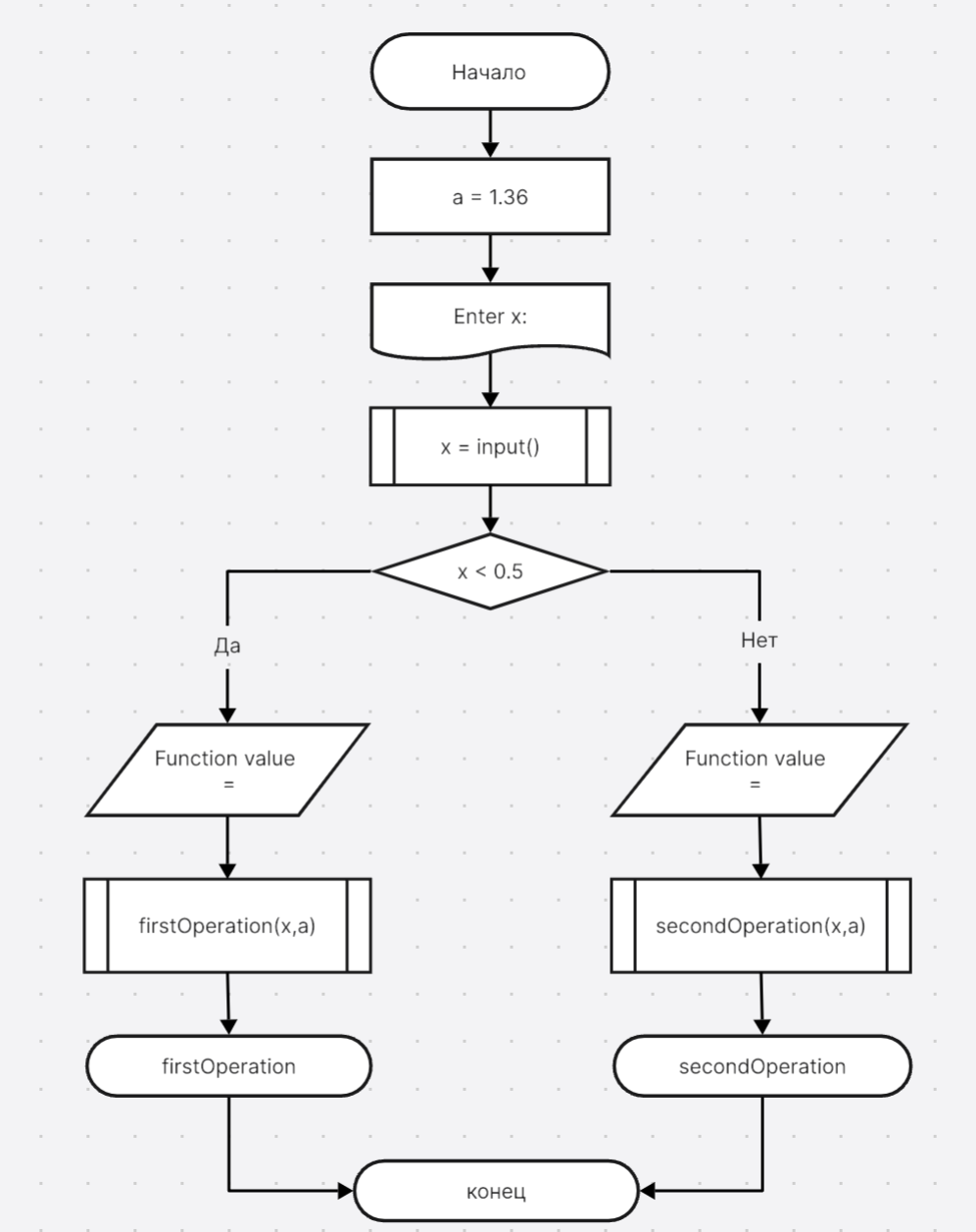


Рисунок 16 – Блок-схема основного алгоритма

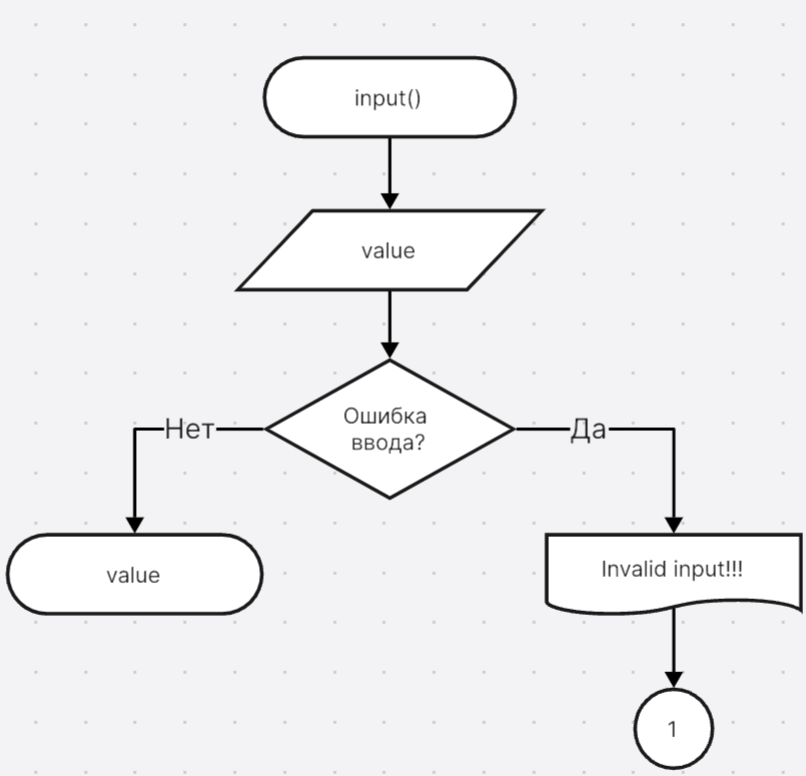


Рисунок 17 – Блок-схема функции input()

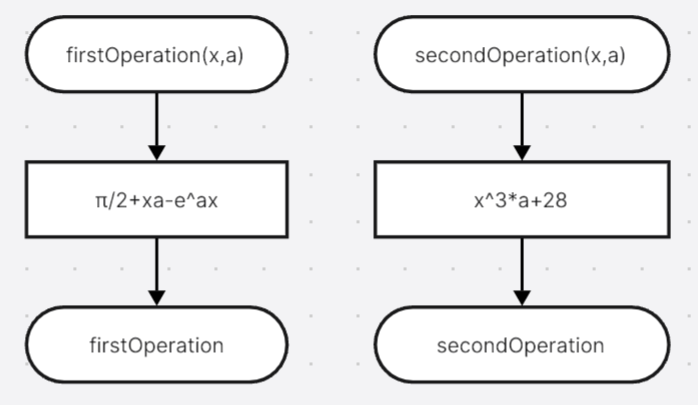


Рисунок 18 – Блок-схема функции firstOperation(x,a) и secondOperation(x,a)

* 1. Код задания 2.2

#define \_USE\_MATH\_DEFINES

#include <stdio.h>

#include <math.h>

#include <stdlib.h>

#include <errno.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 firstOperation(const double x, const double a);

/\*\*

\* @brief рассчитывает значение функции по формуле

\* @param x значение переменной x

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

\*/

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

/\*\*

\* @brief точка входа в программу

\* @return 0 в случае успеха

\*/

int main(void) {

const double a = 1.36;

printf("Enter x: ");

const double x = input();

if (x < 0.5) {

printf("Function value = %lf", firstOperation(x, a));

}

else {

printf("Function value = %lf", secondOperation(x, a));

}

return 0;

}

double input(void)

{

double value = 0.0;

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

if (result != 1) {

errno = EIO;

perror("Input error!!!\n");

exit(EXIT\_FAILURE);

}

return value;

}

double firstOperation(const double x, const double a) {

return ((M\_PI)/2 + x\*a - exp(a\*x));

}

double secondOperation(const double x, const double a) {

return (pow(x, 3)\*a + 28);

}

* 1. Решение тестовых примеров



Рисунок 19 – Вывод, когда переменная x меньше 0.5



Рисунок 20 – Вывод, когда переменная x больше 0.5



Рисунок 21 – Вывод, когда переменная x – буква

* 1. Решение тестовых примеров в Excel

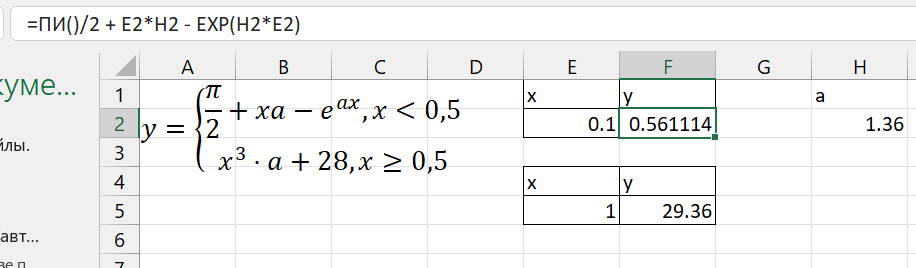


Рисунок 22 – Расчёт y, когда x, меньше 0.5

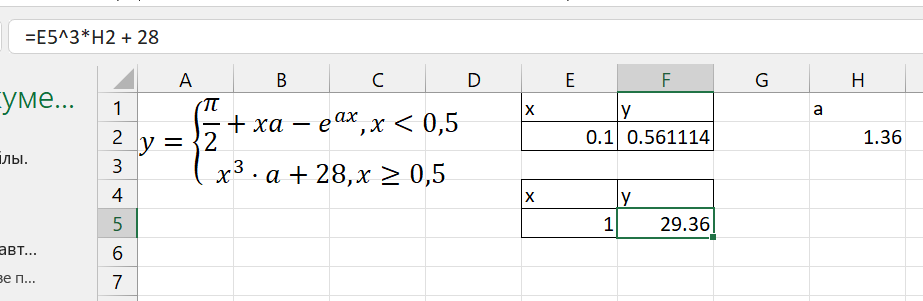


Рисунок 23 – Расчёт y, когда x больше либо равен 0.5

* 1. Зачёт задания в GitHub

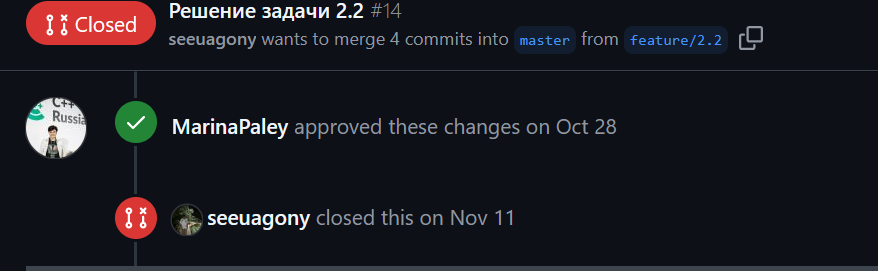


Рисунок 24 – Зачёт задания в GitHub

# Задание 2.3

* 1. 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.

Таблица 3 –Формулировка задания 2.3

|  |  |
| --- | --- |
| Вариант | Задача |
| 14 | Можно ли на прямоугольном участке застройки размером *aˣb* метров разместить два дома размером *pˣq* и *rˣs* метров? Дома можно располагать только параллельно сторонам участка. |

* 1. Блок-схема алгоритма

Блок-схема алгоритма представлена на рисунках (Рисунок 25, Рисунок 26, Рисунок 27)

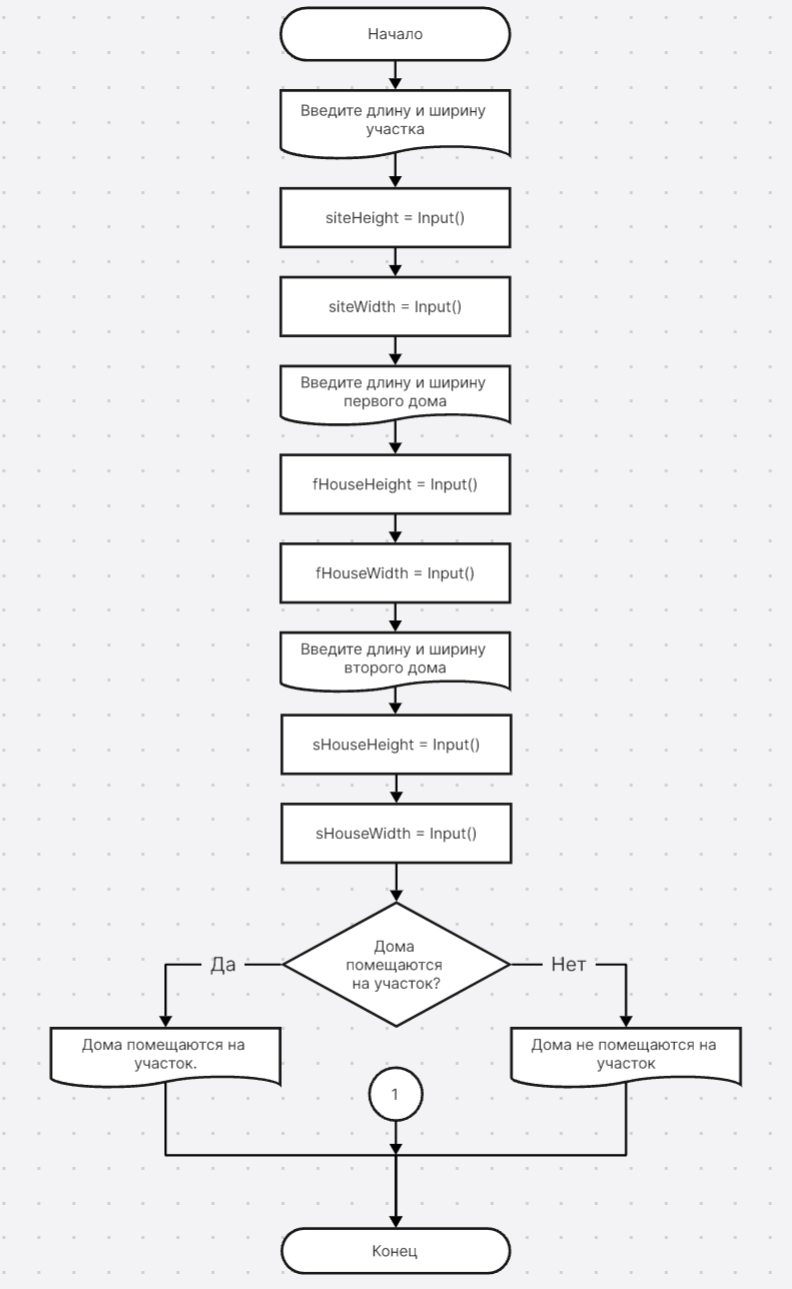


Рисунок 25 – Блок-схема основного алгоритма

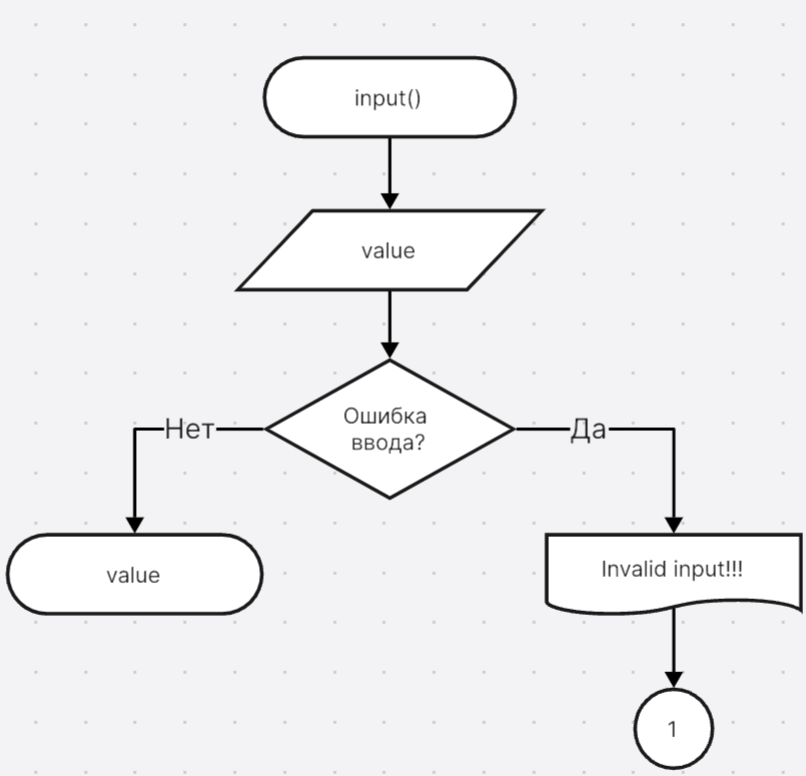


Рисунок 26 – блок-схема функции Input()

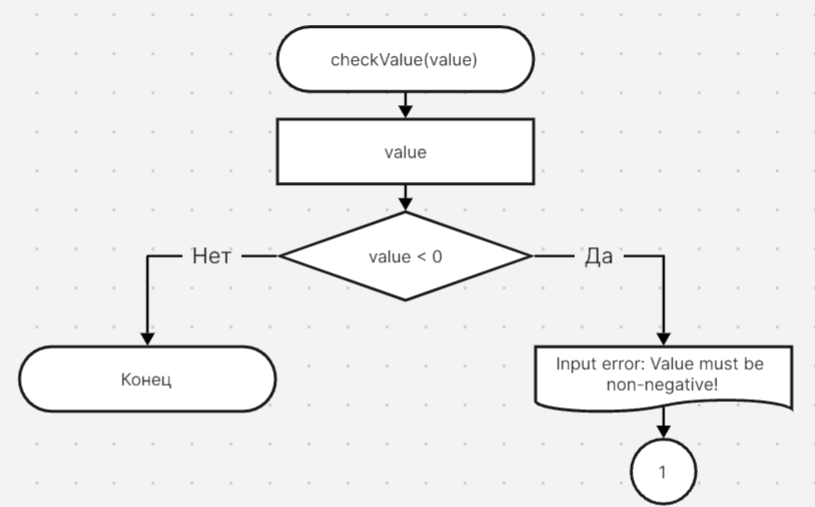


Рисунок 27 – Блок-схема функции checkValue(value)

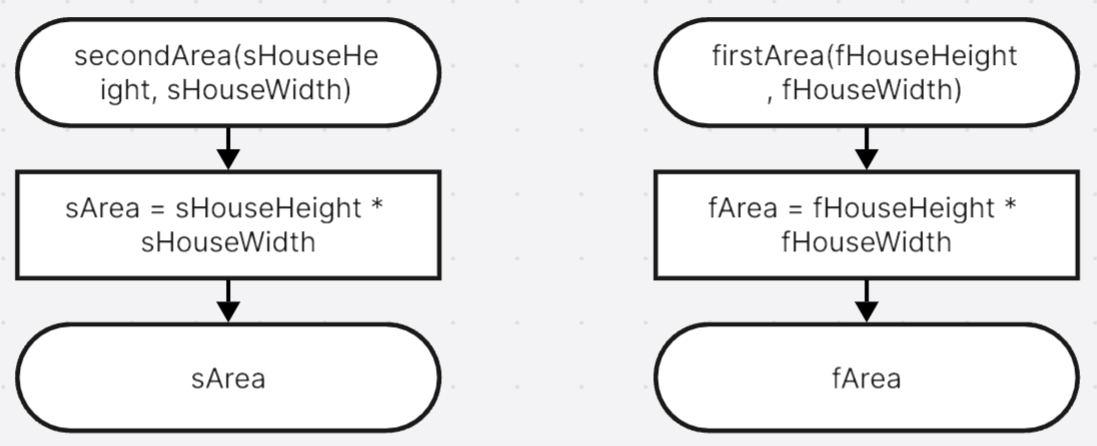


Рисунок 28 – Блок-схема функций firstArea и secondArea

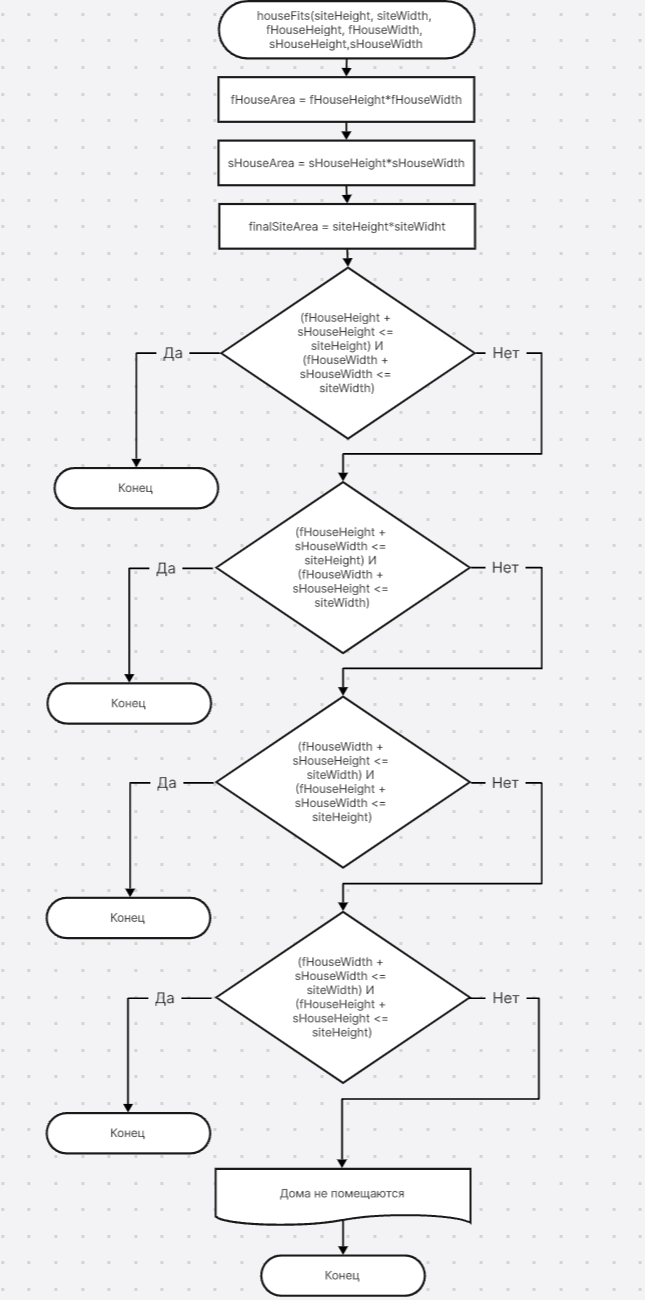


Рисунок 29 – Блок-схема функции houseFits

* 1. Код задания 2.3

#include <stdio.h>

#include <errno.h>

#include <stdlib.h>

#include <float.h>

#include <math.h>

#include <stdbool.h>

/\*\*

\* @brief проверка введенного значения

\* @return возвращает значение

\*/

double Input(void);

/\*\*

\* @brief Проверяет корректность введённого значения

\* @param value Значение для проверки

\* @return Возвращает ошибку в случае неверно введенного значения

\*/

void checkValue(double value);

/\*\*

\* @brief рассчитывает площадь участка

\* @param siteHeight значение переменной длины участка

\* @param siteWidth значение переменной ширины участка

\* @return возвращает площадь участка

\*/

double siteArea(const double siteHeight, const double siteWidth);

/\*\*

\* @brief рассчитывает площадь первого дома

\* @param fHouseHeight значение переменной длины дома

\* @param fHouseWidth значение переменной ширины дома

\* @return возвращает площадь первого дома

\*/

double firstArea(const double fHouseHeight, const double fHouseWidth);

/\*\*

\* @brief рассчитывает площадь первого дома

\* @param sHouseHeight значение переменной длины дома

\* @param sHouseWidth значение переменной ширины дома

\* @return возвращает площадь первого дома

\*/

double secondArea(const double sHouseHeight, const double sHouseWidth);

/\*\*

\* @brief рассчитывает площадь первого дома

\* @param sHouseHeight значение переменной длины дома

\* @param sHouseWidth значение переменной ширины дома

\* @return возвращает площадь первого дома

\*/

bool houseFits(const double siteHeight, const double siteWidth, const double fHouseHeight, const double fHouseWidth, const double sHouseHeight, const double sHouseWidth);

/\*\*

\* @brief точка входа в программу

\* @return 0 в случае успеха

\*/

int main(void) {

printf("Введите длину и ширину участка\n");

const double siteHeight = Input();

const double siteWidth = Input();

printf("Введите длину и ширину первого дома\n");

const double fHouseHeight = Input();

const double fHouseWidth = Input();

printf("Введите длину и ширину второго дома\n");

const double sHouseHeight = Input();

const double sHouseWidth = Input();

if (houseFits(siteHeight, siteWidth, fHouseHeight, fHouseWidth, sHouseHeight, sHouseWidth)) {

printf("Дома помещаются на участок.\n");

}

else {

printf("Дома не помещаются на участок.\n");

}

return 0;

}

double Input(void) {

double value = 0.0;

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

if (result != 1)

{

errno = EIO;

perror("Input error!");

exit(EXIT\_FAILURE);

}

checkValue(value);

return value;

}

void checkValue(double value) {

if (value < DBL\_EPSILON) {

errno = EIO;

perror("Input error: Value must be non-negative!");

exit(EXIT\_FAILURE);

}

}

double firstArea(const double fHouseHeight, const double fHouseWidth) {

return fHouseHeight \* fHouseWidth;

}

double secondArea(const double sHouseHeight, const double sHouseWidth) {

return sHouseHeight \* sHouseWidth;

}

double siteArea(const double siteHeight, const double siteWidth) {

return siteHeight \* siteWidth;

}

bool houseFits(const double siteHeight, const double siteWidth,

const double fHouseHeight, const double fHouseWidth,

const double sHouseHeight, const double sHouseWidth) {

if ((fHouseHeight + sHouseHeight <= siteHeight &&

fHouseWidth + sHouseWidth <= siteWidth) ||

(fHouseHeight + sHouseWidth <= siteHeight &&

fHouseWidth + sHouseHeight <= siteWidth) ||

(fHouseWidth + sHouseHeight <= siteWidth &&

fHouseHeight + sHouseWidth <= siteHeight) ||

(fHouseWidth + sHouseWidth <= siteWidth &&

fHouseHeight + sHouseHeight <= siteHeight)) {

return true;

}

return false;

}

* 1. Решение тестового примера

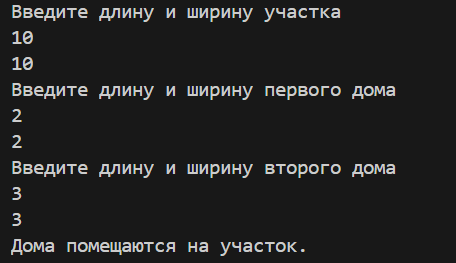


Рисунок 30 – Вывод проверки, помещается ли дом

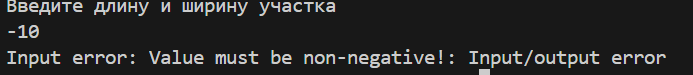


Рисунок 31 – Вывод программы, когда введённое значение отрицательное

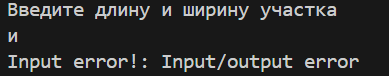


Рисунок 32 – Вывод программы, когда значение выбора – буква

* 1. Решение тестовых примеров на Python

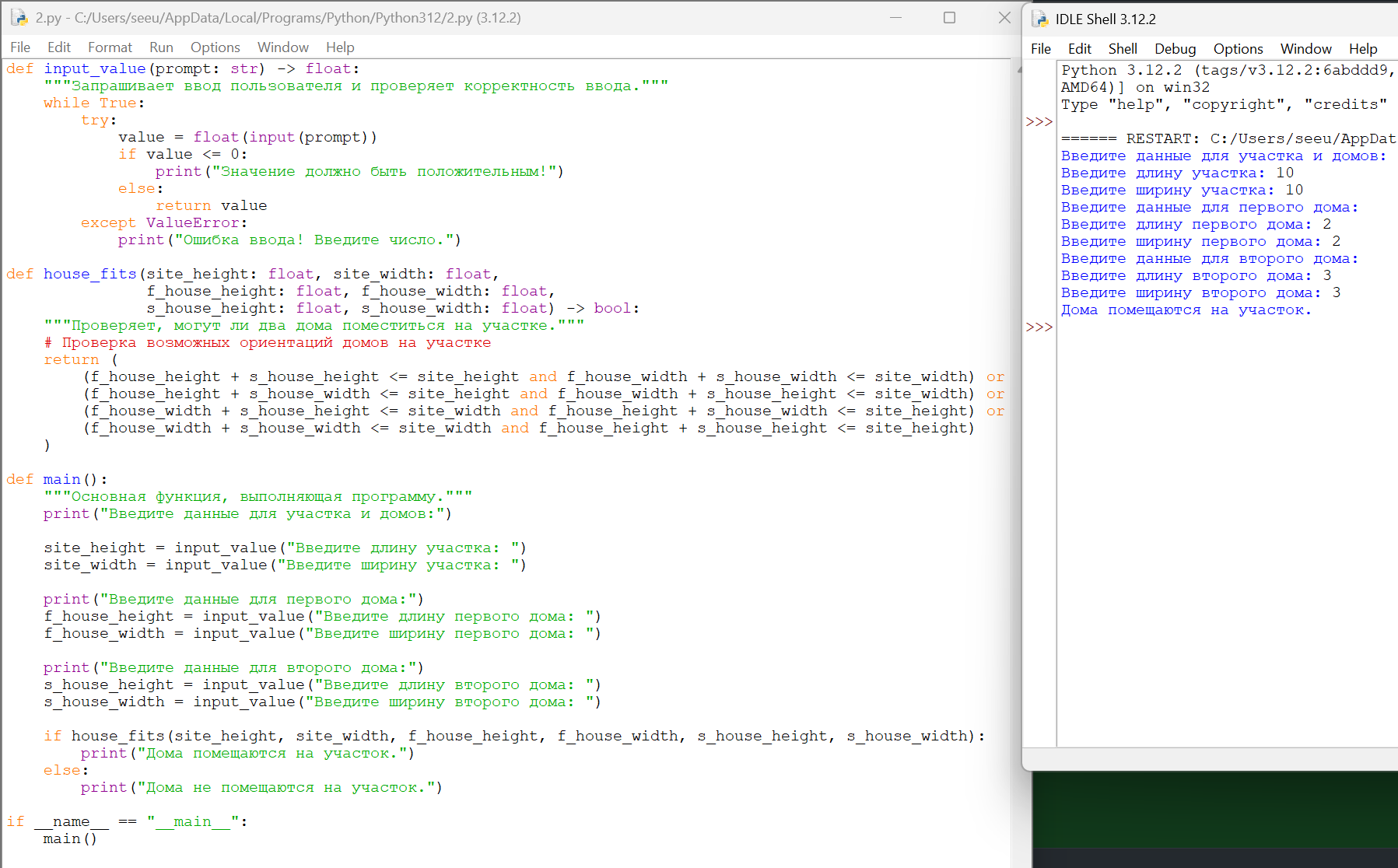


Рисунок 32 – Условие, когда дома помещаются

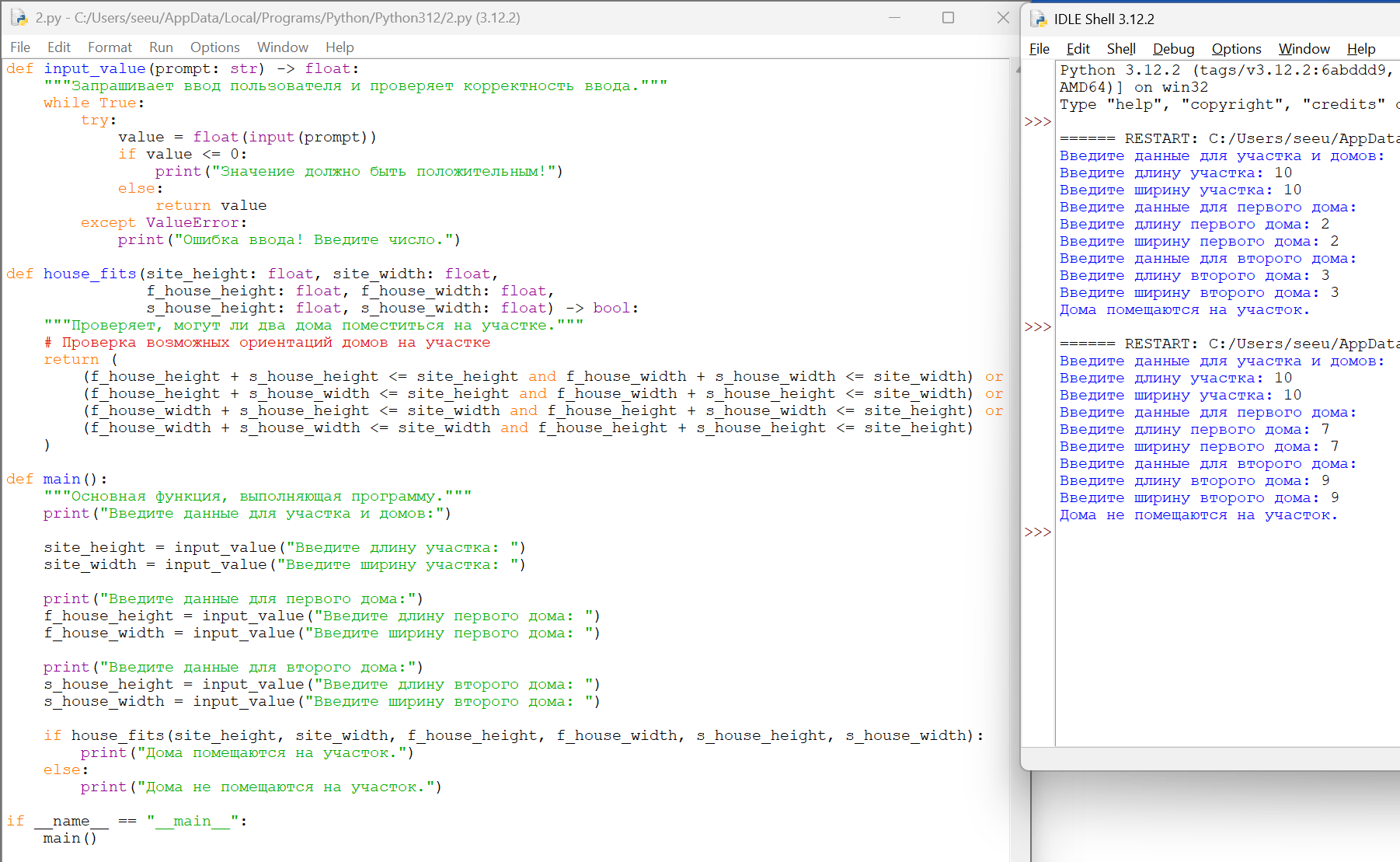


Рисунок 33 – Условие, когда дома не помещаются

* 1. Зачёт задания в GitHub

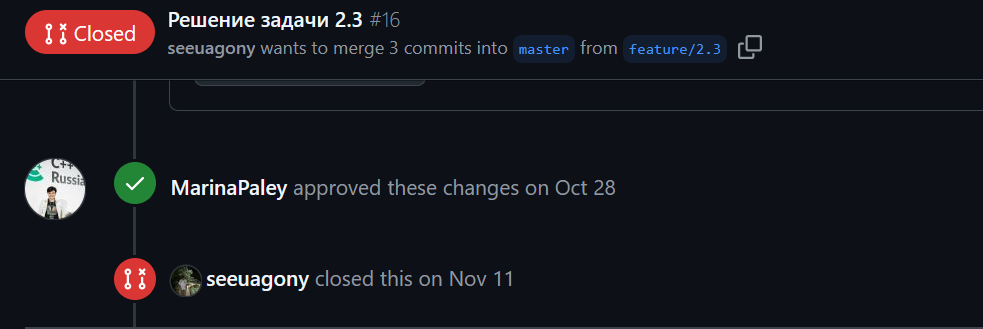


Рисунок 34 – Зачёт задания в GitHub