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 №4.3

По дисциплине «Языки программирования»

ВАРИАНТ 3

Выполнил: ст. гр. ТКИ-142

Величко Иван Сергеевич

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

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

Москва 2024

1. Формулировка задания

Создать многомерный массив nˣm из n целых чисел. Реализовать возможность заполнения массива, как случайными числами, так и с помощью клавиатуры по желанию пользователя. Пункты задания организовать в виде функций (методов). Вывести массив на экран.

Таблица 1 – Исходные данные

|  |  |
| --- | --- |
| Вариант | Задачи |
| 3 | 1. Заменить все элементы первых трех столбцов на их квадраты. 2. Вставить после каждой нечетной строки первую строку. |

1. Блок-схема алгоритма

Ниже представлены:

Блок-схема основного алгоритма (Рисунок 1).

Блок-схема основного алгоритма (Рисунок 1.2).

Блок-схема функции input() (Рисунок 2).

Блок-схема функции positive\_input() (Рисунок 3).

Блок-схема функции check\_array() (Рисунок 4).

Блок-схема функции create\_array() (Рисунок 5).

Блок-схема функции create\_2d\_array() (Рисунок 6).

Блок-схема функции check\_range() (Рисунок 7).

Блок-схема функции random\_array\_filling() (Рисунок 8).

Блок-схема функции random\_2d\_array\_filling() (Рисунок 9).

Блок-схема функции input\_array\_filling() (Рисунок 10).

Блок-схема функции input\_2d\_array\_filling() (Рисунок 11).

Блок-схема функции print\_array() (Рисунок 12).

Блок-схема функции print\_2d\_array() (Рисунок 13).

Блок-схема функции first\_3\_elements\_to\_root() (Рисунок 14).

Блок-схема функции first\_3\_columns\_to\_root() (Рисунок 15).

Блок-схема функции odd\_rows\_count() (Рисунок 16).

Блок-схема функции copy\_row() (Рисунок 17).

Блок-схема функции create\_an\_extended\_2d\_array() (Рисунок 18).

Блок-схема функции free\_2d\_array() (Рисунок 19).

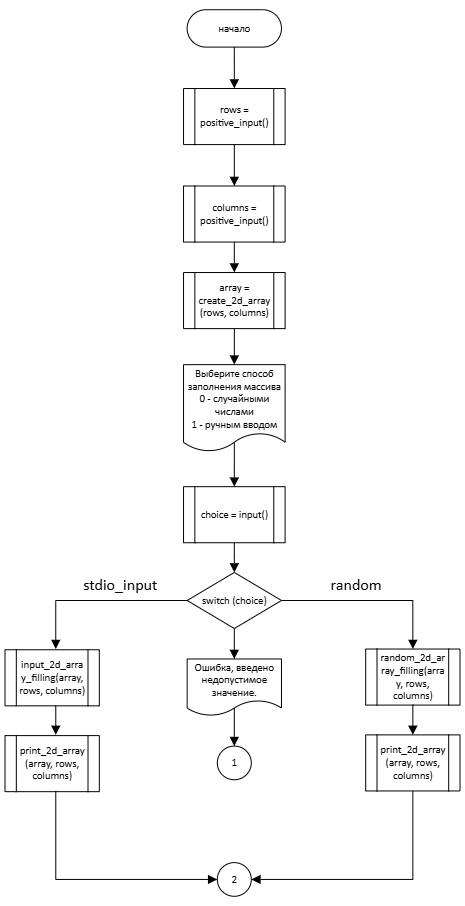


Рисунок 1– Блок-схема основного алгоритма

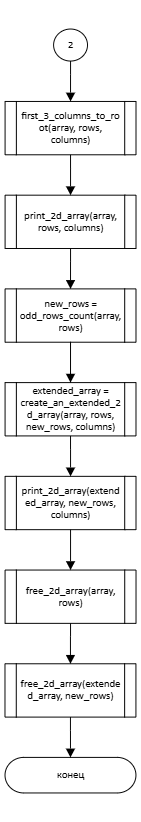


Рисунок 1.2 – Блок-схема основного алгоритма



Рисунок 2 – Блок-схема функции input()

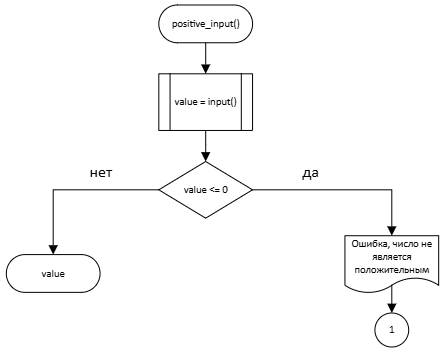


Рисунок 3 – Блок-схема функции positive\_input()

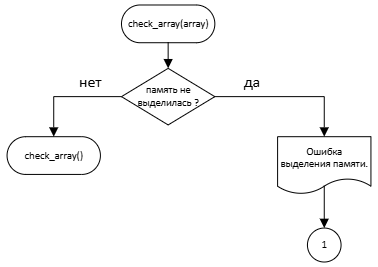


Рисунок 4 – Блок-схема функции check\_array()

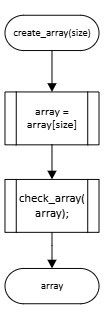


Рисунок 5 – Блок-схема функции create\_array()

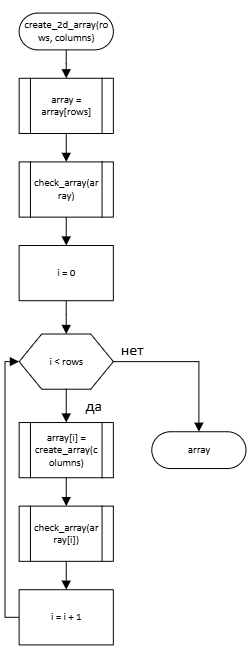


Рисунок 6 – Блок-схема функции create\_2d\_array()

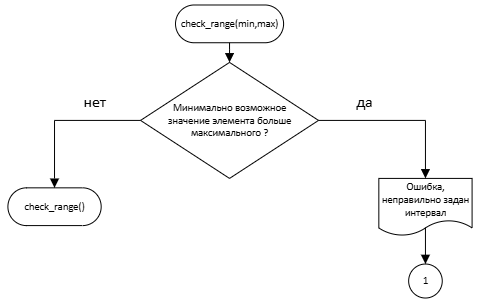


Рисунок 7 – Блок-схема функции check\_range()

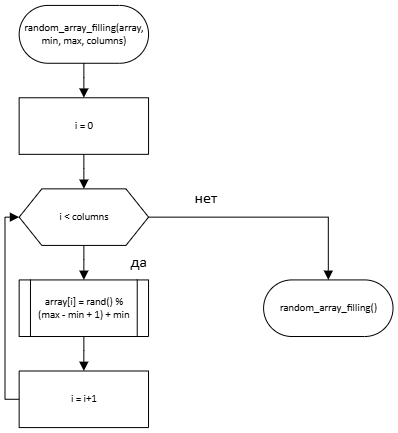


Рисунок 8 – Блок-схема функции random\_array\_filling()

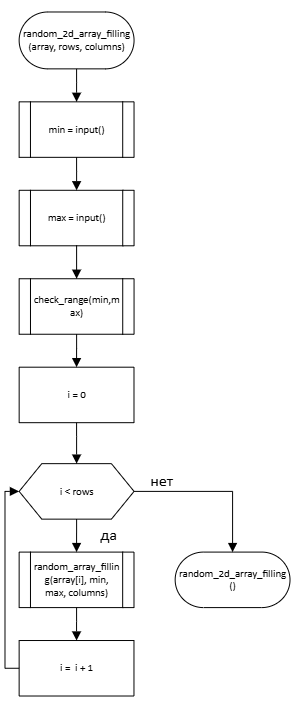


Рисунок 9 – Блок-схема функции random\_2d\_array\_filling()

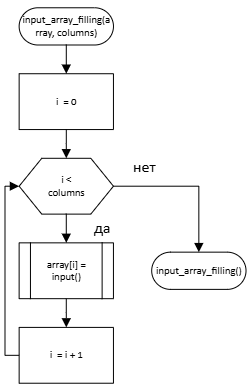


Рисунок 10 – Блок-схема функции input\_array\_filling()

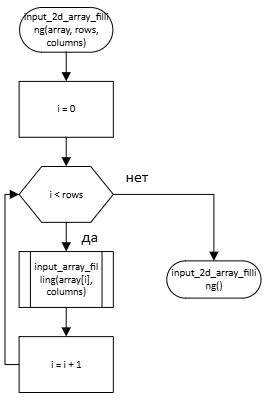


Рисунок 11 – Блок-схема функции input\_2d\_array\_filling()

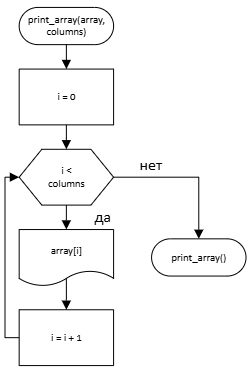


Рисунок 12 – Блок-схема функции print\_array()

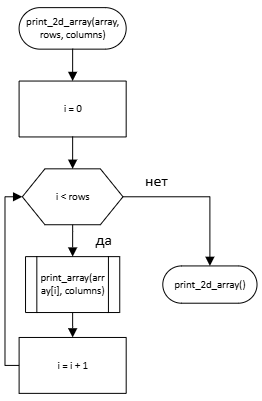


Рисунок 13 – Блок-схема функции print\_2d\_array()

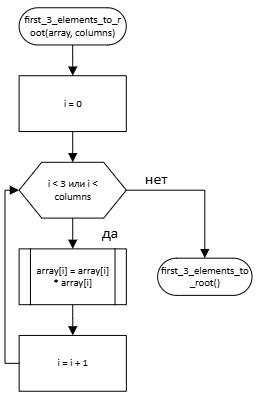


Рисунок 14 – Блок-схема функции first\_3\_elements\_to\_root()

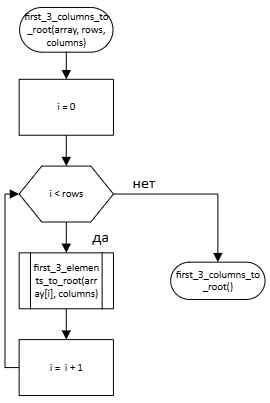


Рисунок 15 – Блок-схема функции first\_3\_columns\_to\_root()

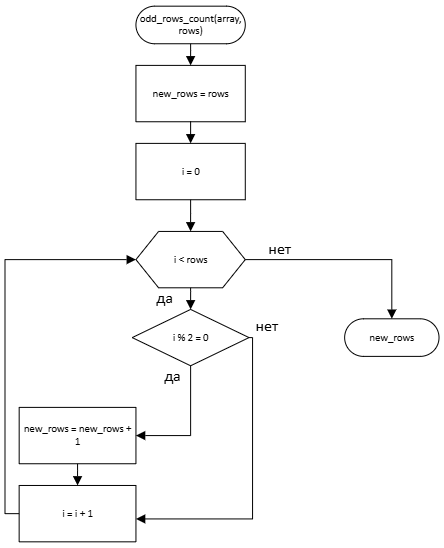


Рисунок 16 – Блок-схема функции odd\_rows\_count()

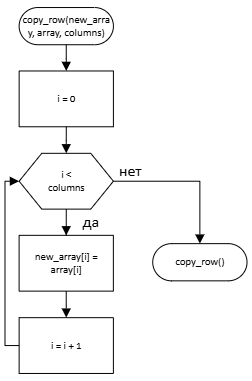


Рисунок 17 – Блок-схема функции copy\_row()

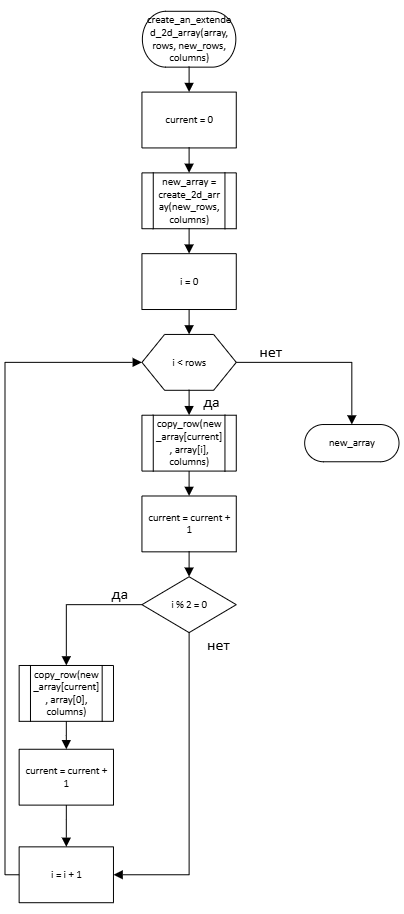


Рисунок 18 – Блок-схема функции create\_an\_extended\_2d\_array()

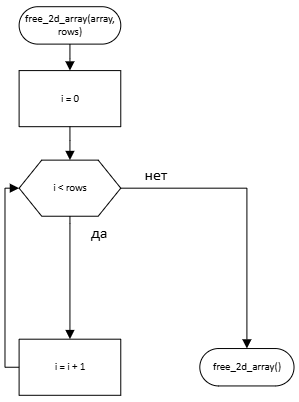


Рисунок 19 – Блок-схема функции free\_2d\_array()

1. Текст программы на языке C

Файл: myheader.c

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

#include <errno.h>

#include <time.h>

#include <math.h>

enum array\_filling {

RANDOM,

STDIO\_INPUT

};

/\*\*

\* @brief принимает целочисленное число из stdin

\* @return целочисленное число из stdin

\*/

int input(void);

/\*\*

\* @brief принимает целочисленное неотрицательное число из stdin

\* @return целочисленное неотрицательное число из stdin

\*/

size\_t positive\_input(void);

/\*\*

\* @brief проверяет выделилась ли память под массив

\* @param array указатель на массив

\*/

void check\_array(const long long\* array);

/\*\*

\* @brief создает массив

\* @param size размер массива

\* @return массив

\*/

long long\* create\_array(const size\_t size);

/\*\*

\* @brief создает двумерный массив

\* @param rows количество массивов(строк) в двумерном массиве

\* @param columns количество элементов каждого массива(столбцов) в двумерном массиве

\* @return двумерный массив

\*/

long long\*\* create\_2d\_array(const size\_t rows, const size\_t columns);

/\*\*

\* @brief проверяет что наибольшее значение интервала больше наимешьшего

\* @param min наимешьшее значение интервала

\* @param max наибольшее щначение интервала

\*/

void check\_range(const long long min, const long long max);

/\*\*

\* @brief заполняет массив случайными числами

\* @param array указатель на массив

\* @param min наимешьшее значение интервала возможных значений элемента массива

\* @param max наибольшее щначение интервала возможных значений элемента массива

\* @param columns количество элементов массива

\*/

void random\_array\_filling(long long\* array, const long long min, const long long max, const size\_t columns);

/\*\*

\* @brief заполняет двумерный массив случайными числами

\* @param array двойной указатель на массив

\* @param rows количество массивов(строк) в двумерном массиве

\* @param columns количество элементов каждого массива(столбцов) в двумерном массиве

\*/

void random\_2d\_array\_filling(long long\*\* array, const size\_t rows, const size\_t columns);

/\*\*

\* @brief заполняет массив значениями из stdin

\* @param array указатель на массив

\* @param columns количество элементов каждого массива(столбцов) в двумерном массиве

\*/

void input\_array\_filling(long long\* array, const size\_t columns);

/\*\*

\* @brief заполняет двумерный массив значениями из stdin

\* @param array двойной указатель на массив

\* @param rows количество массивов(строк) в двумерном массиве

\* @param columns количество элементов каждого массива(столбцов) в двумерном массиве

\*/

void input\_2d\_array\_filling(long long\*\* array, const size\_t rows, const size\_t columns);

/\*\*

\* @brief выводит содержимое массива

\* @param array указатель на массив

\* @param columns количество элементов каждого массива(столбцов) в двумерном массиве

\*/

void print\_array(const long long\* array, const size\_t columns);

/\*\*

\* @brief выводит содержимое двумерного массива

\* @param array двойной указатель на массив

\* @param rows количество массивов(строк) в двумерном массиве

\* @param columns количество элементов каждого массива(столбцов) в двумерном массиве

\*/

void print\_2d\_array(const long long\*\* array, const size\_t rows, const size\_t columns);

/\*\*

\* @brief возводит в степень первые 3 элемента массива

\* @param array указатель на массив

\* @param columns количество элементов каждого массива(столбцов) в двумерном массиве

\*/

void first\_3\_elements\_to\_root(long long\* array, const size\_t columns);

/\*\*

\* @brief возводит в степень первые 3 элемента в каждом из массивов двумерного массива

\* @param array двойной указатель на массив

\* @param rows количество массивов(строк) в двумерном массиве

\* @param columns количество элементов каждого массива(столбцов) в двумерном массиве

\*/

void first\_3\_columns\_to\_root(long long\*\* array, const size\_t rows, const size\_t columns);

/\*\*

\* @brief считает количество нечетных элементов(массивов) в двумерном массиве

\* @param array двойной указатель на массив

\* @param rows количество массивов(строк) в двумерном массиве

\* @return количество нечетных элементов(массивов)

\*/

size\_t odd\_rows\_count(long long\*\* array, const size\_t rows);

/\*\*

\* @brief копирует содержимое элемента двумерного массива

\* @param new\_array элемент нового двумерного массива

\* @param array элемент двумерного массива

\* @param columns количество элементов каждого массива(столбцов) в двумерном массиве

\*/

void copy\_row(long long\* new\_array, const long long\* array, const size\_t columns);

/\*\*

\* @brief создвет новый массив где после каждого нечетного элемента(строка матрици) вставлется первый элемент(строка матрици) старого массива

\* @param array двойной указатель на массив

\* @param rows количество массивов(строк) в двумерном массиве

\* @param количество элементов каждого массива(столбцов) в двумерном массиве

\* @return новый массив

\*/

long long\*\* create\_an\_extended\_2d\_array(long long\*\* array, const size\_t rows, const size\_t new\_rows, const size\_t columns);

/\*\*

\* @brief освобождает выделенную память для элементов двумерого массива(массивов)

\* @param array двойной указатель на массив

\* @param rows количество массивов(строк) в двумерном массиве

\*/

void free\_2d\_array(long long\*\* array, const size\_t rows);

Файл: main.c

#include "myheader.h"

int main(void) {

setlocale(LC\_ALL, "");

puts("Введите количество массивов(строк) в двумерном массиве");

const size\_t rows = positive\_input();

puts("Введите количество элементов каждого массива(столбцов) в двумерном массиве");

const size\_t columns = positive\_input();

long long\*\* array = create\_2d\_array(rows, columns);

printf("Выберите способ заполнения массива\n%d - случайными числами\n%d - ручным вводом\n", RANDOM, STDIO\_INPUT);

enum array\_filling choice = input();

switch (choice) {

case RANDOM:

random\_2d\_array\_filling(array, rows, columns);

puts("Массив: ");

print\_2d\_array(array, rows, columns);

break;

case STDIO\_INPUT:

input\_2d\_array\_filling(array, rows, columns);

puts("Массив: ");

print\_2d\_array(array, rows, columns);

break;

default :

errno = EINVAL;

perror("Ошибка, введено недопустимое значение");

exit(EXIT\_FAILURE);

}

first\_3\_columns\_to\_root(array, rows, columns);

puts("Все элементы первых трех столбцов были возведены в квадрат:");

print\_2d\_array(array, rows, columns);

const size\_t new\_rows = odd\_rows\_count(array, rows);

long long\*\* extended\_array = create\_an\_extended\_2d\_array(array, rows, new\_rows, columns);

puts("После каждой строки с нечетным индексом была добавлена первая строка:");

print\_2d\_array(extended\_array, new\_rows, columns);

free\_2d\_array(array, rows);

free\_2d\_array(extended\_array, new\_rows);

return EXIT\_SUCCESS;

}

Файл: mylib.c

#include "myheader.h"

int input(void) {

int value = 0;

if (scanf\_s("%d", &value) != 1) {

errno = EINVAL;

perror("Ошибка, не числовое значение");

exit(EXIT\_FAILURE);

}

return value;

}

size\_t positive\_input(void) {

int value = input();

if (value <= 0) {

errno = EINVAL;

perror("Ошибка, число не является положительным");

exit(EXIT\_FAILURE);

}

return (size\_t)value;

}

void check\_array(const long long\* array) {

if (!array) {

errno = ENOMEM;

perror("Ошибка выделения памяти.");

exit(EXIT\_FAILURE);

}

}

long long\* create\_array(const size\_t size) {

long long\* array = (long long\*)malloc(size \* sizeof(long long));

check\_array(array);

return array;

}

long long\*\* create\_2d\_array(const size\_t rows, const size\_t columns) {

long long\*\* array = (long long\*\*)malloc(rows \* sizeof(long long\*));

check\_array(array);

for (size\_t i = 0; i < rows; i++) {

array[i] = create\_array(columns);

check\_array(array[i]);

}

return array;

}

void check\_range(const long long min, const long long max) {

if (min > max) {

errno = EINVAL;

perror("Ошибка, неправильно задан интервал");

exit(EXIT\_FAILURE);

}

}

void random\_array\_filling(long long\* array, const long long min, const long long max, const size\_t columns) {

for (size\_t i = 0; i < columns; i++) {

array[i] = rand() % (max - min + 1) + min;

}

}

void random\_2d\_array\_filling(long long\*\* array, const size\_t rows, const size\_t columns) {

srand(time(NULL));

puts("Введите минимальное значение элемента массива:");

long long min = input();

puts("Введите максимальное значение элемента массива:");

long long max = input();

check\_range(min, max);

for (size\_t i = 0; i < rows; i++) {

random\_array\_filling(array[i], min, max, columns);

}

}

void input\_array\_filling(long long \*array, const size\_t columns) {

for (size\_t i = 0; i < columns; i++) {

array[i] = input();

}

}

void input\_2d\_array\_filling(long long\*\* array, const size\_t rows, const size\_t columns) {

for (size\_t i = 0; i < rows; i++) {

input\_array\_filling(array[i], columns);

}

}

void print\_array(const long long\* array, const size\_t columns) {

for (size\_t i = 0; i < columns; i++) {

printf("%lld ", array[i]);

}

}

void print\_2d\_array(const long long\*\* array, const size\_t rows, const size\_t columns) {

for (size\_t i = 0; i < rows; i++) {

print\_array(array[i], columns);

puts("\n");

}

}

void first\_3\_elements\_to\_root(long long\* array, const size\_t columns) {

for (size\_t i = 0; i < 3 && i < columns; i++) {

array[i] \*= array[i];

}

}

void first\_3\_columns\_to\_root(long long\*\* array, const size\_t rows, const size\_t columns) {

for (size\_t i = 0; i < rows; i++) {

first\_3\_elements\_to\_root(array[i], columns);

}

}

size\_t odd\_rows\_count(long long\*\* array, const size\_t rows) {

size\_t new\_rows = rows;

for (size\_t i = 0; i < rows; i++) {

if (i % 2 == 0)

new\_rows++;

}

return new\_rows;

}

void copy\_row(long long\* new\_array, const long long\* array, const size\_t columns) {

for (size\_t i = 0; i < columns; i++) {

new\_array[i] = array[i];

}

}

long long\*\* create\_an\_extended\_2d\_array(long long\*\* array, const size\_t rows, const size\_t new\_rows, const size\_t columns) {

size\_t current = 0;

long long\*\* new\_array = create\_2d\_array(new\_rows, columns);

for (size\_t i = 0; i < rows; i++) {

copy\_row(new\_array[current], array[i], columns);

current++;

if (i % 2 == 0) {

copy\_row(new\_array[current], array[0], columns);

current++;

}

}

return new\_array;

}

void free\_2d\_array(long long\*\* array, const size\_t rows) {

for (size\_t i = 0; i < rows; i++) {

free(array[i]);

}

free(array);

}

1. Результаты выполнения программы

Ниже представлены:

* Результаты выполнения программы (Рисунок 20, Рисунок 21).
* Результаты тестирования программы (Рисунок 22, Рисунок 23, Рисунок 24, Рисунок 25, Рисунок 26, Рисунок 27, Рисунок 28, Рисунок 29).

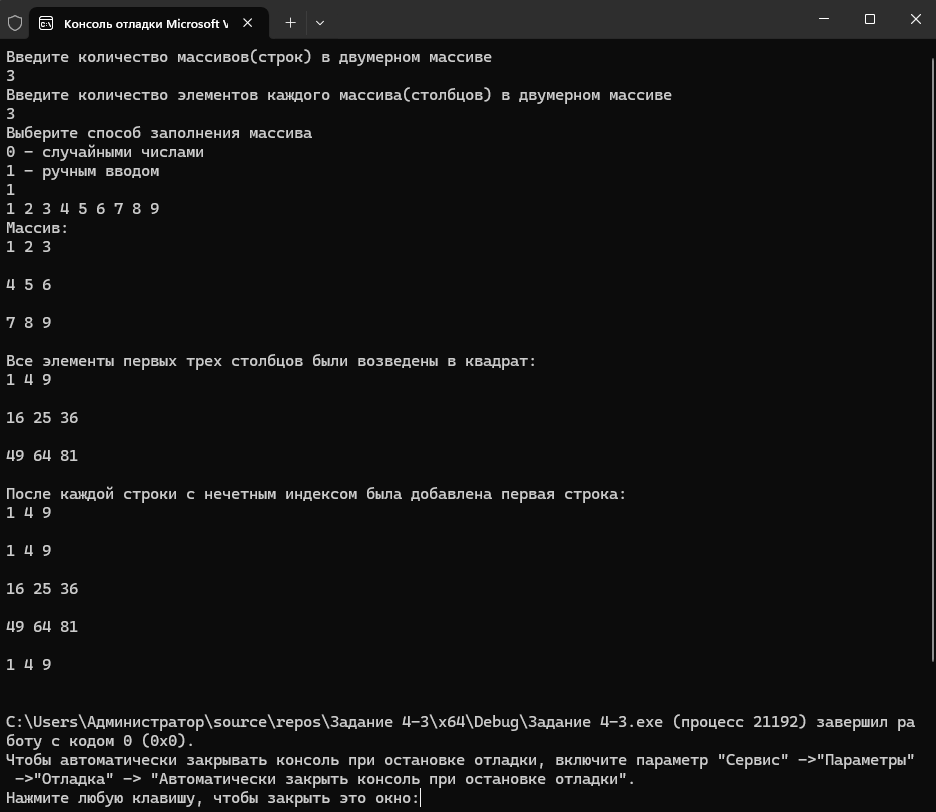


Рисунок 20 – Результат выполнения программы при ручном вводе данных

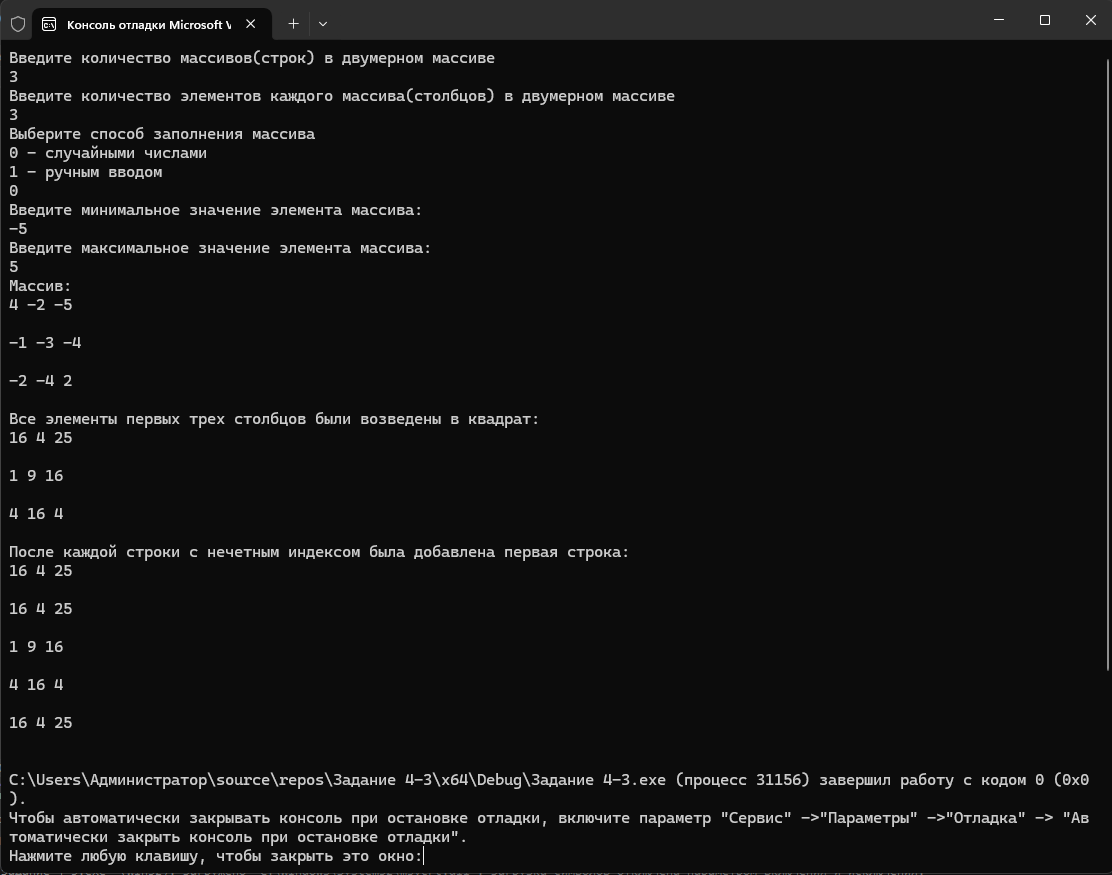


Рисунок 21 – Результат выполнения программы при случайной генерации входных данных

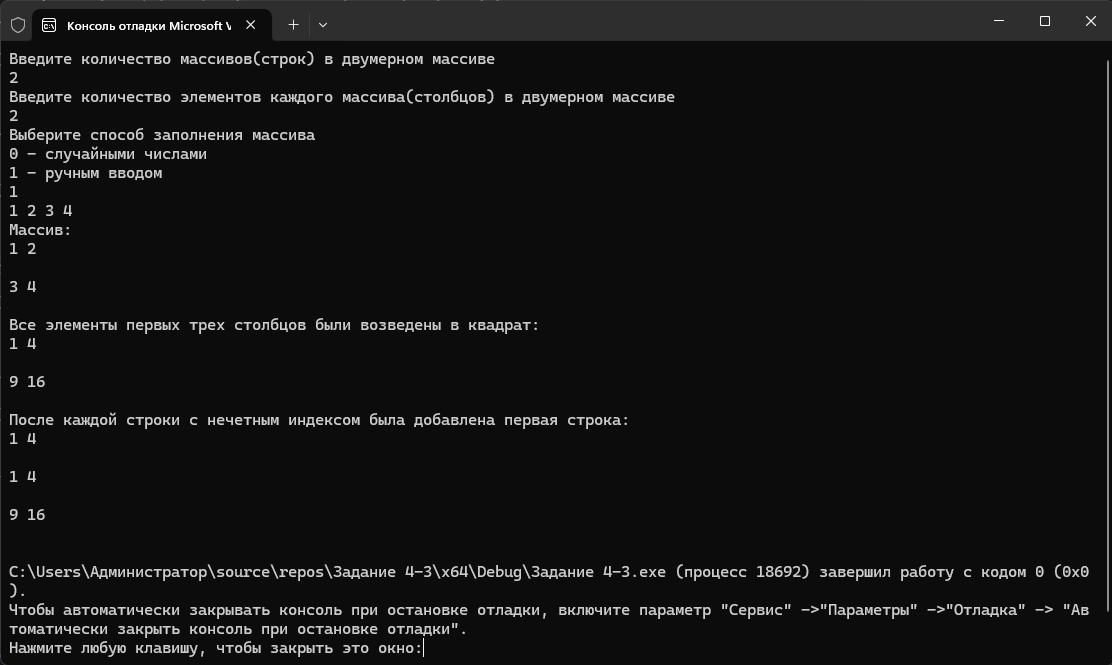


Рисунок 22 – Результат выполнения программы когда размер одномерных массивов меньше 3

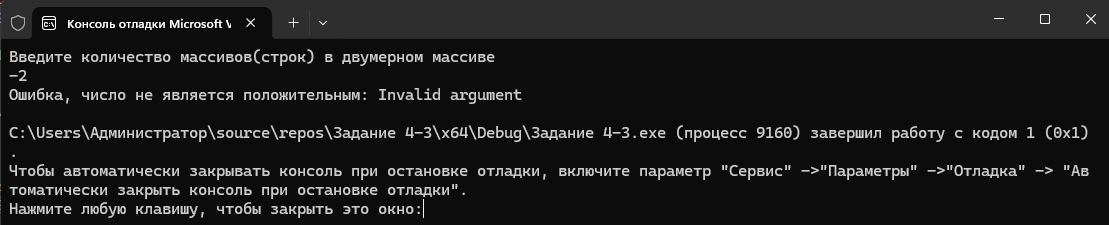


Рисунок 23 – Ввод отрицательного значения размера двумерного массива

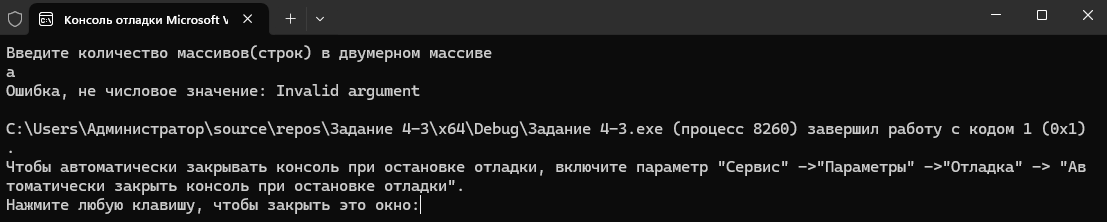


Рисунок 24 – Ввод символьного значения размера двумерного массива

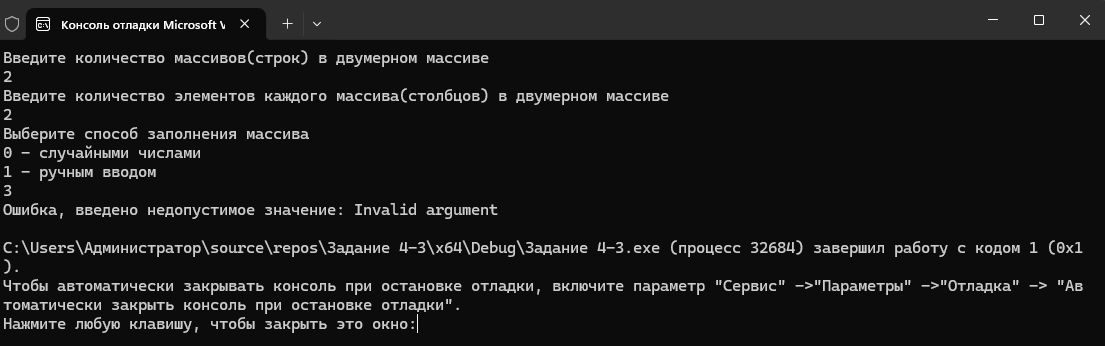


Рисунок 25 – Выбор несуществующего способа заполнения массива

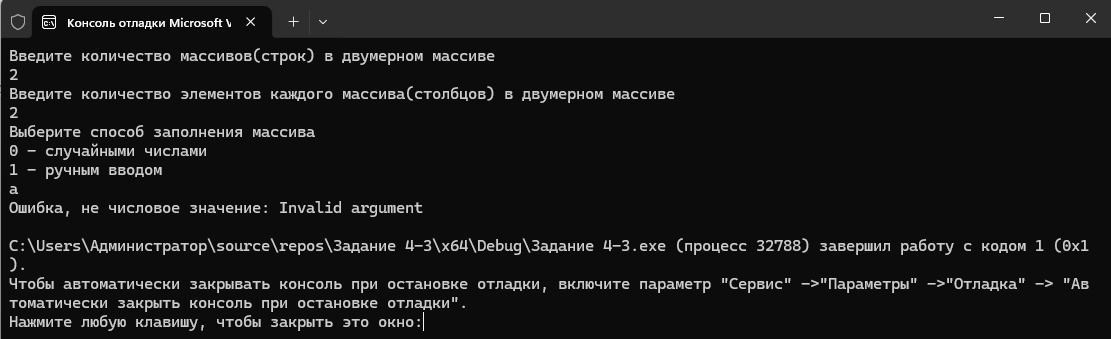


Рисунок 26 – Ввод символьного значения при выборе способа заполнения массива

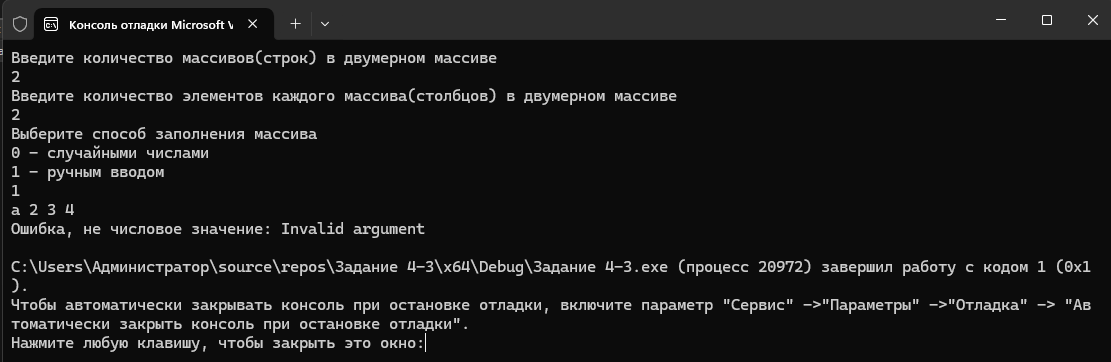


Рисунок 27 – Ввод символьного значения в качестве элемента одномерного массива

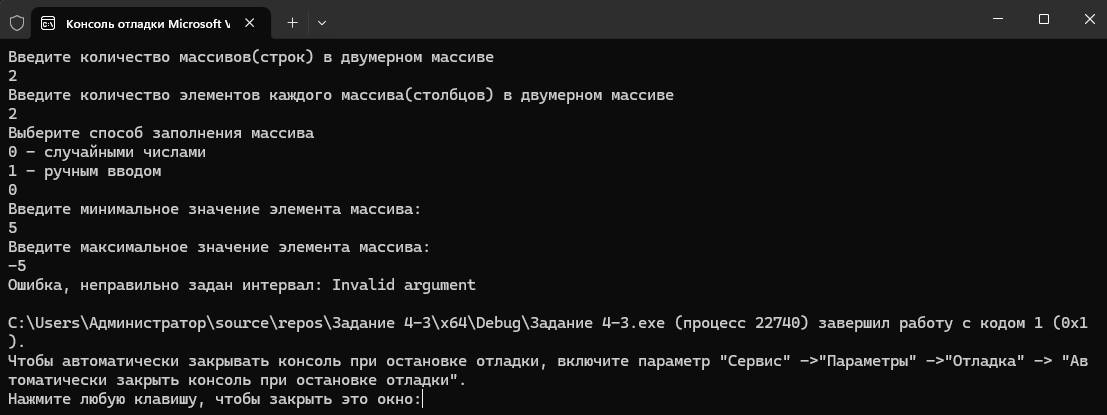


Рисунок 28 – Ввод некорректного интервала

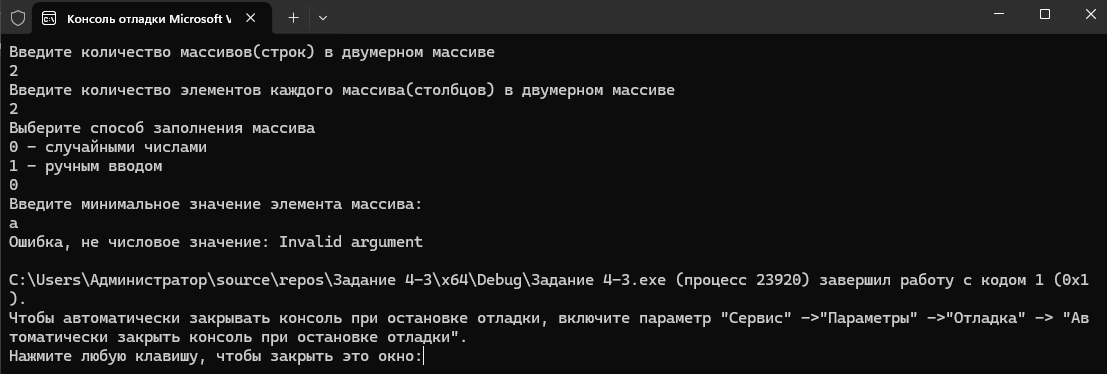


Рисунок 29 – Ввод символьного значения при задании интервала

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

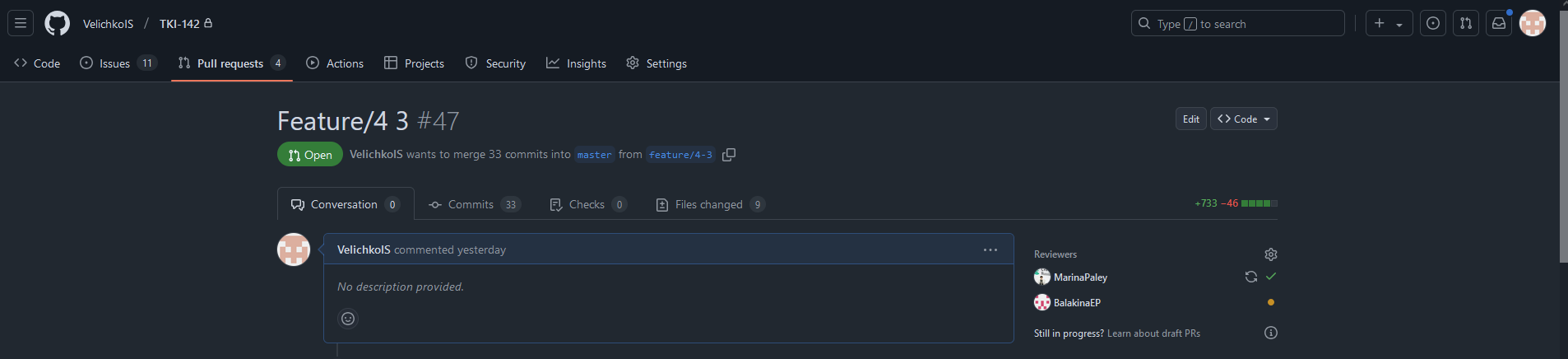


Рисунок – Отметка о выполнение задания