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

Скрипников Егор Сергеевич

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

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

Москва 2024

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 1 – Формулировка задания 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? |

1. Блок-схема алгоритма

Блок-схема основного алгоритма представлена ниже (Рисунок 1). Блок-схемы функций расчета значений функции и ввода представлены ниже (Рисунок 2, 3, 4).



Рисунок  ­ Блок-схема основного алгоритма



Рисунок 2 – Блок-схема функции input()



Рисунок 3 – Блок-схема функции checkValue(value)



­Рисунок 4 – Блок-схема функции getTime(v1, v2, v3, t1, t2, t3)

1. Текст программы на языке C

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <errno.h>

#include <float.h>

*/\*\**

*\* @brief Считывает вещественное значение*

*\* @return Возвращает вещественное 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 время до привала*

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

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

*\* @param v3 t3 скорость и время на третьем участке пути*

*\* @return Возвращает время затраченное на половину пути*

*\*/*

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

*/\*\**

*\* @brief Точка входа в программу*

*\* @return Возвращает 0 в случае успеха*

*\*/*

int main(void)

{

    puts("Enter speed and time for first part");

    const double v1 = input();

    const double t1 = input();

    puts("Enter speed and time for second part");

    const double v2 = input();

    const double t2 = input();

    puts("Enter speed and time for third part");

    const double v3 = input();

    const double t3 = input();

    printf("Time before break %lf", getTime(v1, v2, v3, t1, t2, t3));

    return 0;

}

double input(void)

{

    double value = 0.0;

    int result = scanf\_s("%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 getTime(const double v1, const double v2, const double v3, const double t1, const double t2, const double t3)

{

    double s1 = v1 \* t1;

    double s2 = v2 \* t2;

    double s3 = v3 \* t3;

    double wholepath = s1 + s2 + s3;

    double halfpath = wholepath / 2;

    if (halfpath < s1)

    {

        return halfpath / v1;

    }

    halfpath = halfpath - s1;

    if (halfpath < s2)

    {

        return t1 + (halfpath / v2);

    }

    halfpath = halfpath - s2;

    if (halfpath < s3)

    {

        return t1 + t2 + (halfpath / v3);

    }

    return t1 + t2 + t3;

}

1. Результаты выполнения программы

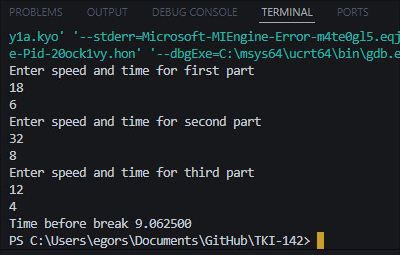
Результаты выполнения программы представлены ниже (Рисунок 5, 6, 7, 8, 9, 10, 11).

Рисунок 5 – Результаты выполнения программы

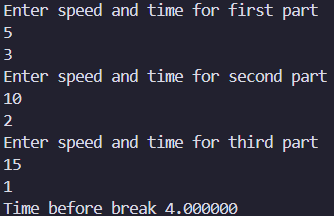


Рисунок 6 – Результаты выполнения программы, когда половина пути меньше первого участка

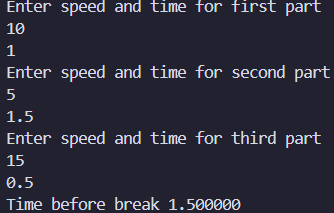


Рисунок 7 – Результаты выполнения программы, когда половина пути меньше второго участка

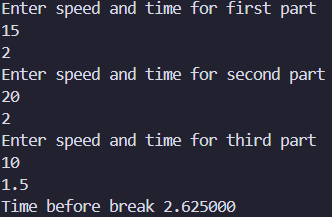


Рисунок 8 – Результаты выполнения программы, когда половина пути меньше третьего участка пути

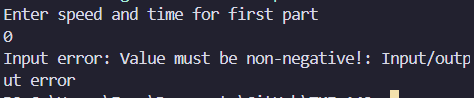


Рисунок 9 – Результаты выполнения программы, когда введенное число – 0

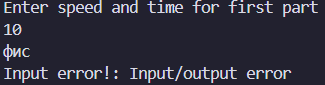


Рисунок 10 – Результат выполнения программы, когда введенное число – буква

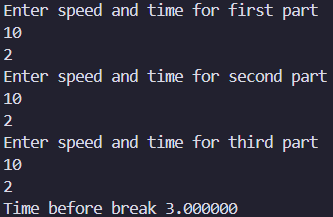


Рисунок 11 – Результат выполнения программы, когда все параметры равны

1. Выполнение тестовых примеров

При помощи языка программирования Python выполнены тестовые примеры. Результаты их выполнения представлены ниже (Рисунок 12).

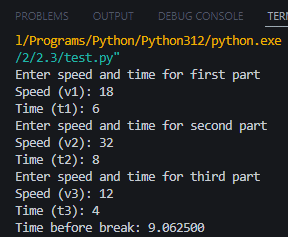
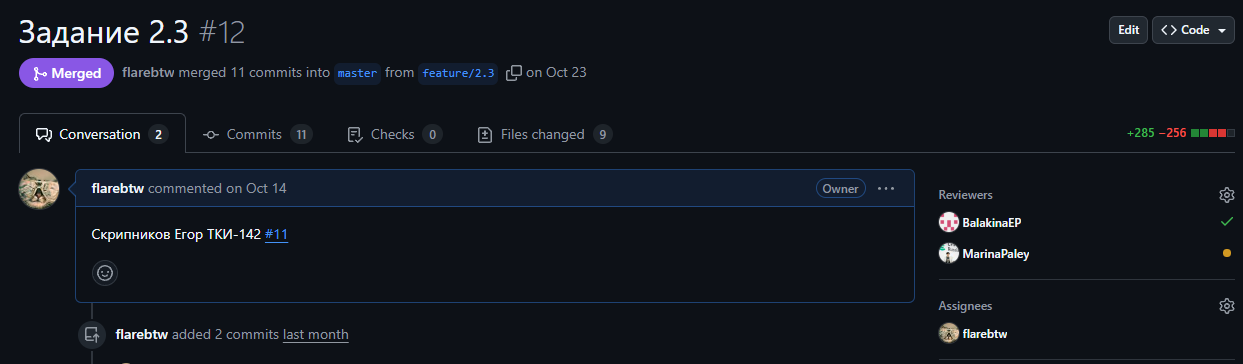


Рисунок 12 – Результат выполнения программы

1. Отметка о выполнении задания в веб-хостинге системы контроля версий