Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Факультет безопасности информационных технологий

Дисциплина:

«Программирование»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2

«Указатели и динамическая память»

Вариант 5-12

Выполнил:

Суханкулиев Мухаммет,

студент группы N3146

(подпись)

Проверила:

Сыдыкова Эмилия,

преподаватель программирования

(отметка о выполнении)

(подпись)

Санкт-Петербург

2023 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель работы – Разработать на языке С для ОС Linux программу, которая выполняет заданную операцию над матрицей чисел заданного типа.

Для достижения поставленной цели необходимо решить следующие задачи:

- Выполнить задание;
- Протестировать программу;
- Заархивировать папку проекта.

1 УКАЗАТЕЛИ И ДИНАМИЧЕСКАЯ ПАМЯТЬ

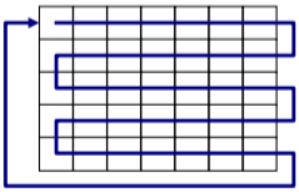
Выполнение преобразования числа в соответствии с вариантом 5-12.

1.1 Задание

Способ представления матрицы – VLA.

Преобразование – Осуществить в матрице циклический сдвиг на T элементов так, как показано на Рис. 1а.

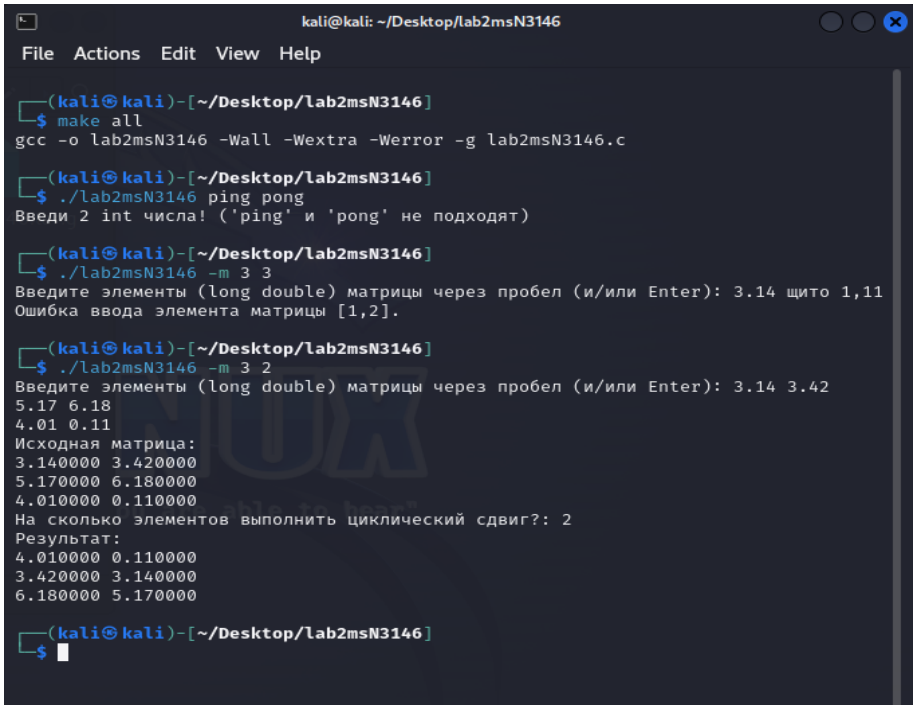
Тип числа – long double.

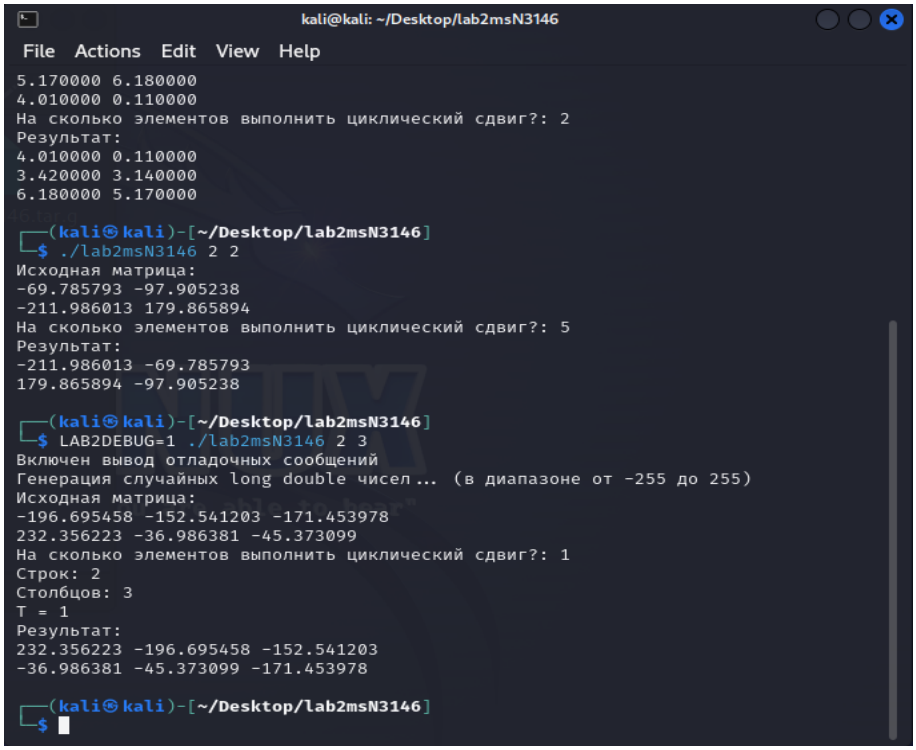


1.2 Make-файл

```
.PHONY: all clean

APP=lab2msN3146
CFLAGS=-Wall -Wextra -Werror -g

all: $(APP)

$(APP): $(APP).c
	gcc -o $(APP) $(CFLAGS) $(APP).c

clean:
	rm $(APP)
```

1.2.1 Примеры работы программы





1.3 Исходный текст программы

1.3.1 .c-файл

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
```

```
#include <time.h>

#include "functions.h"

int main(int argc, char *argv[]) {
    char *DEBUG = getenv("LAB2DEBUG");
    if (DEBUG) {
        fprintf(stderr, "Включен вывод отладочных сообщений\n");
    }

    if ((argc != 4 || strcmp(argv[1], "-m")) && (argc != 3)) {
        fprintf(stderr, "Usage: %s [-m] число_строк число_столбцов\n", argv[0]);
        return EXIT_FAILURE;

    int rows, columns, t;
    // Проверка входных данных
    if (argc == 3) {
        // Если введено 3 элемента, то проверить, что второй и третий - положительные
числа.
        rows = atoi(argv[1]);
        columns = atoi(argv[2]);
        if (rows > 0 && columns > 0) {
            // Создание матрицы
            long double **matrix = malloc(rows * sizeof(long double *));
            for (int i = 0; i < rows; i++) {
                matrix[i] = malloc(columns * sizeof(long double));
            }
            // Заполнение матрицы случайными long double значениями
            srand(time(NULL));
            for (int i = 0; i < rows; i++) {
                for (int j = 0; j < columns; j++) {
                    matrix[i][j] = getRandomLD(-255, 255);
                }
            }
            // Вывод
            if (DEBUG) printf("Генерация случайных long double чисел... (в диапазоне от
-255 до 255)\n");
            printf("Исходная матрица:\n");
            print_matrix(matrix, rows, columns);
            // Получить значение T
            t = get_t(rows,columns,DEBUG);
            if (t == 69) { // Вывод ошибки если функция возвратила 69
                return EXIT_FAILURE;
            }

            // Преобразование матрицы
            shift(matrix, rows, columns, t);

            // Вывод матрицы на экран
            printf("Результат:\n");
            print_matrix(matrix, rows, columns);
            // Освобождение памяти
            for (int i = 0; i < rows; i++) {
                free(matrix[i]);
            }
            free(matrix);
        } else {
            // Вывести сообщение об ошибке.
            fprintf(stderr, "Введи 2 int числа! ('%s' и '%s' не подходят)\n",
argv[1],argv[2]);
            return EXIT_FAILURE;
        }
```

```
    } else if (argc == 4) {
        if (strcmp(argv[1], "-m") == 0) {
            rows = atoi(argv[2]);
            columns = atoi(argv[3]);
            if (rows > 0 && columns > 0) {
                // Создание матрицы
                long double **matrix = malloc(rows * sizeof(long double *));
                for (int i = 0; i < rows; i++) {
                    matrix[i] = malloc(columns * sizeof(long double));
                }
                // Заполнение матрицы
                printf("Введите элементы (long double) матрицы через пробел (и/или
Enter): ");
                for (int i = 0; i < rows; i++) {
                    for (int j = 0; j < columns; j++) {
                        // Проверка на корректность ввода
                        if (scanf("%Lf", &matrix[i][j]) != 1) {
                            // Вывод ошибки
                            printf("Ошибка ввода элемента матрицы [%d,%d].\n", i+1,
j+1);
                            return EXIT_FAILURE;
                        }
                    }
                }
                // Вывод
                printf("Исходная матрица:\n");
                print_matrix(matrix, rows, columns);
                // Получить значение T
                t = get_t(rows,columns,DEBUG);
                if (t == 69) {
                    return EXIT_FAILURE;
                }

                // Преобразование
                shift(matrix, rows, columns, t);

                // Вывод матрицы на экран
                printf("Результат:\n");
                print_matrix(matrix, rows, columns);
                // Освобождение памяти
                for (int i = 0; i < rows; i++) {
                    free(matrix[i]);
                }
                free(matrix);
            } else {
                // Вывести сообщение об ошибке.
                fprintf(stderr, "Даже вместе с '-m' нужно ввести 2 int числа!\n");
                return EXIT_FAILURE;
            }
        }
    }

    return EXIT_SUCCESS;
}
```

1.3.2 .h-файл

```
#ifndef FUNCTIONS_H
#define FUNCTIONS_H
```

```
void print_matrix(long double **matrix, int rows, int columns) {
  for (int i = 0; i < rows; i++) {
    for (int j = 0; j < columns; j++) {
      char buffer[100];
      sprintf(buffer, "%Lf", matrix[i][j]);
      printf("%*s ", columns, buffer);
    }
    printf("\n");
  }
}

int get_t(int rows, int columns, char* D){
  int t;
  if (D) {
    printf("На сколько элементов выполнить циклический сдвиг?:\n");
    // прочесть t
    scanf("%d", &t);
    printf("Строк: %d\nСтолбцов: %d\nT = %d\n",rows,columns,t);
  } else {
    printf("На сколько элементов выполнить циклический сдвиг?:\n");
    scanf("%d", &t);
  }
  // Проверка на корректность ввода
  if (t>0 && t<100) {
    t %= rows * columns;
    return t;
  } else {
    // Вывод ошибки
    printf("Значение T должно быть положительным целым числом.\n");
    t = 69; // ;)
    return t;
  }
}

long double getRandomLD(long double min, long double max) {
    return min + (max - min) * ((long double)rand() / (long double)RAND_MAX);
}

void shift(long double **matrix, int rows, int cols, int t) {
  // Преобразование матрицы в вектор по строкам
  long double vector[rows * cols];
```

```
  int index = 0;
  for (int i = 0; i < rows; i++) {
    if (i % 2 == 0) {
      for (int j = 0; j < cols; j++) {
        vector[index++] = matrix[i][j];
      }
    } else {
        for (int j = cols - 1; j >= 0; j--) {
          vector[index++] = matrix[i][j];
        }
      }
  }
  // Сдвиг вектора на t элементов вправо
  long double shiftedVector[rows * cols];
  for (int i = 0; i < rows * cols; i++) {
    int newIndex = (i + t) % (rows * cols);
    shiftedVector[newIndex] = vector[i];
  }
  // Преобразование вектора обратно в матрицу
  index = 0;
  for (int i = 0; i < rows; i++) {
    if (i % 2 == 0) {
      for (int j = 0; j < cols; j++) {
        matrix[i][j] = shiftedVector[index++];
      }
    } else {
        for (int j = cols - 1; j >= 0; j--) {
          matrix[i][j] = shiftedVector[index++];
        }
      }
  }
} // Было сложно, но я понял это и написал эту функцию... за 3 дня...

#endif
```

ЗАКЛЮЧЕНИЕ

Был выполнен сдвиг long double элементов матрицы на T элементов, представленной способом VLA.

Это позволило закрепить навыки работы с указателями и динамической памятью.

Я все сделал, я молодец. =)

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Лекции Гирика Алексея Валерьевича по программированию – 2023. – URL : https://drive.google.com/drive/folders/1eAiMW4hD9TLhZH2vtpKPWzZkzKp10BnL (дата обращения: 29.11.2023).
2. Открытые онлайн-источники по программированию – URL: https://www.progler.ru/, http://stdufile.net/, https://studassistent.ru/, https://ru.stackoverflow.com/ и т.д. (дата обращения: 02.12.2023).