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 №1

По дисциплине «Введение в языки программирования»

Выполнил: ст. гр. ТКИ – 142

Плешанов И.С.

Проверил: к.т.н., доц.

Васильева М.А.

Москва 2024

Оглавление

[**Задание 2–1** 3](#_Toc91434532)

[**Условие задания** 3](#_Toc91434533)

[**Блок-схема алгоритма** 4](#_Toc91434534)

[**Программа на языке С** 5](#_Toc91434535)

[**Решение тестовых примеров на С** 6](#_Toc91434536)

[**Решение тестовых примеров в Python** 7](#_Toc91434537)

[**Задание 2–2** 8](#_Toc91434538)

[**Формулировка задания** 8](#_Toc91434539)

[**Блок-схема алгоритма** 9](#_Toc91434540)

[**Программа на языке С** 10](#_Toc91434541)

[**Решение тестовых примеров на С** 11](#_Toc91434542)

[**Решение тестовых примеров в** **Python** 12](#_Toc91434543)

[**Задание 2–3** 14](#_Toc91434545)

[**Формулировка задания** 14](#_Toc91434546)

[**Блок-схема алгоритма** 15](#_Toc91434547)

[**Программа на языке С** 16](#_Toc91434548)

[**Решение тестовых примеров на С** 17](#_Toc91434549)

[**Решение тестовых примеров в** **Python** 18](#_Toc91434550)

# **Задание 2–1**

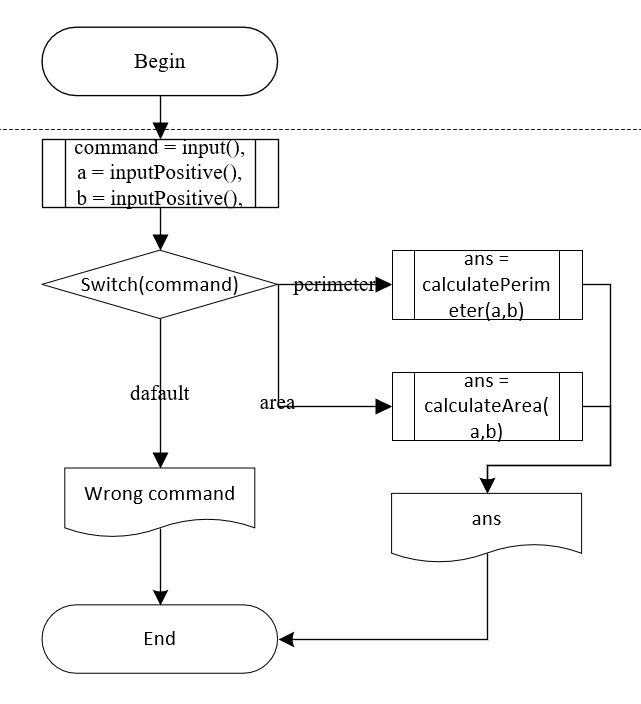
## **Условие задания**

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.

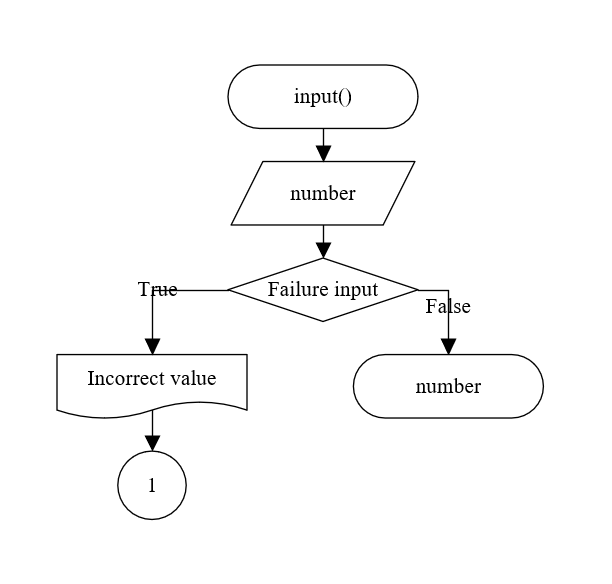
Таблица 1

|  |  |
| --- | --- |
| 2 | По заданным длинам двух катетов прямоугольного треугольника вычислить его:   * периметр; * площадь. |

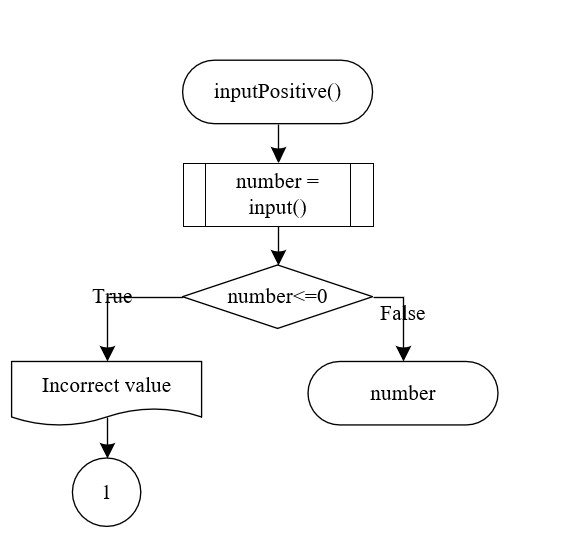
## **Блок-схема алгоритма**

****

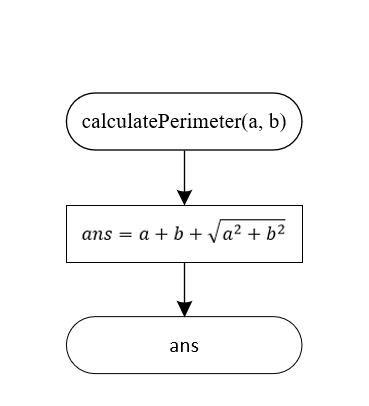
**Рисунок 1 – Блок-схема main 2-1**



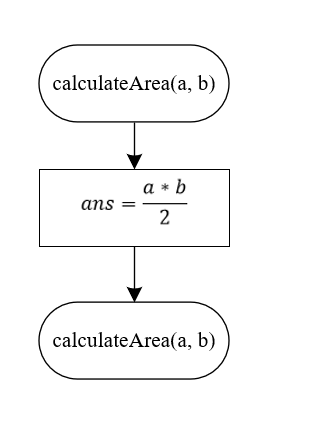
**Рисунок 2 – Блок схема inputDouble**



**Рисунок 3 - Блок схема inputPositiveDouble**

****

**Рисунок 4 – Блок схема calculatePerimeter**

****

**Рисунок 5 – Блок схема calculateArea**

## **Программа на языке С**

#include <stdio.h>

#include <math.h>

#include <stdlib.h>

#include <float.h>

#include <locale.h>

/\*

\* @brief функция ввода числа типа int из консоли с проверкой на корректность ввода.

\* @return Возвращает введенное из консоли число типа int.

\*/

С

/\*

\* @brief функция ввода числа типа double из консоли с проверкой на корректность ввода.

\* @return Возвращает введенное из консоли число типа double.

\*/

double inputDouble(void);

/\*

\* @brief функция ввода положительного числа типа double из консоли с проверкой на корректность ввода.

\* @return Возвращает введенное из консоли положительное число типа double.

\*/

double inputPositiveDouble(void);

/\*

\* @brief функция расчета периметра прямоугольного треугольника.

\* @param a длина первого катета.

\* @param b длина второго катета.

\* @return Возвращает значение периметра.

\*/

double calculatePerimeter(const double a, const double b);

/\*

\* @brief функция расчета площади прямоугольного треугольника.

\* @param a длина первого катета.

\* @param b длина второго катета.

\* @return Возвращает значение площади.

\*/

double calculateArea(const double a, const double b);

/\*

\* @brief перечисление команд вычисления параметров треугольника.

\* @param perimeter команда для вычисления периметра.

\* @param area команда для вычисления площади.

\*/

enum Triangle {

perimeter = 1,

area

};

/\*

\* @brief Точка входа в программу

\* @return Возвращает код ошибки

\*/

int main(void) {

setlocale(LC\_ALL, "");

printf("Введите номер команды:\n\

%d - рассчитать периметр треугольника\n\

%d - рассчитать площадь треугольника\n", perimeter, area);

const int command = inputInt();

printf("Введите длину первого катета: ");

const double a = inputPositiveDouble();

printf("Введите длину второго катета: ");

const double b = inputPositiveDouble();

switch (command) {

case perimeter:

printf("Периметр треугольника: %lf\n", calculatePerimeter(a, b));

break;

case area:

printf("Площадь треугольника: %lf\n", calculateArea(a, b));

break;

default:

printf("Ошибка ввода (неправильная команда)");

return 1;

}

return 0;

}

double calculatePerimeter(const double a, const double b) {

double c = sqrt(a \* a + b \* b);

return a + b + c;

}

double calculateArea(const double a, const double b) {

return (a \* b) / 2;

}

int inputInt(void) {

int a = 0;

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

printf("Ошибка ввода\n");

exit(EXIT\_FAILURE);

}

return a;

}

double inputDouble(void) {

double a = 0;

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

printf("Ошибка ввода\n");

exit(EXIT\_FAILURE);

}

return a;

}

double inputPositiveDouble(void) {

double value = inputDouble();

if (value <= DBL\_EPSILON) {

printf("Ошибка ввода: число должно быть положительным\n");

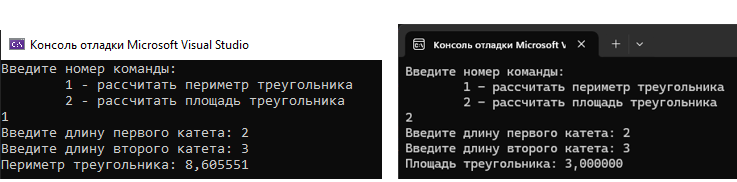
exit(EXIT\_FAILURE);

}

return value;

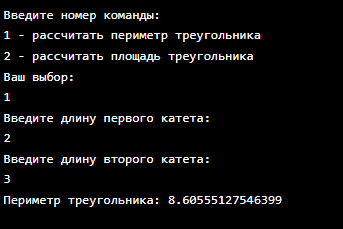
}

## **Решение тестовых примеров на С**



**Рисунок 6 – Решение тестового примера для задания 2-1 на С**

## **Решение тестовых примеров в Python**



**Рисунок 7 - Решение тестового примера задания 2-1 в Python**

# **Задание 2–2**

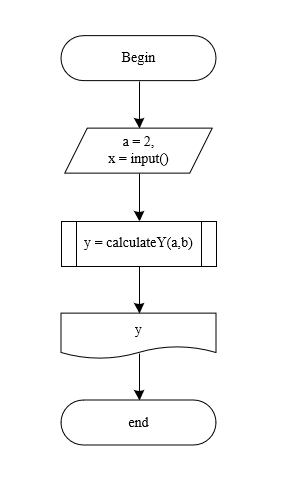
## **Формулировка задания**

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.

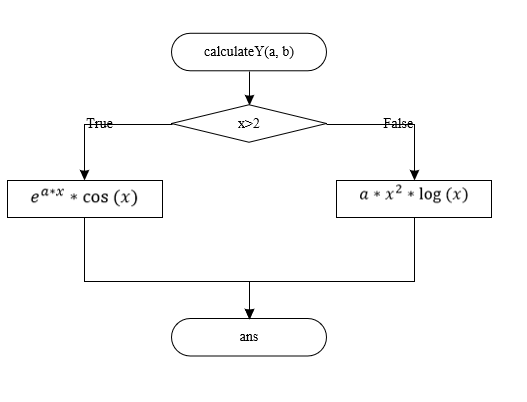
Таблица 2

|  |  |  |
| --- | --- | --- |
| 2 |  |  |

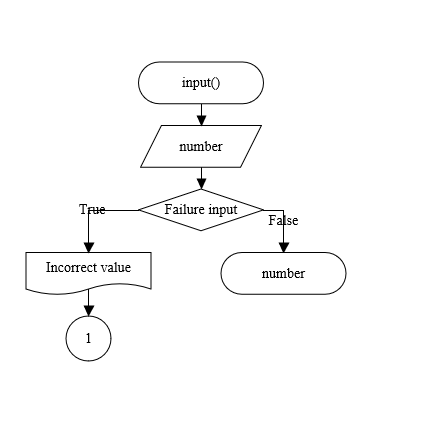
## **Блок-схема алгоритма**



**Рисунок 8 – Блок-схема main 2-2**



**Рисунок 9 – Блок схема calculate**



**Рисунок 10 – Блок схема input**

## **Программа на языке С**

#define \_USE\_MATH\_DEFINES

#include <stdio.h>

#include <math.h>

#include <stdlib.h>

#include <locale.h>

/\*

\* @brief Функция для ввода числа типа double с консоли.

\* @return Возвращает введённое пользователем число типа double.

\*/

double inputDouble(void);

/\*

\* @brief Функция для вычисления значения функции y в зависимости от аргумента x и параметра a.

\* @param x Значение аргумента функции.

\* @param a Параметр a, используемый в вычислениях.

\* @return Значение y, вычисленное на основе введённых x и a.

\*/

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

/\*

\* @brief Главная функция программы.

\* В этой функции выполняется ввод пользователя, проводится вычисление

\* на основе указанной функции и выводится результат.

\* Программа запрашивает значение x у пользователя, использует функцию

\* calculateY для выполнения вычислений и выводит результат на экран.

\* @return Возвращает 0 в случае успешного выполнения программы.

\*/

int main(void) {

setlocale(LC\_ALL, "");

const double a = 2.0;

printf("Введите значение x: ");

const double x = inputDouble();

const double y = calculateY(x, a);

printf("Результат y = %lf\n", y);

return 0;

}

double inputDouble(void) {

double number = 0;

while (scanf("%lf", &number) != 1) {

printf("Неверный ввод. Пожалуйста, введите число: ");

while (getchar() != '\n');

}

return number;

}

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

if (x > 2) {

return exp(a \* x) \* cos(x);

}

else if (x < 1) {

return 1;

}

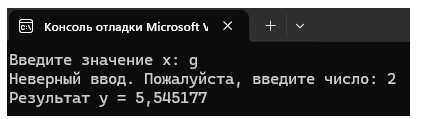
else {

return a \* pow(x, 2) \* log(x);

}

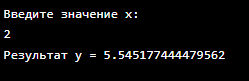
}

## **Решение тестовых примеров на С**



**Рисунок 11 – Решение тестового примера для задания 2-2 на С**

## **Решение тестовых примеров в Python**



**Рисунок 12 – Решение тестового примера для задания 2-2 в Python**

# **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.

Таблица 3

|  |  |
| --- | --- |
| 2 | Пройдет ли кирпич со сторонами *x, y, z* сквозь прямоугольное отверстие со сторонами *r* и *s*. Стороны отверстия должны быть параллельны граням кирпича. |

## **Блок-схема алгоритма**

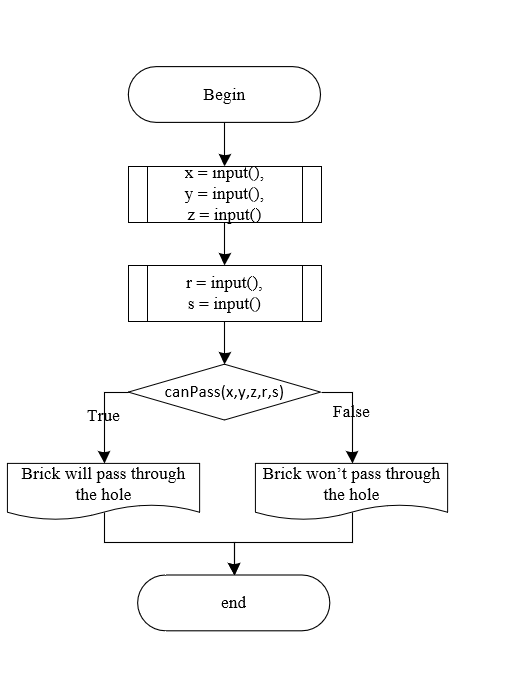


Рисунок 13 – Блок-схема main для задания 2-3

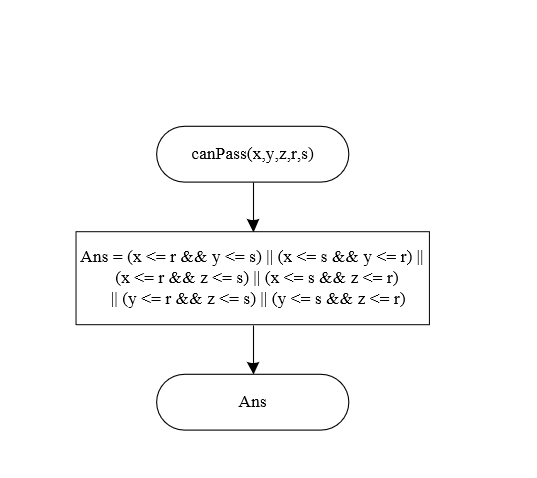


Рисунок 14 – Блок схема canPass

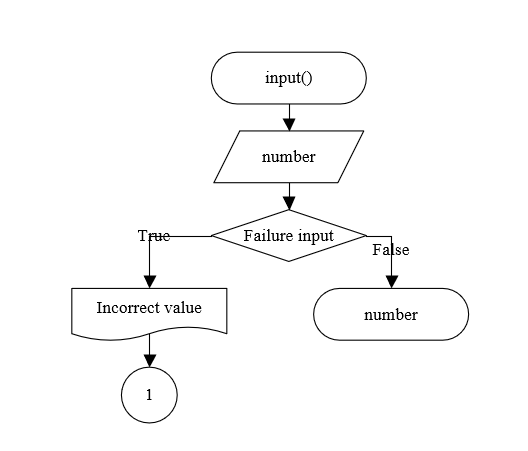


Рисунок 15 – Блок схема inputDouble

## **Программа на языке С**

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

#include <stdbool.h>

#include <float.h>

/\*\*

\* @brief Главная функция программы.

\* Программа запрашивает размеры кирпича и отверстия, и определяет,

\* сможет ли кирпич пройти через отверстие.

\* @return Возвращает 0 в случае успешного выполнения программы.

\*/

double inputDouble(void);

/\*\*

\* @brief Функция для проверки возможности прохождения кирпича через отверстие.

\* @return Возвращает true, если кирпич может пройти через отверстие, иначе false.

\*/

bool canPass(double x, double y, double z, double r, double s);

/\*\*

\* @brief Главная функция программы.

\* Программа запрашивает размеры кирпича (x, y, z) и отверстия (r, s),

\* после чего определяет, может ли кирпич пройти через отверстие.

\* Пользователь вводит размеры с консоли.

\* @return Возвращает 0 в случае успешного выполнения программы.

\*/

int main(void) {

setlocale(LC\_ALL, "");

printf("Введите размеры кирпича (x, y, z): \n");

printf("x: ");

double x = inputDouble();

printf("y: ");

double y = inputDouble();

printf("z: ");

double z = inputDouble();

printf("Введите размеры отверстия (r, s): \n");

printf("r: ");

double r = inputDouble();

printf("s: ");

double s = inputDouble();

if (canPass(x, y, z, r, s)) {

printf("Кирпич пройдет через отверстие.\n");

}

else {

printf("Кирпич не пройдет через отверстие.\n");

}

return 0;

}

/\*\*

\* @brief Функция для ввода числа типа double с консоли с проверкой корректности.

\* @return Возвращает введённое пользователем положительное число типа double.

\*/

double inputDouble(void) {

double number = 0.0;

while (scanf("%lf", &number) != 1 || number <= DBL\_EPSILON) {

printf("Ошибка ввода. Пожалуйста, введите положительное число: ");

while (getchar() != '\n');

}

return number;

}

bool canPass(double x, double y, double z, double r, double s) {

return (x <= r && y <= s) || (x <= s && y <= r) ||

(x <= r && z <= s) || (x <= s && z <= r) ||

(y <= r && z <= s) || (y <= s && z <= r);

}

## **Решение тестовых примеров на С**

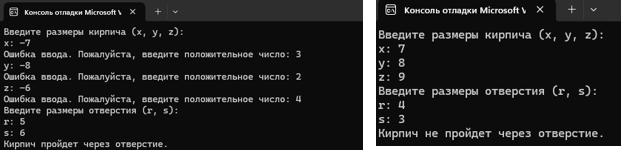


Рисунок 16 – Решение тестового примера для задания 2-3 на С

## **Решение тестовых примеров в Python**

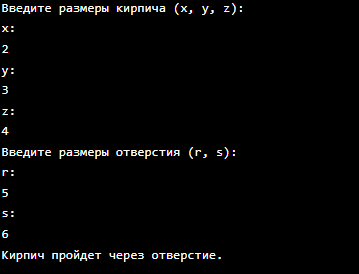


Рисунок 17 – Решение тестового примера для задания 2-3 в Python