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 1

Выполнил: ст. гр. ТКИ - 142

Шабин Константин Олегович

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

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

Москва 2023

Оглавление

[1. Задание 2.1 3](#_Toc156863868)

[1.1. Формулировка задания 3](#_Toc156863869)

[1.2. Блок-схема алгоритма 4](#_Toc156863870)

[1.3. Код задания 2.1 7](#_Toc156863871)

[1.4. Решение тестовых примеров 9](#_Toc156863872)

[1.5. Зачёт задания в GitHub 10](#_Toc156863873)

[2. Задание 2.2 11](#_Toc156863874)

[2.1. Формулировка задания 11](#_Toc156863875)

[2.2. Блок-схема алгоритма 12](#_Toc156863876)

[2.3. Код задания 2.2 14](#_Toc156863877)

[2.4. Решение тестовых примеров 15](#_Toc156863878)

[2.5. Решение тестовых примеров в Excel 15](#_Toc156863879)

[2.6. Зачёт задания в GitHub 16](#_Toc156863880)

[3. Задание 2.3 17](#_Toc156863881)

[3.1. Формулировка задания 17](#_Toc156863882)

[3.2. Блок-схема алгоритма 18](#_Toc156863883)

[3.3. Код задания 2.3 20](#_Toc156863884)

[3.4. Решение тестового примера 21](#_Toc156863885)

[3.5. Зачёт задания в GitHub 22](#_Toc156863886)

# 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 |
| 1 | По известному радиусу вычислить:   * объём шара; * площадь поверхности шара. |

* 1. Блок-схема алгоритма

Блок-схема алгоритма представлена на рисунках (Рисунок 1, Рисунок *2*, Рисунок *3*, Рисунок *4*, Рисунок 5)

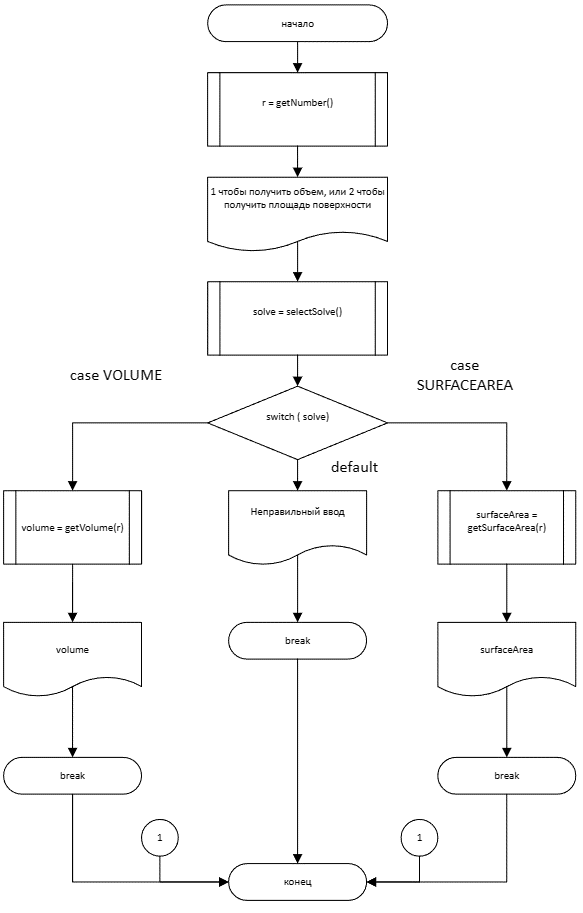


Рисунок 1 – Блок-схема к функции main()

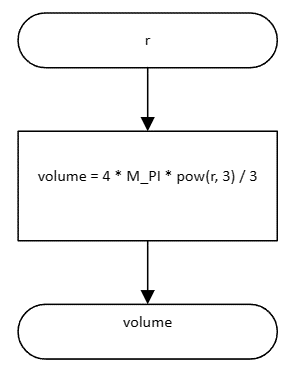


Рисунок 2 – Блок-схема функции getVolume(r)

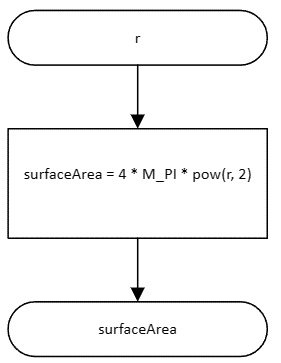


Рисунок 3 – Блок-схема функции getSurfaceArea(r)

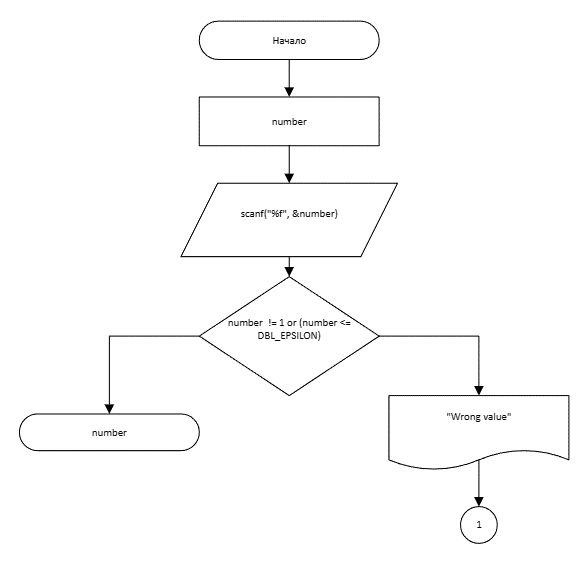
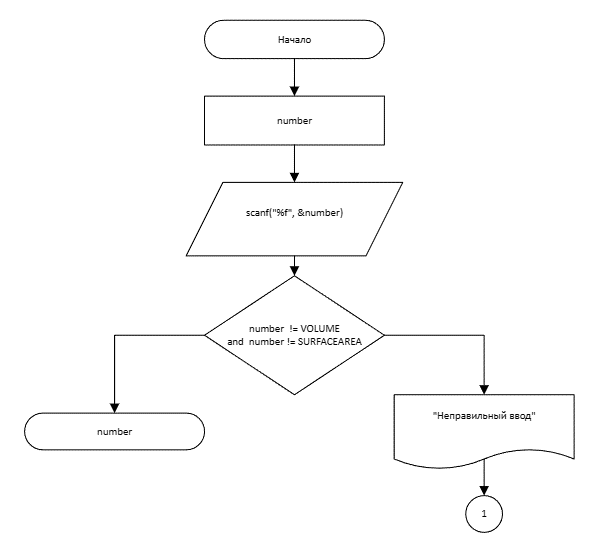
Рисунок 4 – Блок-схема функции get*Number*()

Рисунок 5 – Блок-схема функции selectSolve()

* 1. Код задания 2.1

#include <stdio.h>

#include <math.h>

#include <stdlib.h>

#include <float.h>

enum solve

{

VOLUME = 1,

SURFACEAREA = 2

};

/\*\*

\* @brief Получешние введеного значения из консоли

\* @return Возвращает число при условии правильного ввода

\*/

float getNumber();

/\*\*

\* @brief Выбор типа расчёта

\* @return Возвращает выбор пользователя

\*/

int selectSolve();

/\*\*

\* @brief Поиск значения volume

\* @param r радиус для функции getVolume

\* @return Возвращает результат функции getVolume

\*/

float getVolume(float r);

/\*\*

\* @brief Поиск значения surfaceArea

\* @param r радиус для функции getSurfaceArea

\* @return Возвращает площадь поверхности

\*/

float getSurfaceArea(float r);

/\*\*

\* @brief Точка входа в программу

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

\*/

int main()

{

float r = getNumber();

printf("введите %d чтобы получить объем, или %d чтобы получить площадь поверхности: ", VOLUME, SURFACEAREA);

int solve = selectSolve();

switch ("%d", solve)

{

case VOLUME:

printf("volume = %f", getVolume(r));

break;

case SURFACEAREA:

printf("surface area = %f", getSurfaceArea(r));

break;

default:

printf("\n " "Неправильный ввод" "\n");

break;

}

return 0;

}

float getVolume(float r)

{

return 4 \* M\_PI \* pow(r, 3) / 3;

}

float getSurfaceArea(float r)

{

return 4 \* M\_PI \* pow(r, 2);

}

float getNumber()

{

float number;

if (scanf("%f", &number) != 1 || (number < DBL\_EPSILON))

{

printf("%s" "Неправильный ввод");

abort();

}

return number;

}

int selectSolve()

{

int number;

if (scanf("%d", &number) != VOLUME && number != SURFACEAREA)

{

printf("\n" "Неправильный ввод" "\n");

abort();

}

return number;

}

* 1. Решение тестовых примеров



Рисунок 6 – Вывод объема



Рисунок 7 – Вывод площади поверхности

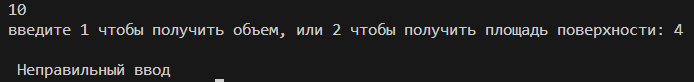


Рисунок 8 – Вывод, когда выбор не входит в указанные значения

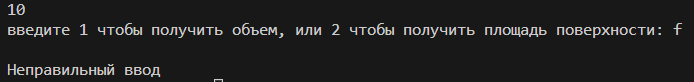


Рисунок 9 – Вывод, когда выбор – буква



Рисунок 10 – Вывод, когда радиус отрицательный

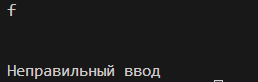


Рисунок 11 – Вывод, когда радиус – буква

* 1. Зачёт задания в GitHub

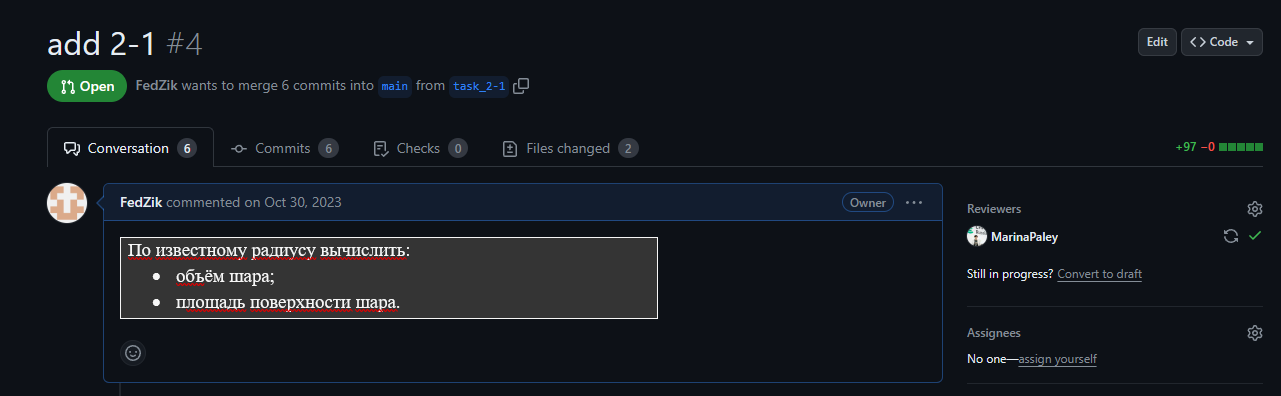


Рисунок 12 – Зачёт задания в GitHub

# 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 |
| 1 |  |  |

Таблица 2 – Формулировка задания 2.2

* 1. Блок-схема алгоритма

Блок-схема алгоритма представлена на рисунках(Рисунок 13, Рисунок 14, Рисунок 15)

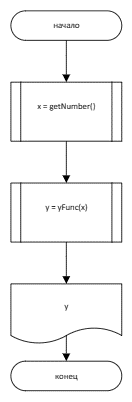


Рисунок 13 – Блок-схема функции main()

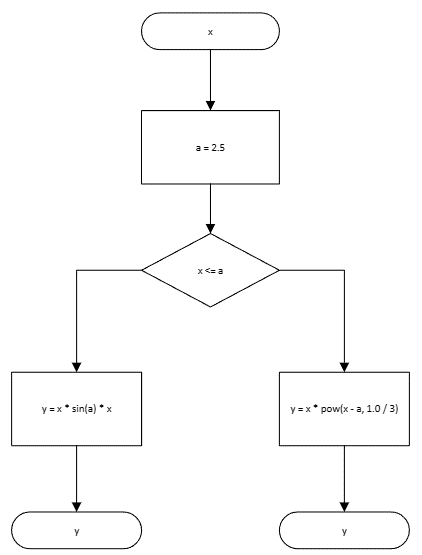


Рисунок 14 – Блок-схема функции yFunc(x)

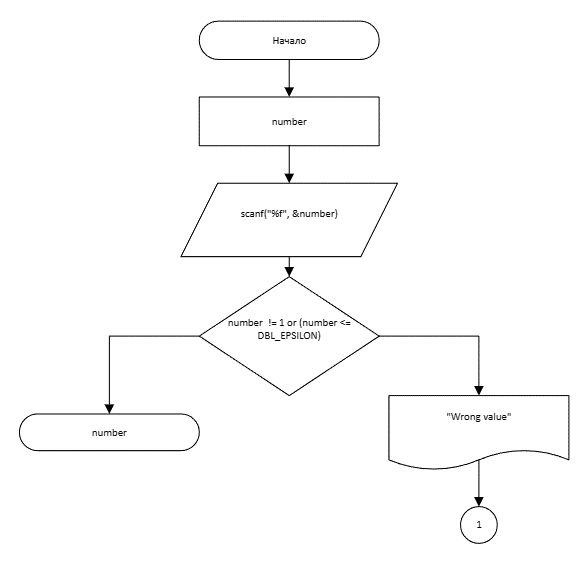


Рисунок 15 – Блок-схема функции getNumber()

* 1. Код задания 2.2

#include <stdio.h>

#include <math.h>

#include <float.h>

/\*\*

\* @brief расчет y

\* @param x значение х

\* @return возвращает значение у

\*\*/

double yFunc(double x);

/\*\*

\* @brief проверка введеного значения

\* @return возвращает значение если ввод правильный, иначе выводит сообщение об ошибке

\*/

float getNumber();

/\*\*

\* @brief точка входа в программу

\* @return возвращает 0, если все верно, возвращает 1 в случае ошибки

\*\*/

int main()

{

double x;

x = getNumber();

double y = yFunc(x);

printf("\n y= %lf", y);

return 0;

}

float getNumber()

{

float number;

if (scanf("%f", &number) != 1 || (number < DBL\_EPSILON))

{

printf("%s" "Wrong value");

abort();

}

return number;

}

double yFunc(double x)

{

const double a = 2.5;

if (x <= a){return x \* sin(a) \* x;}

else {return x \* pow(x - a, 1.0 / 3);};

}

* 1. Решение тестовых примеров



Рисунок 16 – Вывод, когда переменная x меньше либо равна 1

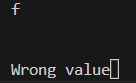


Рисунок 17 – Вывод, когда переменная x – буква

* 1. Решение тестовых примеров в Excel

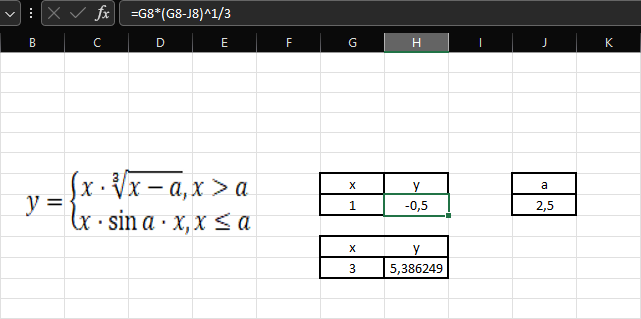


Рисунок 18 – Расчёт y, когда x, меньше a

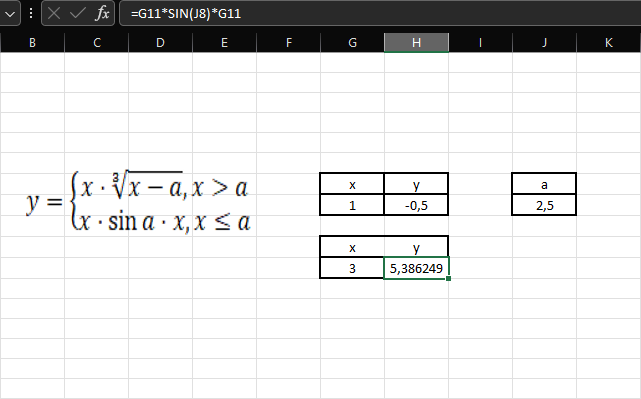


Рисунок 19 – Расчёт y, когда x меньше либо равен a

* 1. Зачёт задания в GitHub

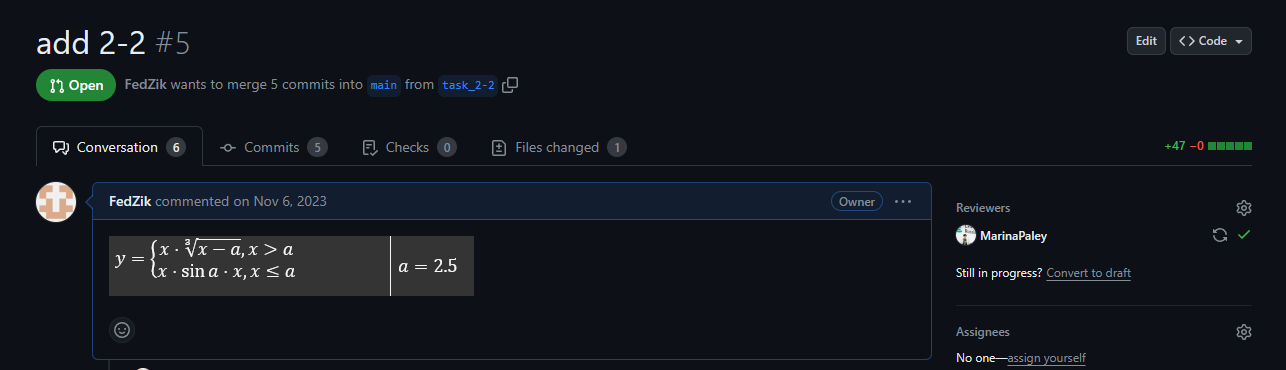


Рисунок 20 – Зачёт задания в GitHub

# 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 |
| 1 | Заданы два натуральных числа. Определить, является ли среднее арифметическое этих чисел целым числом. |

* 1. Блок-схема алгоритма

Блок-схема алгоритма представлена на рисунках (Рисунок 21, Рисунок 22, Рисунок 23, Рисунок 24)

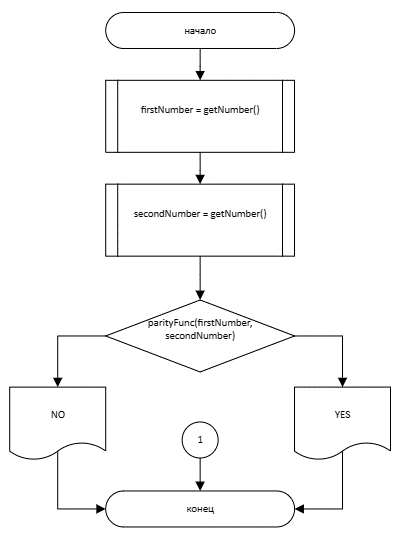


Рисунок 21 – Блок-схема к функции main()

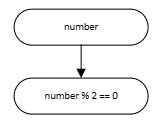


Рисунок 22 – блок-схема функции isEven(number)

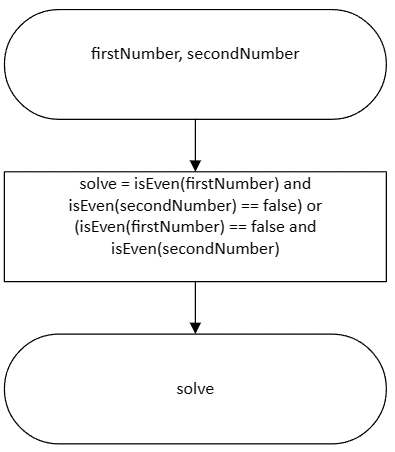


Рисунок 23 – блок-схема функции parityFunc(firstNumber, secondNumber)

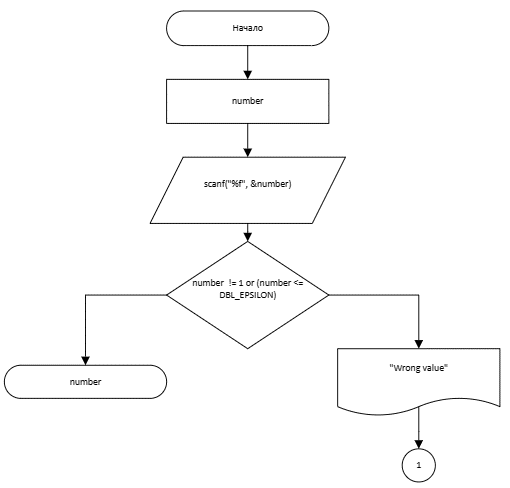


Рисунок 24 – Блок-схема функции getNumber()

* 1. Код задания 2.3

#include <stdio.h>

#include <stdbool.h>

#include <stdlib.h>

/\*\*

\* @brief проверяет целое ли среднее значение

\* @param firstNumber первое число

\* @param secondNumber второе число

\* @return возвращает 1 если среднее значение - целое число, иначе 0

\*/

bool parityFunc(int firstNumber, int secondNumber);

/\*\*

\* @brief проверка введеного значения

\* @return возвращает значение если ввод правильный, иначе выводит сообщение об ошибке

\*/

float getNumber();

/\*\*

\* @brief проверяет число на четность

\* @param number проверяемое число

\* @return возвращает 1 если число четное, иначе 0

\*/

bool isEven(int number);

/\*\*

\* @brief Точка входа в программу

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

\*/

int main()

{

int firstNumber, secondNumber;

firstNumber = getNumber();

secondNumber = getNumber();

if (parityFunc(firstNumber, secondNumber))

{

printf("\n %s", "NO");

}

else

{

printf("\n %s", "YES");

}

return 0;

}

bool parityFunc(int firstNumber, int secondNumber)

{

return (isEven(firstNumber) && isEven(secondNumber) == false) || (isEven(firstNumber) == false && isEven(secondNumber));

}

bool isEven(int number)

{

return ((number % 2) == 0);

}

float getNumber()

{

float number;

if (scanf("%f", &number) != 1 || (number < 0))

{

printf("%s" "Wrong value");

abort();

}

return number;

}

* 1. Решение тестового примера



Рисунок 25 – Вывод в случае успеха



Рисунок 26 – Вывод в случае провала

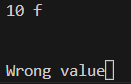


Рисунок 27 – Вывод программы, когда значение выбора – буква

* 1. Зачёт задания в GitHub

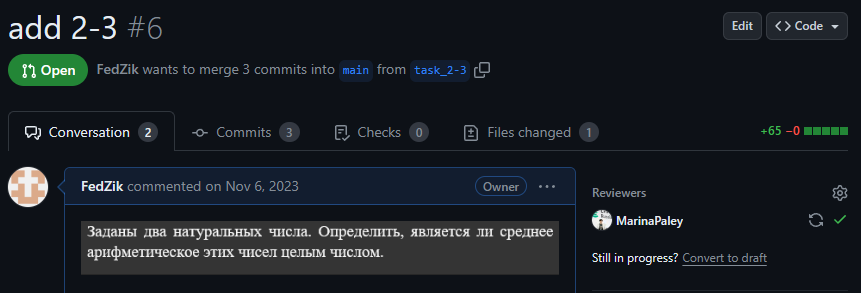


Рисунок 28 – Зачёт задания в GitHub