|  |  |
| --- | --- |
|  | **Отчёт по лабораторной работе** № 15 по курсу 1  Фундаментальная информатика  Студент группы M8O-101Б-23 Терентьев Михаил Андреевич, № по списку: 23  Контакты www, e-mail, icq, skype mihateren2003@gmail.com  Работа выполнена: “11 “ декабря 2023г.  Преподаватель: каф. 806 Крылов Сергей Сергеевич  Входной контроль знаний с оценкой  Отчёт сдан “13“ декабря 2023г., итоговая оценка  Подпись преподавателя |

1. **Тема**: Вложенные циклы с параметрами. Обход и линеаризация матриц
2. **Цель работы**: Освоить работу с матрицами в языке Си.
3. **Задание**:   
   
4. **Оборудование** *Процессор* ***Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz 1.80 GHz*** *с ОП 8 ГБ , ТТН* ***512 ГБ*** *Встроенный монитор 13,2", FHD (1366 x 768) 16:9, частота обновления: 60 Гц*
5. *Другие устройства*
6. **Программное обеспечение:** *ПВЭМ студента* (*лабораторное*):

Программное обеспечение ПЭВМ студента, если использовалось:

Операционная система семейства Windows, наименование версия Windows 11,

Компилятор gcc. Система программирования C.

Редактор текстов Visual Studio Code версия 1.85

1. **Идея, метод, алгоритм** *решения задачи (в формах: словесной, псевдокода, графической [блок-схема, диаграмма, рисунок, таблица] или формальное описание с пред- и постусловиями):*

1. Функция checkSize(const char \*filename) открывает файл с именем filename и считывает первую строку, разбивая ее на отдельные слова. Количество считанных слов определяет размер матрицы, который затем возвращается из функции.

2. Функция createDynamicMatrix(int n) выделяет память под двумерный массив размера n x n с помощью динамического выделения памяти.

3. Функция readMatrix(int\*\* matrix, int size, const char \*filename) открывает файл с именем filename для чтения и считывает содержимое файла в двумерный массив matrix размера size x size.

4. Функция initArray(int size) выделяет память под одномерный массив размера size и инициализирует его нулями.

5. Функция findMinElem(int\*\* matrix, int size) находит минимальный элемент в матрице размера size x size.

6. Функция findMinStr(int\*\* matrix, int\* line, int size, int minim) определяет строки, в которых содержится минимальный элемент, и устанавливает соответствующие значения в массиве line.

7. Функция reverseStr(int\*\* matrix, int\* line, int size) реверсирует строки матрицы, в которых содержится минимальный элемент, с помощью обмена значений элементов.

8. Функция printMatrix(int\*\* matrix, int size) выводит содержимое матрицы размера size x size на экран.

9. Функция freeMatrix(int\*\* matrix, int size) освобождает память, выделенную для двумерного массива, и одномерного массива line.

В функции main() пользователю предлагается ввести название файла. Затем программа использует описанные выше функции для чтения матрицы из файла, нахождения минимального элемента, реверсирования строк с минимальным элементом и вывода результата на экран. После этого программа освобождает выделенную память и завершает свою работу.

1. **Сценарий выполнения работы** [план работы, первоначальный текст программы в черновике (можно на отдельном листе) и тесты либо соображения по тестированию].

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

int checkSize(const char \*filename) {

    FILE \*file;

    char line[100]; // Максимальная длина строки

    int count = 0;

    file = fopen(filename, "r");

    if (file == NULL) {

        printf("Ошибка открытия файла\n");

        return -1; // Возвращаем -1 в случае ошибки открытия файла

    }

    if (fgets(line, 100, file) != NULL) { // Считываем первую строку из файла

        char \*token = strtok(line, " "); // Разбиваем строку на отдельные слова

        while (token != NULL) {

            count++;

            token = strtok(NULL, " ");

        }

    }

    fclose(file);

    return count;

}

int\*\* createDynamicMatrix(int n) {

    int\*\* array = (int\*\*)malloc(n \* sizeof(int\*)); // Выделяем память под указатели на строки

    for (int i = 0; i < n; i++) {

        array[i] = (int\*)malloc(n \* sizeof(int)); // Выделяем память под каждую строку

    }

    return array;

}

int readMatrix(int\*\* matrix, int size, const char \*filename) {

    FILE\* f = fopen(filename, "r"); // Открываем файл для чтения

    if (f == NULL) {

        printf("Ошибка открытия файла\n");

        return -1; // Возвращаем -1 в случае ошибки открытия файла

    }

    fseek(f, 0, SEEK\_SET); // Ставим указатель в начало файла для считывания

    for (int i = 0; i < size; i++) {

        for (int j = 0; j < size; j++)

            fscanf(f, "%d", &matrix[i][j]);

    }

    fclose(f);

}

int\* initArray(int size) {

    int \*arr = (int \*)malloc(size \* sizeof(int));

    for (int i = 0; i < size; i++) {

        arr[i] = 0;

    }

    return arr;

}

int findMinElem(int\*\* matrix, int size) {

    int min = INT\_MAX;

    int i,j;

    for (i = 0; i < size; i++){

        for (j = 0; j < size; j++){

            if (matrix[i][j] < min){

                min = matrix[i][j];

            }

        }

    }

    return min;

}

void findMinStr(int\*\* matrix, int\* line, int size, int minim) {

    for (int i = 0; i < size; i++){

        for (int j = 0; j < size; j++){

            if (minim == matrix[i][j]){

                line[i] = 1;

                break;

            }

        }

    }

}

void printMatrix(int\*\* matrix, int size) {

    int i, j;

    for (i = 0; i < size; i++){

        for (j = 0; j < size; j++){

            printf("%d ", matrix[i][j]);

        }

        printf("\n");

    }

}

void reverseStr(int\*\* matrix, int\* line, int size) {

    for (int i = 0; i < size; i++){

        int start = 0;

        int end = size - 1;

        while (start < end) {

            int temp = matrix[i][start];

            matrix[i][start] = matrix[i][end];

            matrix[i][end] = temp;

            start++;

            end--;

        }

    }

}

void freeMatrix(int\*\* matrix, int size) {

    for (int i = 0; i < size; i++) {

        free(matrix[i]);

    }

    free(matrix);

}

int main() {

    char filename[256];

    printf("Введите название файла: ");

    scanf("%s", filename);

    int size = checkSize(filename); // узнаем порядок матрицы

    int\*\* matrix = createDynamicMatrix(size); // создаем двумерный массив заданного порядка

    readMatrix(matrix, size, filename); // считываем в матрицу данные из файла

    printf("Изначальная матрица: \n");

    printMatrix(matrix, size);

    printf("\n");

    int\* line = initArray(size); // инициализируем массив, инициализированный нулем

    int minim = findMinElem(matrix, size); // находим минимальный элемент матрицы

    findMinStr(matrix, line, size, minim); // находим строки, в которых содержится минимальный элемент

    reverseStr(matrix, line, size); // реверсим строки с минимальным элементом

    printf("Матрица после преобразований: \n");

    printMatrix(matrix, size); // смотрим результат

    freeMatrix(matrix, size); // очищаем память

    free(line);

    return 0;

}

Допущен к выполнению работы**. Подпись преподавателя**

1. **Распечатка протокола** (подклеить листинг окончательного варианта программы с текстовыми примерами, подписанный преподавателем)

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

int checkSize(const char \*filename) {

    FILE \*file;

    char line[100]; // Максимальная длина строки

    int count = 0;

    file = fopen(filename, "r");

    if (file == NULL) {

        printf("Ошибка открытия файла\n");

        return -1; // Возвращаем -1 в случае ошибки открытия файла

    }

    if (fgets(line, 100, file) != NULL) { // Считываем первую строку из файла

        char \*token = strtok(line, " "); // Разбиваем строку на отдельные слова

        while (token != NULL) {

            count++;

            token = strtok(NULL, " ");

        }

    }

    fclose(file);

    return count;

}

int\*\* createDynamicMatrix(int n) {

    int\*\* array = (int\*\*)malloc(n \* sizeof(int\*)); // Выделяем память под указатели на строки

    for (int i = 0; i < n; i++) {

        array[i] = (int\*)malloc(n \* sizeof(int)); // Выделяем память под каждую строку

    }

    return array;

}

int readMatrix(int\*\* matrix, int size, const char \*filename) {

    FILE\* f = fopen(filename, "r"); // Открываем файл для чтения

    if (f == NULL) {

        printf("Ошибка открытия файла\n");

        return -1; // Возвращаем -1 в случае ошибки открытия файла

    }

    fseek(f, 0, SEEK\_SET); // Ставим указатель в начало файла для считывания

    for (int i = 0; i < size; i++) {

        for (int j = 0; j < size; j++)

            fscanf(f, "%d", &matrix[i][j]);

    }

    fclose(f);

}

int\* initArray(int size) {

    int \*arr = (int \*)malloc(size \* sizeof(int));

    for (int i = 0; i < size; i++) {

        arr[i] = 0;

    }

    return arr;

}

int findMinElem(int\*\* matrix, int size) {

    int min = INT\_MAX;

    int i,j;

    for (i = 0; i < size; i++){

        for (j = 0; j < size; j++){

            if (matrix[i][j] < min){

                min = matrix[i][j];

            }

        }

    }

    return min;

}

void findMinStr(int\*\* matrix, int\* line, int size, int minim) {

    for (int i = 0; i < size; i++){

        for (int j = 0; j < size; j++){

            if (minim == matrix[i][j]){

                line[i] = 1;

                break;

            }

        }

    }

}

void printMatrix(int\*\* matrix, int size) {

    int i, j;

    for (i = 0; i < size; i++){

        for (j = 0; j < size; j++){

            printf("%d ", matrix[i][j]);

        }

        printf("\n");

    }

}

void reverseStr(int\*\* matrix, int\* line, int size) {

    for (int i = 0; i < size; i++){

        int start = 0;

        int end = size - 1;

        while (start < end) {

            int temp = matrix[i][start];

            matrix[i][start] = matrix[i][end];

            matrix[i][end] = temp;

            start++;

            end--;

        }

    }

}

void freeMatrix(int\*\* matrix, int size) {

    for (int i = 0; i < size; i++) {

        free(matrix[i]);

    }

    free(matrix);

}

int main() {

    char filename[256];

    printf("Введите название файла: ");

    scanf("%s", filename);

    int size = checkSize(filename); // узнаем порядок матрицы

    int\*\* matrix = createDynamicMatrix(size); // создаем двумерный массив заданного порядка

    readMatrix(matrix, size, filename); // считываем в матрицу данные из файла

    printf("Изначальная матрица: \n");

    printMatrix(matrix, size);

    printf("\n");

    int\* line = initArray(size); // инициализируем массив, инициализированный нулем

    int minim = findMinElem(matrix, size); // находим минимальный элемент матрицы

    findMinStr(matrix, line, size, minim); // находим строки, в которых содержится минимальный элемент

    reverseStr(matrix, line, size); // реверсим строки с минимальным элементом

    printf("Матрица после преобразований: \n");

    printMatrix(matrix, size); // смотрим результат

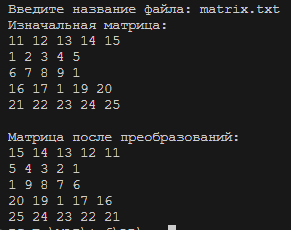
    freeMatrix(matrix, size); // очищаем память

    free(line);

    return 0;

}

*Тесты:*



1. **Дневник отладки** должен содержать дату и время сеансов отладки, и основные ошибки (ошибки в сценарии и программе, не стандартные операции) и краткие комментарии к ним. В дневнике отладки приводятся сведения об использовании других ЭВМ, существенном участии преподавателя и других лиц в написании и отладке программы.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | Лаб.  или  дом. | Дата | Время | Событие | Действие по исправлению | Примечание |
|  |  |  |  |  |  |  |

1. **Замечание автора по существу работы**: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
2. **Выводы**

В ходе выполнения лабораторной работы я научился работе с матрицами на языке Си.

Подпись студента \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_