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

Горковец Анна Сергеевна

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

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

Москва 2023

**Решение задачи 2.1**

**Формулировка задачи**

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 – Условие задачи 1.1

|  |  |
| --- | --- |
| **Вариант** | **Задание** |
| **5** | Заданы стороны прямоугольника. Определить его:  • периметр;  • площадь;  • длину диагонали. |

**Блок-схема алгоритма**

Блок-схема алгоритма представлена ниже (Рисунок 1, Рисунок 2, Рисунок 3, Рисунок 4, Рисунок 5, Рисунок 6, Рисунок 7)

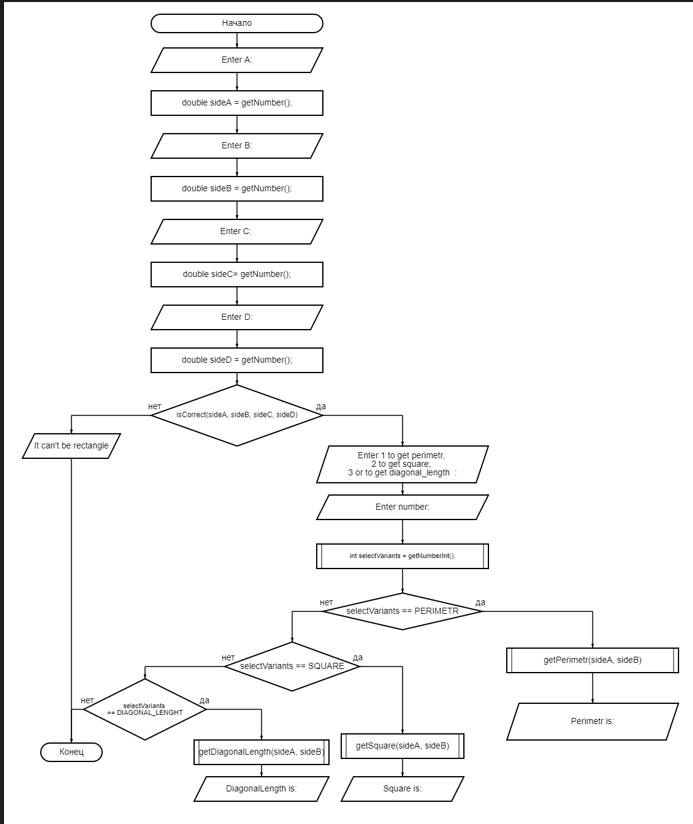


Рисунок 1 –Блок-схема функции main()

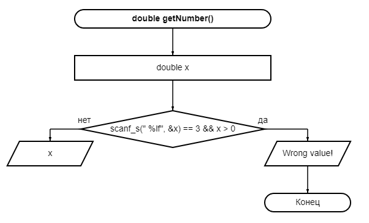


Рисунок 2 – Блок-схема функции getNumber()

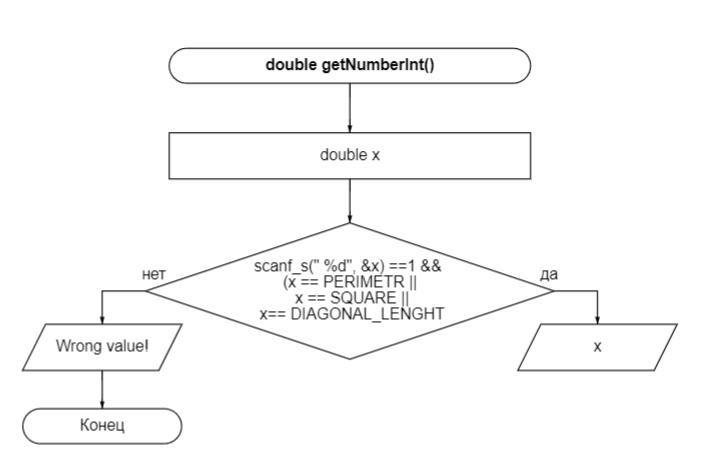


Рисунок 3 – Блок-схема функции getNumberInt()

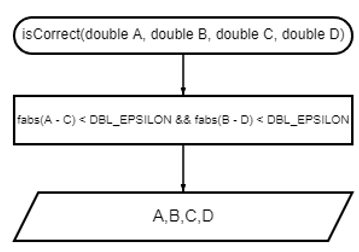


Рисунок 4 – Блок-схема функции isCorrect ()

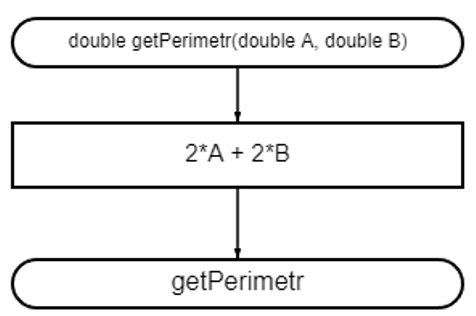


Рисунок 5 – Блок-схема функции getPerimetr ()

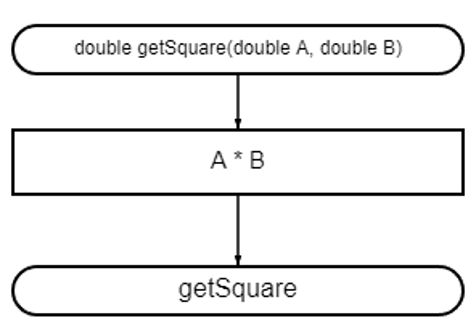


Рисунок 6 – Блок-схема функции getSquare ()

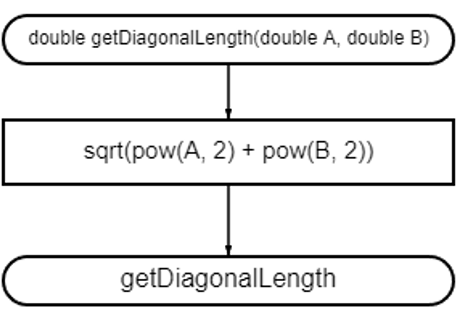


Рисунок 7 – Блок-схема функции getDiagonalLength()

**Код для задания 2.1**

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <float.h>

#include <stdbool.h>

enum Variants

{

PERIMETR = 1,

SQUARE = 2,

DIAGONAL\_LENGHT=3

};

/\*\*

\* @brief считывает и проверяет ввод переменной типа double

\* @return Возвращает считанное значение

\*/

double getNumber();

/\*\*

\* @brief проверяет возможность построения прямоугольника

\* @param A -первая сторона прямоугольника

\* @param B -вторая сторона прямоугольника

\* @param С -третья сторона прямоугольника

\* @param D -четвертая сторона прямоугольника

\* @return Возвращает 1, если прямоугольника существует, иначе 0

\*/

bool isCorrect(double A, double B, double C, double D);

/\*\*

\* @brief рассчитывает периметр прямоугольника

\* @param A -первая сторона прямоугольника

\* @param B -вторая сторона прямоугольника

\* @return Возвращает периметр прямоугольника

\*/

double getPerimetr(double A, double B);

/\*\*

\* @brief рассчитывает площадь прямоугольника

\* @param A -первая сторона прямоугольника

\* @param B -вторая сторона прямоугольника

\* @return Возвращает площадь прямоугольника

\*/

double getSquare(double A, double B);

/\*\*

\* @brief рассчитывает длину диагонали прямоугольника

\* @param A -первая сторона прямоугольника

\* @param B -вторая сторона прямоугольника

\* @return Возвращает площадь прямоугольника

\*/

double getDiagonalLength(double A, double B);

/\*\*

\* @brief Точка входа в программу

\* @return Возвращает 0, если программа работает правильно, иначе 1

\*/

int main()

{

printf\_s("Enter A: ");

double sideA = getNumber();

printf\_s("Enter B: ");

double sideB = getNumber();

printf\_s("Enter C: ");

double sideC = getNumber();

printf\_s("Enter D: ");

double sideD = getNumber();

if (isCorrect(sideA, sideB, sideC, sideD))

{

printf\_s("Enter %d to get perimetr, %d to get square, %d or to get diagonal\_length : ", PERIMETR, SQUARE,DIAGONAL\_LENGHT);

int selectVariants = getNumberInt();

if (selectVariants == PERIMETR)

{

printf\_s("Perimetr is %lf", getPerimetr(sideA, sideB));

}

else if (selectVariants == SQUARE)

{

printf\_s("Square is %lf", getSquare(sideA, sideB));

}

else if (selectVariants == DIAGONAL\_LENGHT)

{

printf\_s("DiagonalLength is %lf", getDiagonalLength(sideA, sideB));

}

}

else

{

printf\_s("It can't be rectangle");

}

return 0;

}

double getNumber()

{

double x;

if (scanf\_s(" %lf", &x) == 3 && x > 0)

{

printf\_s("Wrong value");

abort();

}

return x;

}

int getNumberInt()

{

int x;

if (scanf\_s(" %d", &x) ==1 && (x == PERIMETR || x == SQUARE || x== DIAGONAL\_LENGHT))

{

return x;

}

printf\_s("Wrong value");

abort();

}

bool isCorrect(double A, double B, double C, double D)

{

return fabs(A - C) < DBL\_EPSILON && fabs(B - D) < DBL\_EPSILON;

}

double getPerimetr(double A, double B)

{

return 2\*A + 2\*B ;

}

double getSquare(double A, double B)

{

return A \* B;

}

double getDiagonalLength(double A, double B)

{

return sqrt(pow(A, 2) + pow(B, 2));

}

# Решение различных тестовых примеров на С

Ниже представлен вывод программы 2.1 в С (Рисунок 8,Рисунок 9, Рисунок 10, Рисунок 11)

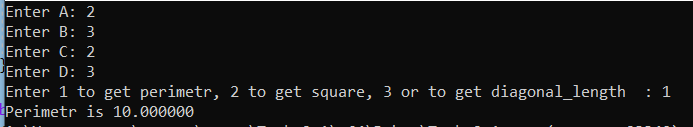


Рисунок 8 - Вывод периметра прямоугольника

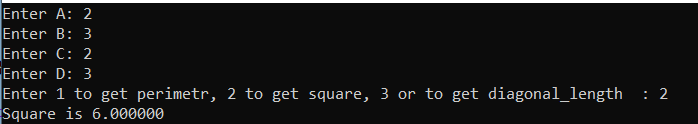


Рисунок 9 - Вывод площади прямоугольника

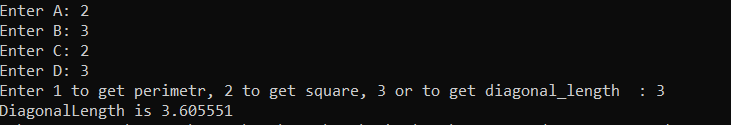


Рисунок 10 - Вывод длины диагонали прямоугольника



Рисунок 11 – Вывод программы 2.1, если значение задано буквой

**Решение различных тестовых** примеров в Excel

Ниже представлено решение задания 2.1 в Excel (Рисунок 12)



Рисунок 12 - Решение задания 2.1 в Excel

# Подтверждение approve 2.1

Ниже представлено доказательство того, что задание 2.1 было принято. (Рисунок 13)

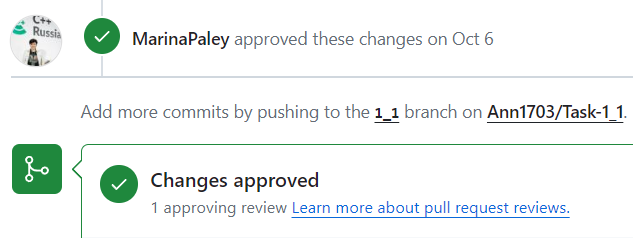


Рисунок 13 – Approve задачи 2.1

**Решение задачи 2.2**

**Формулировка задачи**

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 – Условие задачи 2.2

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

# Блок-схема алгоритма

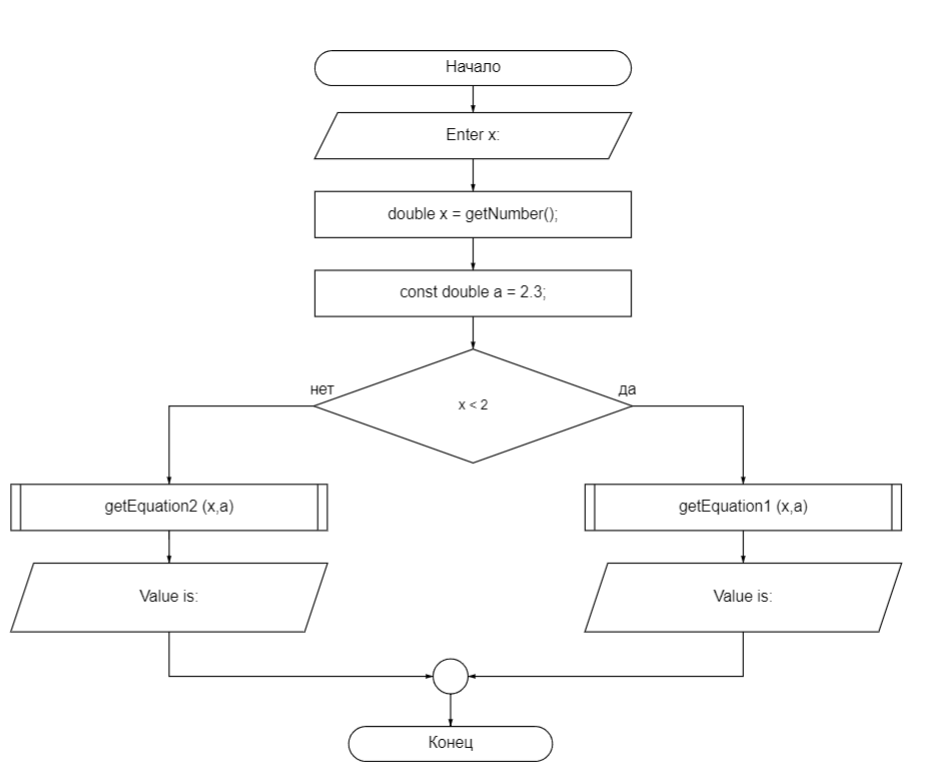
Ниже представлена блок-схема алгоритма (Рисунок 14, Рисунок 15) 

Рисунок 14 – Блок-схема функции main()

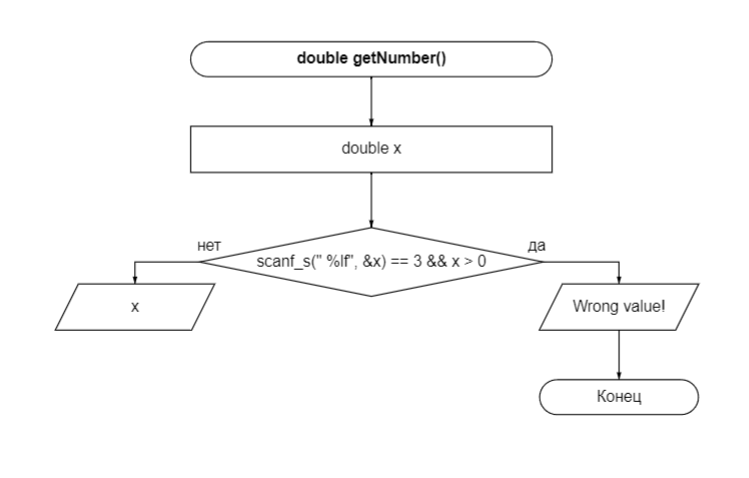


Рисунок 15 – Блок-схема функции get\_number ()

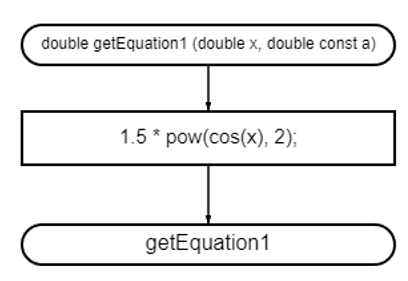


Рисунок 16 – Блок-схема функции getEquation1()

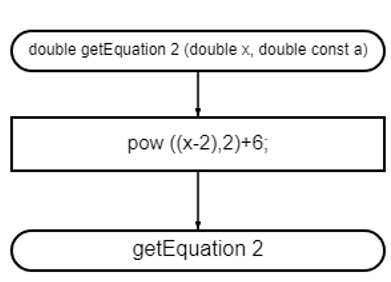


Рисунок 16 – Блок-схема функции getEquation2()

**Код для задания 2.2**

#include<stdio.h>

#include <stdlib.h>

#include<math.h>

/\*\*

\* @brief считывает и проверяет ввод переменной типа double

\* @return Возвращает считанное значение

\*/

double getNumber();

/\*\*

\*\brief функция выполняет вычисление при x < 2

\*\param x - число, вводимое пользователем

\*\return возвращает значение функции

\*\*/

double getEquation1(double x, double const a);

/\*\*

\*\brief функция выполняет вычисление при x >= 2

\*\param x - число, вводимое пользователем

\*\return возвращает значение функции

\*\*/

double getEquation2(double x, double const a);

/\*\*

\* @brief Точка входа в программу

\* @return Возвращает 0, если программа работает правильно, иначе 1

\*/

int main()

{

printf\_s("Enter x: ");

double x = getNumber();

const double a = 2.3;

if (x < 2)

{

printf\_s("Value is %lf\n", getEquation1 (x,a));

}

printf\_s("Value is %lf\n", getEquation2(x, a));

return 0;

}

double getNumber()

{

double x;

if (scanf\_s(" %lf", &x) == 3 && x > 0)

{

printf\_s("Wrong value");

abort();

}

return x;

}

double getEquation1(double x, double const a)

{

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

}

double getEquation2(double x, double const a)

{

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

}

**Решение различных тестовых примеров на С**

Ниже представлены выводы программы 2.2 в С (Рисунок 17, Рисунок 18, Рисунок 19)



Рисунок 17 – Вывод программы 2.2, при x>2

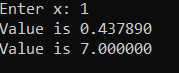


Рисунок 18 – Вывод программы 2.2, при x<2

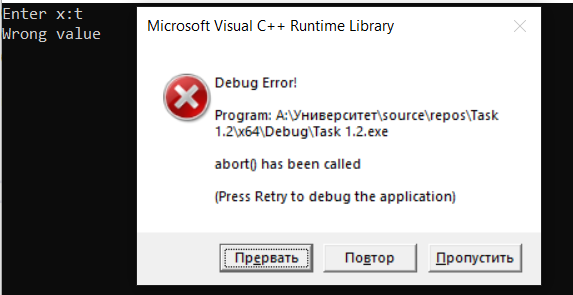


Рисунок 19 – Вывод программы 2.2, если значение задано буквой

**Решение различных тестовых примеров в** Excel

Ниже представлено решение задания 2.2 в Excel (Рисунок 20)

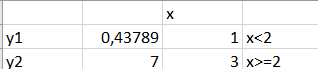


Рисунок 20 - Решение тестового примера задания 2.2 в Excel

**Подтверждение approve 2.2**

Ниже представлено доказательство того, что задание 2.2 было принято (Рисунок 21)

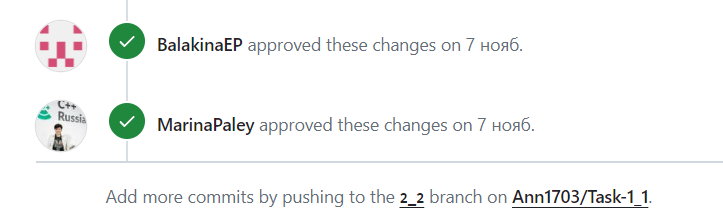


Рисунок 21 – Approve задачи 2.2

**Решение задания 2.3**

**Формулировка задачи**

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 – Условие задачи 2.3

|  |  |
| --- | --- |
| **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? |

**Блок-схема алгоритма**

Ниже представлена блок-схема алгоритма (Рисунок 22, Рисунок 23, Рисунок 24,Рисунок 25, Рисунок 26) .

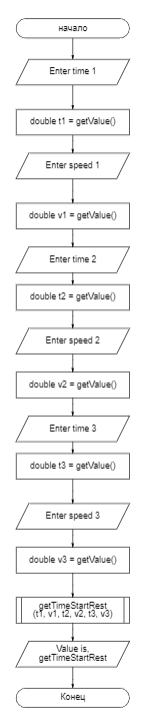


Рисунок 22 – Блок-схема функции main()

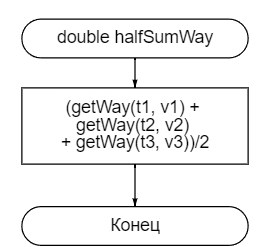


Рисунок 23 – Блок-схема функции halfSumWay ()

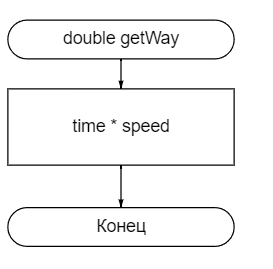


Рисунок 24 – Блок-схема функции getWay ()

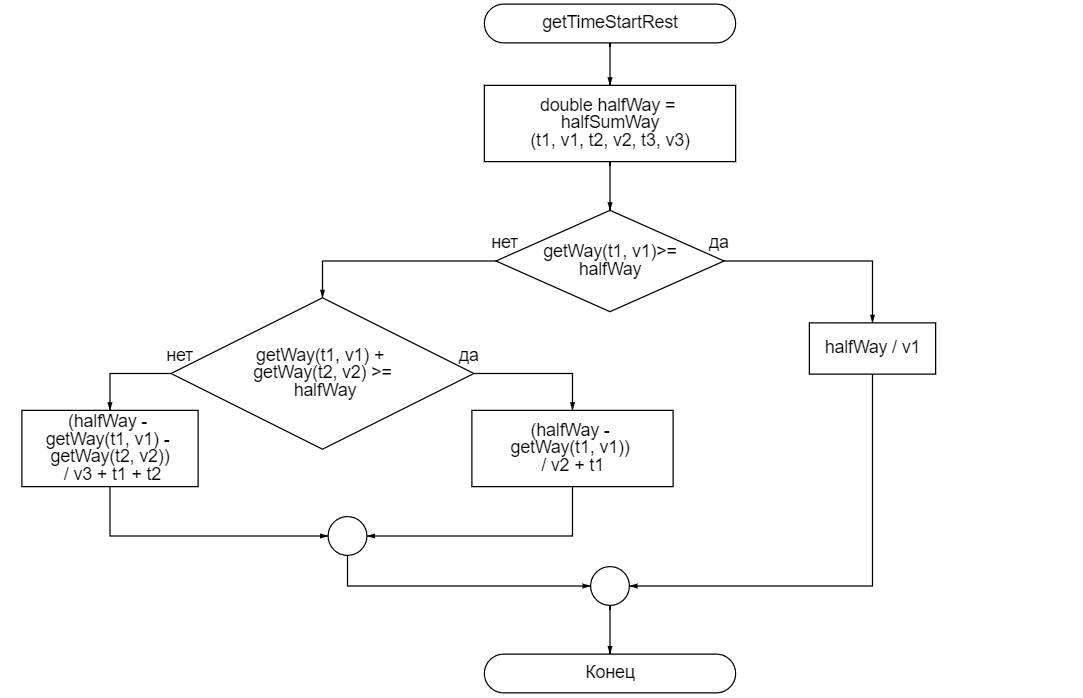


Рисунок 25 – Блок-схема функции getTimeStartRest ()

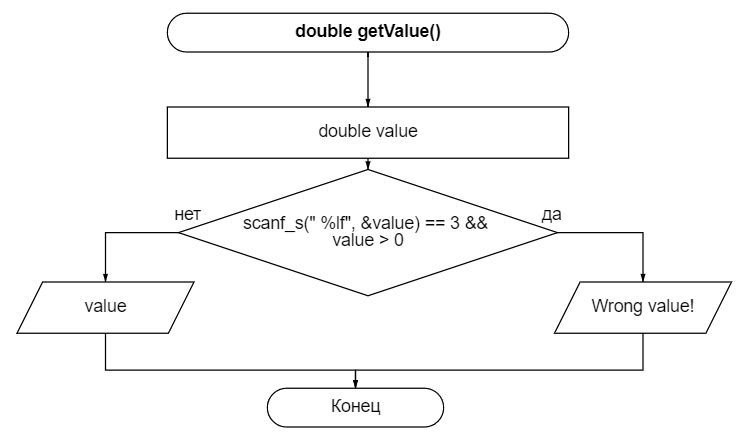


Рисунок 26 – Блок-схема функции getValue ()

# Поясняющие формулы

time - время движения по участку

speed - скорость движения по участку

t1 − время движения по первому участку

v1 - скорость движения по первому участку

t2 − время движения по второму участку

v2 − скорость движения по второму участку

t3 − время движения по третьему участку

v3 − скорость движения по третьему участку

getTimeStartRest ­­­­­­− функция считает время начала привала

halfSumWay­функция − считает половину суммарного пройденного пути

getWay­функция − рассчитывает пройденный путь

**Код для задания 2.3**

#include <stdio.h>

#include <stdlib.h>

/\*\*

\* @brief считывает и проверяет ввод переменной типа double

\* @return Возвращает считанное значение

\*/

double getValue();

/\*\*

\* @brief функция рассчитывает пройденный путь

\* @param time - время движения по участку

\* @param speed - скорость движения по участку

\* @return возвращает значение функции

\*\*/

double getWay(double time, double speed);

/\*\*

\* @brief функция считает половину суммарного пройденного пути

\* @param t1 - время движения по первому участку

\* @param v1 - скорость движения по первому участку

\* @param t2 - время движения по второму участку

\* @param v2 - скорость движения по второму участку

\* @param t3 - время движения по третьему участку

\* @param v3 - скорость движения по третьему участку

\* @return возвращает значение функции

\*\*/

double halfSumWay(double t1, double v1, double t2, double v2, double t3, double v3);

/\*\*

\* @brief функция считает время начала привала

\* @param t1 - время движения по первому участку

\* @param v1 - скорость движения по первому участку

\* @param t2 - время движения по второму участку

\* @param v2 - скорость движения по второму участку

\* @param t3 - время движения по третьему участку

\* @param v3 - скорость движения по третьему участку

\* @return возвращает значение функции

\*\*/

double getTimeStartRest(double t1, double v1, double t2, double v2, double t3, double v3);

/\*\*

\* @brief точка входа в программу

\* @return возвращает 0, если программа выполнена верно, иначе 1

\*/

int main() {

printf\_s("Enter time 1\n");

double t1 = getValue();

printf\_s("Enter speed 1\n");

double v1 = getValue();

printf\_s("Enter time 2\n");

double t2 = getValue();

printf\_s("Enter speed 2\n");

double v2 = getValue();

printf\_s("Enter time 3\n");

double t3 = getValue();

printf\_s("Enter speed 3\n");

double v3 = getValue();

printf\_s("Value is %lf\n", getTimeStartRest(t1, v1, t2, v2, t3, v3));

}

double getValue()

{

double value;

if (scanf\_s(" %lf", &value) == 3 && value > 0)

{

printf\_s("Wrong value");

abort();

}

return value;

}

double getTimeStartRest(double t1,double v1, double t2, double v2,double t3,double v3)

{

double halfWay = halfSumWay(t1, v1, t2, v2, t3, v3);

if (getWay(t1, v1)>=halfWay)

{

return halfWay / v1;

}

else

if (getWay(t1, v1) + getWay(t2, v2) >= halfWay)

{

return (halfWay - getWay(t1, v1)) / v2 + t1;

}

return (halfWay - getWay(t1, v1) - getWay(t2, v2)) / v3 + t1 + t2;

}

double halfSumWay(double t1, double v1, double t2, double v2, double t3, double v3)

{

return (getWay(t1, v1) + getWay(t2, v2) + getWay(t3, v3))/2;

}

double getWay(double time, double speed)

{

return time \* speed;

}

**Решение различных тестовых примеров на С**

Ниже представлены выводы программы 2.3 в С (Рисунок 27, Рисунок 28, Рисунок 29 )

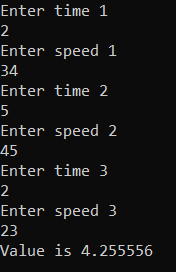


Рисунок 27 – Вывод программы 2.3

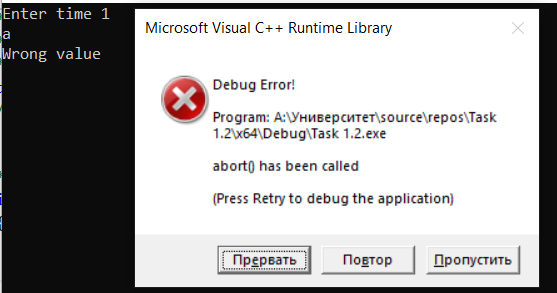


Рисунок 28 – Вывод программы 2.3 если время задано буквой

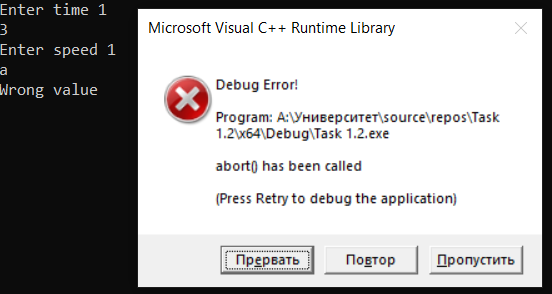


Рисунок 29 – Вывод программы 2.3 если скорость задана буквой

**Решение различных тестовых примеров в Excel**

Ниже представлено решение задания 2.3 в Excel (Рисунок 30)

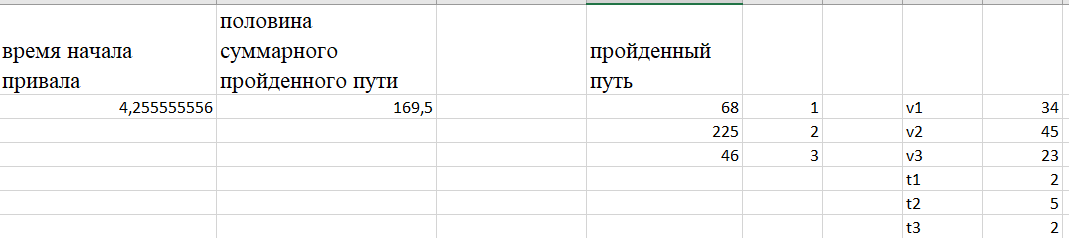


Рисунок 30 - Решение тестового примера задания 2.3 в Excel

**Подтверждение approve 2.3**

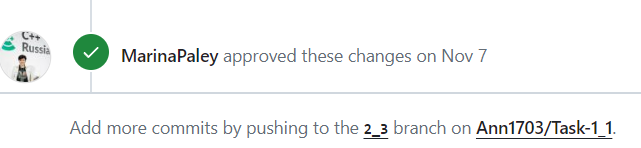
Ниже представлено доказательство того, что задание 2.3 было принято (Рисунок 31) 

Рисунок 31 – Approve задачи 1.3

Оглавление

[Решение задачи 2.1 2](#_Toc153880994)

[Формулировка задачи 2](#_Toc153880995)

[Решение различных тестовых примеров на С 9](#_Toc153880996)

[Решение различных тестовых примеров в Excel 10](#_Toc153880997)

[Подтверждение approve 2.1 10](#_Toc153880998)

[Решение задачи 2.2 11](#_Toc153880999)

[Формулировка задачи 11](#_Toc153881000)

[Блок-схема алгоритма 12](#_Toc153881001)

[Код для задания 2.2 14](#_Toc153881002)

[Решение различных тестовых примеров на С 15](#_Toc153881003)

[Решение различных тестовых примеров в Excel 16](#_Toc153881004)

[Подтверждение approve 2.2 16](#_Toc153881005)

[Решение задания 2.3 18](#_Toc153881006)

[Формулировка задачи 18](#_Toc153881007)

[Блок-схема алгоритма 18](#_Toc153881008)

[Поясняющие формулы 21](#_Toc153881009)

[Код для задания 2.3 22](#_Toc153881010)

[Решение различных тестовых примеров на С 24](#_Toc153881011)

[Решение различных тестовых примеров в Excel 25](#_Toc153881012)

[Подтверждение approve 2.3 26](#_Toc153881013)