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 №3

По дисциплине «Языки программирования»

ВАРИАНТ 6

Выполнил: ст. гр. ТКИ-141

Винтфельд Рина Дмитриевна

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

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

Москва 2023

Оглавление

[1 Задача 2.1 3](#_Toc153666539)

[1.1 Формулировка задания 2.1 3](#_Toc153666540)

[1.2 Блок-схема алгоритма 2.1 4](#_Toc153666541)

[1.3 Текст программы на языке C 2.1 6](#_Toc153666542)

[1.4 Результаты выполнения программы 2.1 9](#_Toc153666543)

[1.5 Выполнение тестовых примеров 2.1 10](#_Toc153666544)

[1.6 Отметка о выполнении задания в веб-хостинге системы контроля версий 2.1 11](#_Toc153666545)

[2 Задача 2.2 12](#_Toc153666546)

[2.1 Формулировка задания 2.2 12](#_Toc153666547)

[2.2 Блок-схема алгоритма 2.2 13](#_Toc153666548)

[2.3 Текст программы на языке C 2.2 15](#_Toc153666549)

[2.4 Результаты выполнения программы 2.2 17](#_Toc153666550)

[2.5 Выполнение тестовых примеров 2.2 18](#_Toc153666551)

[2.6 Отметка о выполнении задания в веб-хостинге системы контроля версий 1.2 19](#_Toc153666552)

[3 Задача 2.3 20](#_Toc153666553)

[3.1 Формулировка задания 2.3 20](#_Toc153666554)

[3.2 Блок-схема алгоритма 2.3 21](#_Toc153666555)

[3.3 Текст программы на языке C 2.3 24](#_Toc153666556)

[3.4 Результаты выполнения программы 2.3 27](#_Toc153666557)

[3.5 Выполнение тестовых примеров 2.3 28](#_Toc153666558)

[3.6 Отметка о выполнении задания в веб-хостинге системы контроля версий 2.3 29](#_Toc153666559)

1. Задача 3.1
   1. Формулировка задания 3.1

Протабулировать заданную в таблице функцию. Использовать данные в таблице значения шага и интервала в качестве ввода пользователя для решения тестового примера. При невозможности расчёта функции в конкретной точке выводить её значение и надпись, означающую отсутствие решения.

Таблица 1 – Исходные данные

|  |  |  |
| --- | --- | --- |
| Вариант | Функция | Константы |
| 6 |  |  |

* 1. Блок-схема алгоритма 3.1

Блок-схема основного алгоритма представлена ниже (Рисунок 1). Блок-схемы используемых функций представлены ниже (Рисунок 2, Рисунок 3).

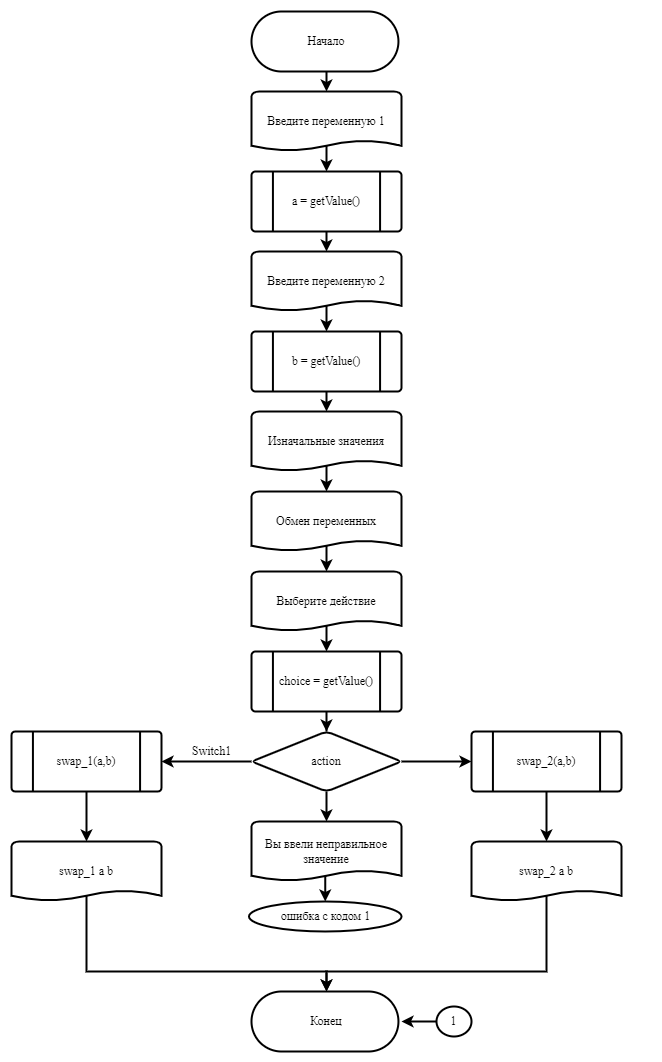


Рисунок 1 ­- Блок-схема основного алгоритма

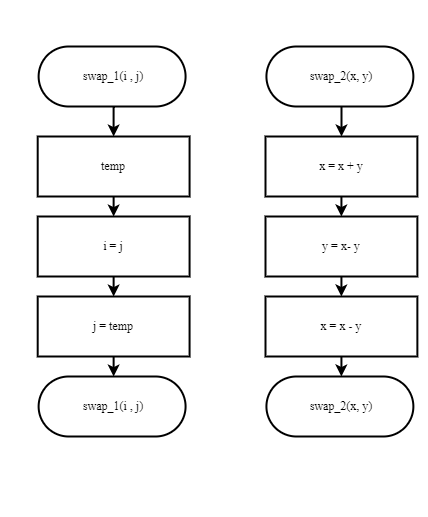


Рисунок 2 – Блок-схема используемых функций

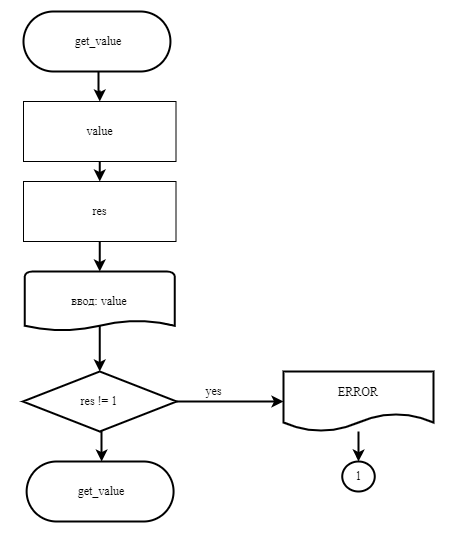


Рисунок 3 - Блок-схема используемой функций

* 1. Текст программы на языке C 3.1

#include <stdio.h>

#include <stdlib.h> //abort

#include <errno.h> //error

/\*

\* @brief Функция меняет между собой значения двух переменных при помощи третьей

\* @param i - первая переменная

\* @param j - вторая переменная

\*/

void swap\_1(double\* i, double\* j);

/\*

\* @brief Функция меняет между собой значения двух переменных

\* @param x - первая переменная

\* @param y - вторая переменная

\*/

void swap\_2(double\* x, double\* y);

/\*

\* @brief проверяет пользовательский ввод

\* @return введеное значение

\*/

double get\_value();

/\*

\* @brief Пользовательский ввод

\* @param Switch1 действие 1) обмен переменных с помощью 3 переменной

\* @param Switch2 действие 2) обмен переменных без помощи 3 переменной

\*/

enum Action{Switch1=1, Switch2};

/\*

\* @brief является точкой входа в программу

\* @return 0, в случае успеха

\*/

int main() {

printf\_s("Введите переменную 1\n");

double a = get\_value();

printf\_s("Введите переменную 2\n");

double b = get\_value();

printf\_s("Изначальные значения: a = %lf, b = %lf\n", a, b);

printf\_s("%d - Обмен переменных при помощи 3 переменной\n%d - Обмен переменных без помощи 3 переменной\n", Switch1, Switch2);

printf\_s("Выберите действие: ");

int choice = get\_value();

enum Action action = (enum Action)choice;

switch (action)

{

case Switch1:

swap\_1(&a, &b);

printf\_s("swap\_1 a = %.1lf, b = %.1lf\n", a, b);

break;

case Switch2:

swap\_2(&a, &b);

printf\_s("swap\_2 a = %.1lf, b = %.1lf\n", a, b);

break;

default:

printf\_s("Вы ввели неправильное значение");

return 1;

break;

}

return 0;

}

double get\_value()

{

double value;

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

if (res != 1)

{

errno = EIO;

perror("ERROR");

abort();

}

return value;

}

void swap\_1(double\* i, double\* j) {

double temp = \*i;

\*i = \*j;

\*j = temp;

}

void swap\_2(double \*x, double \*y){

\*x = \*x + \*y;

\*y = \*x - \*y;

\*x = \*x - \*y;

}

* 1. Результаты выполнения программы 3.1

Результаты выполнения программы представлены ниже (Рисунок 4, Рисунок 5).

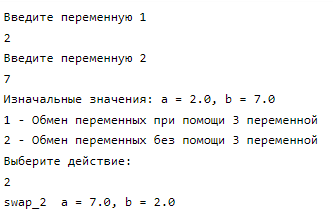


Рисунок 4 - Результат выполнения второго пункта программы

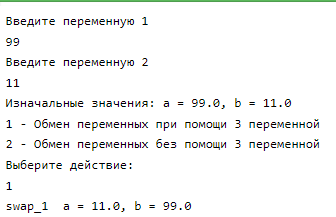


Рисунок 5 - Результат выполнения первого пункта программы

* 1. Выполнение тестовых примеров 3.1

В программе Paint выполнен тестовый пример. Результат его выполнения представлен ниже (Рисунок 6).

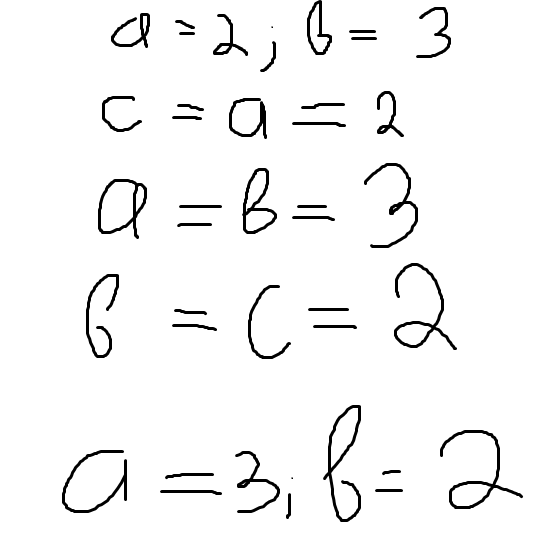


Рисунок 6 – Тестовый пример

* 1. Отметка о выполнении задания в веб-хостинге системы контроля версий 3.1

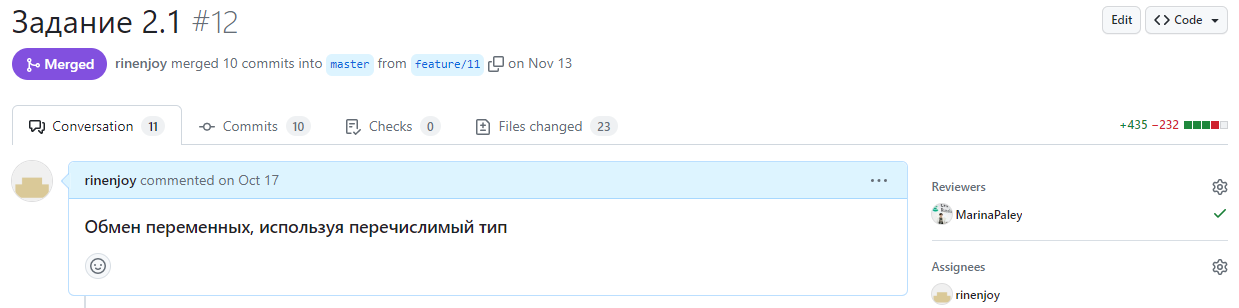


Рисунок 7 - Approved task 2.1

1. Задача 3.2
   1. Формулировка задания 3.2

Составьте две программы:

1. вычислить сумму первых *n* членов последовательности (*k* = 1, 2, 3 ..., *n*).
2. вычислить сумму всех членов последовательности, не меньших заданного числа *e*.

Помните о проверке пользовательского ввода. Все результаты вывести на экран. Отчёт дополнить блок-схемой. При вычислении факториалов рекомендуется отказаться от использования рекурсивных методов.

Таблица 2 - Исходные данные

|  |  |
| --- | --- |
| Вариант | Ряд |
| 6 |  |

* 1. Блок-схема алгоритма 3.2

Блок-схема основного алгоритма представлена ниже (Рисунок 8). Блок-схемы используемых функций представлены ниже (Рисунок 9, Рисунок 10)

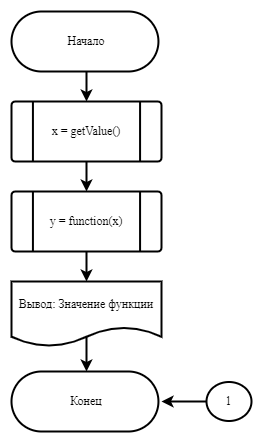


Рисунок 8 - Блок-схема основного алгоритма

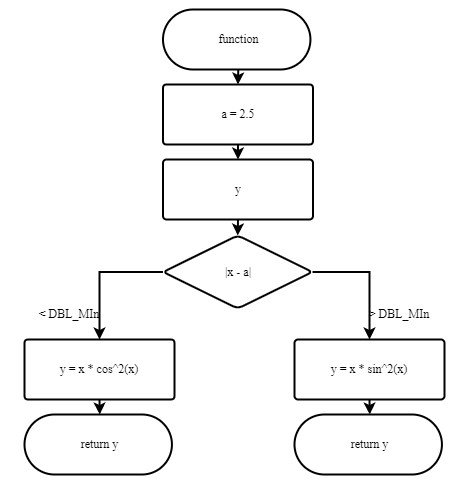


Рисунок 9 - Блок-схема используемой функции

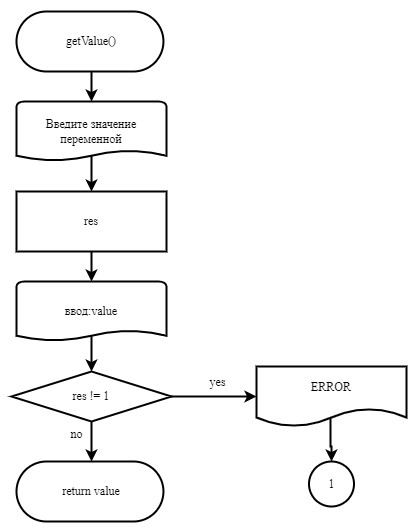


Рисунок 10 - Блок-схема используемой функции

* 1. Текст программы на языке C 3.2

#include <stdio.h>

#include <stdlib.h> //abort

#include <errno.h> //error

#include <math.h>

#include <float.h> //минимальные значения чисел с плавающей точкой

/\*

\* @brief проверяет пользовательский ввод

\* @return введеное значение

\*/

double getValue();

/\*

\* @brief считает значение функции

\* @param x - переменная

\* @return значение функции

\*/

double function(double x);

/\*

\* @brief является точкой входа в программу

\* @return 0, в случае успеха

\*/

int main() {

double x = getValue();

double y = function(x);

printf\_s("Значение функции = %.3lf", y);

return 0;

}

double getValue(){

printf\_s("Введите значение переменной\n");

double value;

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

if (res != 1)

{

errno = EIO;

perror("ERROR");

abort();

}

return value;

}

double function(double x)

{

const double a = 2.5;

double y;

if ((x - a) > DBL\_MIN)

{

y = x \* pow(sin(x),2);

}

else

{

y = x \* pow(cos(x),2);

}

return y;

}

* 1. Результаты выполнения программы 3.2

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

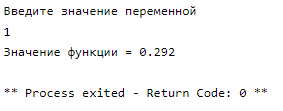


Рисунок 11 - Результат выполнения программы

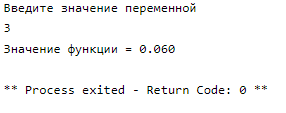


Рисунок 12 - Результат выполнения программы

* 1. Выполнение тестовых примеров 3.2

В приложении Photomath выполнены тестовые примеры. Результаты их выполнения представлены ниже (Рисунок 13, Рисунок 14).



Рисунок 13 – Тестовый расчёт



Рисунок 14 - Тестовый расчёт

* 1. Отметка о выполнении задания в веб-хостинге системы контроля версий 3.2

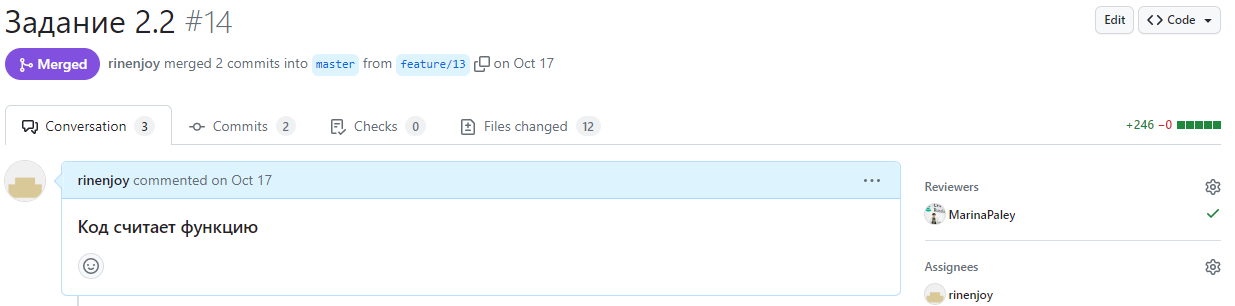


Рисунок 15 - Approved task 2.2

1. Задача 3.3
   1. Формулировка задания 3.3

Протабулировать заданную функцию и сумму функционального ряда разложения этой функции на интервале [*a,b*] и с шагом *h* (шаг и интервал задается в константах). Функциональнй ряд вычисляется по соответствующей рекуррентной формуле с заданной точностью *ɛ*. В результате показать три столбца: значение аргумента, значение функции в данной точке и значение суммы ряда, вычисленное с заданной точностью в данной точке. Два последних столбца должны иметь близкие результаты.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Вар | Функция y | Сумма | Интервал | ɛ |
| 6 |  |  |  | 10-4 |

* 1. Блок-схема алгоритма 3.3

Блок-схема основного алгоритма представлена ниже (Рисунок 16). Блок-схемы используемых функций представлены ниже (Рисунок 17, Рисунок 18, Рисунок 19).

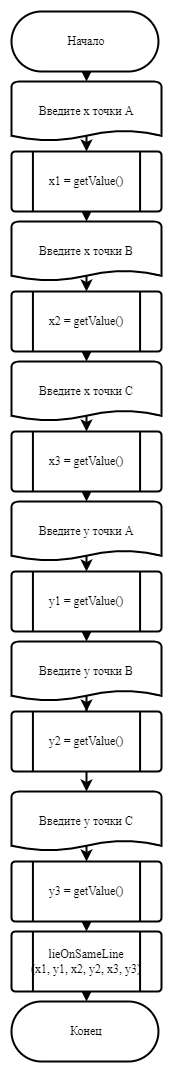


Рисунок 16 - Блок-схема основного алгоритма

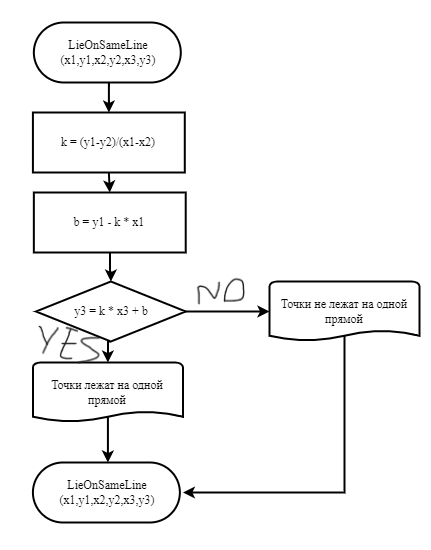


Рисунок 17 - Блок-схема используемой функции

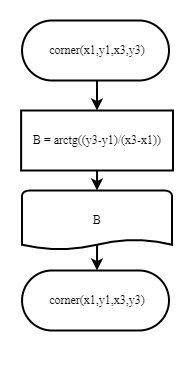


Рисунок 18 - Блок-схема используемой функции

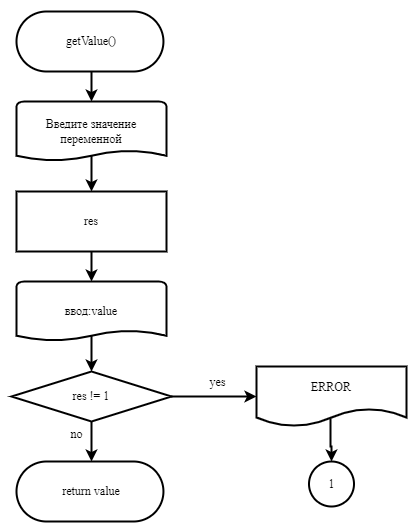


Рисунок 19 - Блок-схема используемой функции

* 1. Текст программы на языке C 3.3

#define \_USE\_MATH\_DEFINES

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <errno.h>

#include <stdbool.h>

#include <float.h>

/\*

\* @brief проверяет пользовательский ввод

\* @return введеное значение

\*/

double getValue();

/\*

\* @brief Функция считает значение угла между 3 точками

\* @param Ax координата х первой точки

\* @param Cx координата х третей точки

\* @param Ay координата у первой точки

\* @param Cy координата у третей точки

\* @return величинa угла В

\*/

double corner(double x1,double y1,double x3,double y3);

/\*

\* @brief Функция определяет лежат ли 3 точки на одной линии

\* @param Ax координата х первой точки

\* @param Bx координата х второй точки

\* @param Cx координата х третей точки

\* @param Ay координата у первой точки

\* @param By координата у второй точки

\* @param Cy координата у третей точки

\* @return если точки лежат на одной прямой, то пишет ответ и завершает программу, в противном случае обращается к функции corner

\*/

double lieOnSameLine(double x1, double y1, double x2, double y2, double x3, double y3);

/\*

\* @brief является точкой входа в программу

\* @return 0, в случае успеха

\*/

int main() {

printf\_s("Введите значение x точки A\n");

double x1 = getValue();

printf\_s("Введите значение y точки A\n");

double y1 = getValue();

printf\_s("Введите значение x точки B\n");

double x2 = getValue();

printf\_s("Введите значение y точки B\n");

double y2 = getValue();

printf\_s("Введите значение x точки C\n");

double x3 = getValue();

printf\_s("Введите значение y точки C\n");

double y3 = getValue();

lieOnSameLine(x1, y1, x2, y2, x3, y3);

return EXIT\_SUCCESS;

}

double getValue() {

double value;

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

if (res != 1)

{

errno = EIO;

perror("ERROR");

abort();

}

return value;

}

double corner(double x1,double y1,double x3,double y3) {

double B = atan((double)(y3 - y1)/(x3 - x1));

B = B \* (180.0 / M\_PI); //пеевод из радиан в градусы

printf\_s("Угол В = %.2lf\n", B);

return B;

}

double lieOnSameLine(double x1, double y1, double x2, double y2, double x3, double y3) {

double k = (double)(y1 - y2) / (x1 - x2);

double b = y1 - k \* x1;

if (y3 - k \* x3 - b <= DBL\_MIN)

{

printf\_s("Точки лежат на одной прямой\n");

return 0.0;

}

else

{

printf\_s("Точки не лежат на одной прямой\n");

return corner(x1, y1, x3, y3);

}

}

* 1. Результаты выполнения программы 3.3

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

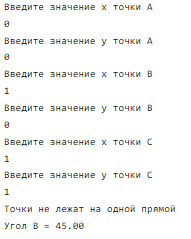


Рисунок 20 - Результат выполнения программы

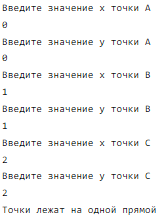


Рисунок 21 - Результат выполнения программы

* 1. Выполнение тестовых примеров 3.3

На сайте snipp.ru выполнены тестовые примеры. Результаты их выполнения представлены ниже ().

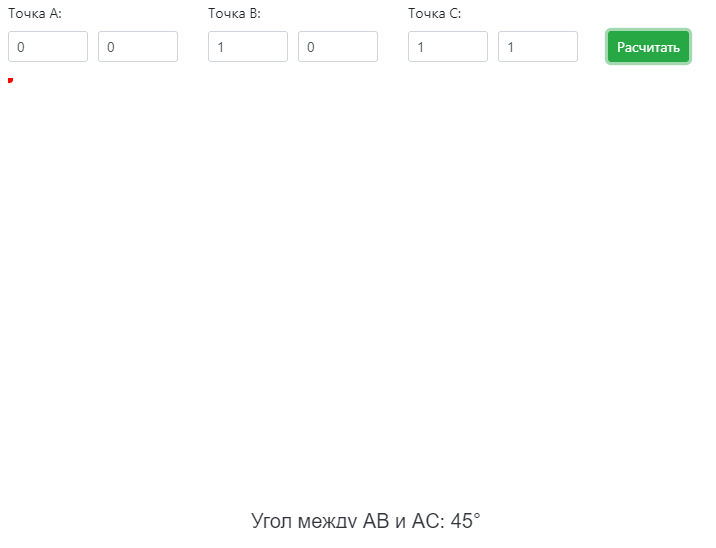


Рисунок 22 - Тестовый расчёт

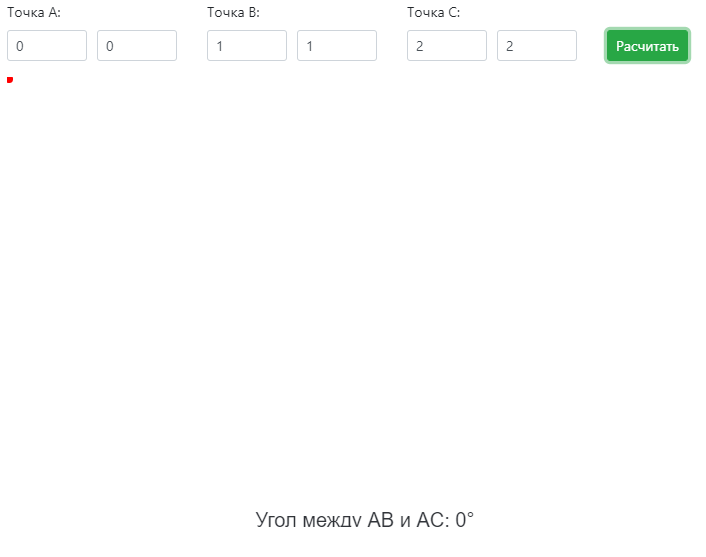


Рисунок 23 - Тестовый расчёт

* 1. Отметка о выполнении задания в веб-хостинге системы контроля версий 3.3

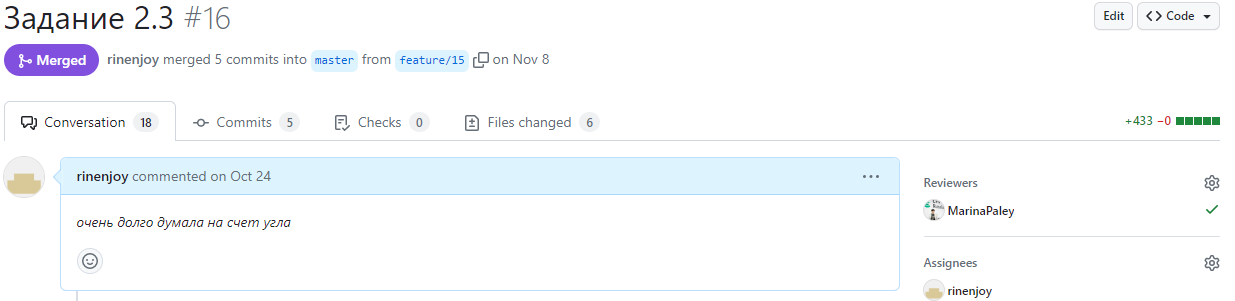


Рисунок 24 - Approved task 2.3