МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Московский Авиационный Институт

(Национальный Исследовательский Университет)»

ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И ПРИКЛАДНОЙ МАТЕМАТИКИ

Кафедра вычислительной математики и программирования

**Курсовой проект**

по курсу «Практикум на ЭВМ» 2 семестра

Задание 7. Разреженные матрицы

Выполнил: студент 108 группы

Саженов Константин Станиславович

Научный руководитель:

Поповкин Александр Викторович

Москва, 2020

**Содержание**

[Задача 3](#__RefHeading___Toc496_579521979)

[Общий метод решения 4](#__RefHeading___Toc498_579521979)

[Общие сведения о программе 5](#__RefHeading___Toc500_579521979)

[Функциональное назначение 6](#__RefHeading___Toc502_579521979)

[Описание программы 7](#__RefHeading___Toc504_579521979)

[Описание функции программы 9](#__RefHeading___Toc506_579521979)

[Используемые переменные 10](#__RefHeading___Toc508_579521979)

[Входные значения 11](#__RefHeading___Toc510_579521979)

[Заключение 30](#__RefHeading___Toc512_579521979)

[Список используемых источников 31](#__RefHeading___Toc514_579521979)

# **Задача**

**Общее задание**

Составить программу на языке Си с процедурами и/или функциями для обработки прямоугольных разряженных матриц с элементами целого типа, которая:

1. Вводит матрицы различного размера, представленные во входном текстовом файле в обычном формате (по строкам), с одновременным размещением ненулевых элементов в разряженной матрице в соответствии с заданной схемой;
2. Печатает введенные матрицы во внутреннем представлении согласно заданной схеме размещения и в обычном (естественном) виде;
3. Выполняет необходимые преобразования разреженных матриц (или вычисления над ними) путем обращения к соответствующим процедурам и/или функциям;
4. Печатает результат преобразования (вычисления) согласно заданной схеме размещения в обычном виде.

В процедурах и функциях предусмотреть проверки и печать сообщений в случаях ошибок в задании параметров.

**Вариант 22**

Таблица 1 – задание.

|  |  |
| --- | --- |
| Схема размещения матрицы | Один вектор (все матрицы m x n хранятся по строкам, в порядке возрастания индексов ненулевых элементов) |
| Преобразование | Транспонировать разреженную матрицу относительно побочной диагонали. Выяснить, является ли полученная матрица кососимметрической |
| Физическое представление | Отображение на динамические структуры |



# **Общий метод решения**

Создаем текстовый файл, в который записываем исходную разреженную матрицу. Создаем несколько функций по считыванию матрицы из файла, ее размещению по заданной схеме и преобразованию, а также выводу. Предусматриваем возможные ошибки, которые могут возникнуть при невыполнении некоторых функций, и возможность выбора действий с матрицей пользователем.

# **Общие сведения о программе**

Аппаратное обеспечение: ноутбук;

Операционная система: Arch Linux rolling release;

Язык и система программирования: GNU C;

Число строк: 408;

Компиляция программы в терминале: gcc -I. -O2 matrix.c main.c

Вызов программы: ./a.out;

# **Функциональное назначение**

Основная задача программы состоит в прочтении текстового файла и в дальнейшей работе с ним. В файле содержится исходная разреженная матрица, с которой программа будет работать. В зависимости от выбора, который совершает пользователь, меняется и действие, которое будет совершаться над матрицей: она может считываться из текстового файла, размещаться по схеме «один вектор», транспонирование матрицы(по побочной диагонали и по главной диагонали), вывод матрицы в обычном представлении и в формате схемы «один вектор», проверка матрицы на кососимметричность.

# **Описание программы**

1. Подключение библиотеки <stdlib.h> и <stdio.h>;
2. Создание некоторых функций:

* Matrix \*init():

Инициализация матрицы;

* void enter(Matrix \*m, char\*filename):

Ввод матрицы;

* void print(Matrix \*m)

вывод матрицы в стандартном виде**;**

* void print\_inner(Matrix \*m)

вывод матрицы в виде схемы размещения;

* int max().

преобразование: транспонируем матрицу по побочной диагонали**;**

* void push\_back(Matrix \*m, int element)

добавление элемента в конец списка(в котором представлена матрица)

* void next(Node \*node) — следущий элемент
* void prev(Node \*node) — предыдущий элемент
* void insert\_xy(Matrix \*m, int x, int y, int raw\_index, int raw\_data)

Вставка двух элементов(по схеме размещения) в матрицу по координатам x, y

* void print\_row\_zeros(int n) — вспомогательная функция. Печатает n нулей
* bool skew\_symmetric(Matrix \*m) — проверяет матрицу на кососимметричность
* Matrix \*transpose(Matrix \*m) — транспонирует матрицу
* Matrix \*transpose\_by\_side\_diagonal(Matrix \*m) — транспонирует матрицу по побочной диагонали
* bool matrices\_equal(Matrix\*m1, Matrix \*m2) — проверяет матрицы на равенство
* Matrix\* negativeMatrix(Matrix \*m) — возвращает отрицание данной матрицы
* void incorrectFilename():

вывод сообщения об ошибке;

1. С помощью switch реализовать выбор пользователем действий с матрицей:

* «1» – считывание матрицы из текстового файла;
* «2» – вывод матрицы в обычном виде;
* «3» - вывод матрицы согласно схеме;
* «4» - транспонирование матрицы;
* «5» - транспонирование матрицы по побочной диагонали
* «6» - проверяет, является ли матрица кососимметричной
* «7» - выход из программы.

# **Описание функции программы**

Таблица 2 - функции программы.

|  |  |
| --- | --- |
| printf | Вывод текста в заданном формате |
| calloc | Выделение памяти |
| scanf | Чтение с консоли |
| FILE \*file | Объявление файлового потока |
| file = fopen(filename, “r”) | Открытие файлового поткока ввода из файла, имя которого находится в переменной filename |
| switch(choice){  case 1:  break;  } | Обрабатывает ввод с клавиатуры при работе с меню(в частности, обрабатывает поступившие варианты ) |
| fseek(file, 0, SEEK\_SET) | Начинает читать текстовый файл с самого начала |
| puts | Вывод текста на экран |
| fgetc(file) | Считыает один символ из файла |
| feof(file) | Проверка на то, закрыт ли файл |
| fscanf(file, ..) | Считывание данных с файла |
| fclose(file) | Закрытие файлового потока |
| goto end | Переход к пометке end |

# **Используемые переменные**

Таблица 3 – переменные.

|  |  |  |
| --- | --- | --- |
| Переменная | Тип | Значение |
| choice | int | Выбранный пользователем пункт меню |
| x,y | Координаты элементов |
| elem | Элемент матрицы |
| rows | Количество строк в матрице |
| cols | Количество столбцов в матрице |
| totalCols | Общее количество столбцов в матрице |
| filename | char\* | Название файла |
| m | Matrix\* | Исходная матрица |
| elem | Node\* | Элемент матрицы в виде узла в списке |
| file | FILE\* | Считываемый файл |
| flag | Node\* | Флаг, который показывает, какие данные за ним следуют |
| data | Node\* | Сами данные |
| nm, m2 | Matrix\* | Создаваемая матрица |

# **Входные значения**

Test1.txt:

0 1

-1 0

Test2.txt:

1 2 3 4

0 0 0 0

1 0 0 0

1 0 0 1

1 2 3 4

76 0 0 0

Test3.txt:

6 4 1

0 0 0

0 0 0

1 2 4

Test4.txt

0 0 0 1

0 0 1 0

0 -1 0 0

-1 0 0 0

**Протокол**

sakost@sakost-pc  ~/projects/CLionProjects/lab/cmake-build-debug/cw7  ./cw7

Print name of file: ../../cw7/test1.txt

1. Read matrix.

2. Print matrix.

3. Print inner data of matrix

4. Transpose matrix.

5. Transpose by side-diagonal matrix.

6. Check the whether the matrix is skew-matrix.

7. Quit.

1

The matrix was successfully read

2

0 1

-1 0

6

Matrix is skew-symmetric.

2

0 1

-1 0

3

0 1 2 1 0 2 1 -1 0 0

4

2

0 -1

1 0

5

2

0 -1

1 0

6

Matrix is skew-symmetric.

7

The program is closed

sakost@sakost-pc  ~/projects/CLionProjects/lab/cmake-build-debug/cw7  ./cw7

Print name of file: ../../cw7/test2.txt

1. Read matrix.

2. Print matrix.

3. Print inner data of matrix

4. Transpose matrix.

5. Transpose by side-diagonal matrix.

6. Check the whether the matrix is skew-matrix.

7. Quit.

1

The matrix was successfully read

2

1 2 3 4

0 0 0 0

1 0 0 0

1 0 0 1

1 2 3 4

76 0 0 0

4

2

1 0 1 1 1 76

2 0 0 0 2 0

3 0 0 0 3 0

4 0 0 1 4 0

7

The program is closed

sakost@sakost-pc  ~/projects/CLionProjects/lab/cmake-build-debug/cw7  ./cw7

Print name of file: ../../test3.txt

1. Read matrix.

2. Print matrix.

3. Print inner data of matrix

4. Transpose matrix.

5. Transpose by side-diagonal matrix.

6. Check the whether the matrix is skew-matrix.

7. Quit.

1

The matrix was successfully read

2

0 0 0 1

0 0 1 0

0 -1 0 0

-1 0 0 0

4

2

0 0 0 -1

0 0 -1 0

0 1 0 0

1 0 0 0

5

2

0 0 0 -1

0 0 -1 0

0 1 0 0

1 0 0 0

6

Matrix is skew-symmetric.

7

1. Read matrix.

2. Print matrix.

3. Print inner data of matrix

4. Transpose matrix.

5. Transpose by side-diagonal matrix.

6. Check the whether the matrix is skew-matrix.

7. Quit.

The program is closed

sakost@sakost-pc  ~/projects/CLionProjects/lab/cmake-build-debug/cw7  cat ../../cw7/main.c

//

// Created by sakost on 23.04.2020.

//

#include <stdio.h>

#include <stdlib.h>

#include "matrix.h"

void incorrectFilename(){

puts("The filename is incorrect");

}

void menu(){

puts("1. Read matrix.");

puts("2. Print matrix.");

puts("3. Print inner data of matrix");

puts("4. Transpose matrix.");

puts("5. Transpose by side-diagonal matrix.");

puts("6. Check the whether the matrix is skew-matrix.");

puts("7. Quit.");

}

int main(){

int choice = 0;

printf("Print name of file: ");

char\* filename = (char\*)calloc(40, sizeof(char));

scanf("%s", filename);

FILE \*file = fopen(filename, "r");

if (!file) {

incorrectFilename();

return 1;

}

Matrix \*m = init();

menu();

while (choice < 7) {

scanf("%d", &choice);

switch (choice) {

case 1:

enter(m, filename);

printf("The matrix was successfully read\n\n");

break;

case 2:

print(m);

break;

case 3:

print\_inner(m);

break;

case 4:

m = transpose(m);

break;

case 5:

m = transpose\_by\_side\_diagonal(m);

break;

case 6:

if(skew\_symmetric(m)){

puts("Matrix is skew-symmetric.");

}else {

puts("Matrix is not skew-symmetric.");

}

break;

default:

menu();

break;

}

}

printf("The program is closed\n");

}

sakost@sakost-pc  ~/projects/CLionProjects/lab/cmake-build-debug/cw7  cat ../../cw7/matrix.

zsh: correct '../../cw7/matrix.' to '../../cw7/matrix.c' [nyae]? a

sakost@sakost-pc  ~/projects/CLionProjects/lab/cmake-build-debug/cw7  cat ../../cw7/matrix.c

//

// Created by sakost on 22.04.2020.

//

#include "matrix.h"

#include <stdio.h>

#include <stdlib.h>

Matrix\* init(){

Matrix\* m = (Matrix\*)malloc(sizeof(Matrix));

m->rows = m->cols = 0;

return m;

}

void push\_back(Matrix\* m, int element){

if(m->end == NULL){

m->start = m->end = (Node\*)malloc(sizeof(Node));

m->start->data = element;

m->start->next = m->start->prev = NULL;

return;

}

Node\* elem = m->end;

elem->next = (Node\*)malloc(sizeof(Node));

elem->next->data = element;

elem->next->next = NULL;

m->end = elem->next;

m->end->prev = elem;

}

void enter(Matrix\* m, char\* filename){

FILE\* file = fopen(filename, "r");

int elem, rows=1;

int totalCols = 0;

while(fscanf(file, "%d", &elem) != 0){

totalCols++;

if(fgetc(file) == '\n'){

break;

}

}

m->cols = totalCols;

fseek(file, 0, SEEK\_SET);

push\_back(m, 0);

push\_back(m, 1);

while(!feof(file)){

for (int i = 0; i < m->cols; i++){

if(fscanf(file, "%d", &elem) == EOF){

goto end;

}

if (elem != 0){

push\_back(m, i+1);

push\_back(m, elem);

}

}

push\_back(m, 0);

push\_back(m, ++rows);

m->rows = rows-1;

fseek(file, 1, SEEK\_CUR); // skip \n

}

end:

m->end->data = 0;

fclose(file);

}

Node\* next(Node\* node){

return node->next;

}

Node\* prev(Node\* node){

return node->prev;

}

void insert\_xy(Matrix \*m, int x, int y, int raw\_index, int raw\_data){

Node\* index = (Node\*)malloc(sizeof(Node)),\*data = (Node\*)malloc(sizeof(Node));

index->data = raw\_index; data->data = raw\_data;

index->next = data; data->prev = index;

index->prev = data->next = NULL;

Node\* cur\_index = m->start;

Node\* cur\_data = next(cur\_index);

int zero\_count = 0;

while(zero\_count != y){

if(cur\_index->data == 0) {

zero\_count++;

}

cur\_index = next(cur\_data);

cur\_data = next(cur\_index);

}

int col = cur\_index->data;

while (col < x && col > 0){

cur\_index = next(cur\_data);

cur\_data = next(cur\_index);

col = cur\_index->data;

}

Node\* prev\_cur\_data = prev(cur\_index);

// Node\* prev\_cur\_index = prev(prev\_cur\_data);

prev\_cur\_data->next = index;

index->prev = prev\_cur\_data;

data->next = cur\_index;

cur\_index->prev = data;

}

void print\_row\_zeros(int n){

for (int i = 0; i < n; ++i) {

printf("0 ");

}

}

void print(Matrix \*m){

Node \*flag = m->start, \*data = next(flag);

for (int i = 0; i < m->rows; ++i){

while(flag->data == 0){

if(data->data - i > 1 || (prev(flag) && prev(prev(flag)) && prev(prev(flag))->data == 0 && data->data == 0)) {

print\_row\_zeros(m->cols);

i++;

}

if(data->data == 0) {

puts("");

return;

}

puts("");

flag = next(data);

data = next(flag);

}

for (int j = 0; j < m->cols; ++j) {

if(flag->data == 0){

if(data->data == 0){

if(prev(prev(flag)) != 0)printf("0 ");

puts("");

return;

}

for (int k = j; k < m->cols; ++k) {

printf("0 ");

}

break;

}

if(flag->data > j + 1){

printf("0 ");

continue;

}

printf("%d ", data->data);

flag = next(data);

data = next(flag);

}

}

puts("");

}

void print\_inner(Matrix \*m){

Node\* el = m->start;

while(el != m->end){

printf("%d ", el->data);

el = el->next;

}

printf("%d\n", el->data);

}

bool skew\_symmetric(Matrix \*m){

return matrices\_equal(transpose(m), negative\_matrix(m));

}

Matrix\* transpose(Matrix \*m){

Matrix \*m2 = (Matrix \*)malloc(sizeof(Matrix));

m2->rows = m->cols;

m2->cols = m->rows;

m2->end = m2->start = NULL;

for (int i = 0; i < m2->cols; ++i) {

push\_back(m2, 0);

push\_back(m2, i+1);

}

push\_back(m2, 0);

push\_back(m2, 0);

Node \*cur\_index = next(next(m->start));

Node \*cur\_data = next(cur\_index);

int x, y = 1;

while(!(cur\_index->data == 0 && cur\_data->data == 0)){

if(cur\_index->data == 0){

y = cur\_data->data;

goto renew;

}

x = cur\_index->data;

insert\_xy(m2, y, x, y, cur\_data->data);

renew:

cur\_index = next(cur\_data);

cur\_data = next(cur\_index);

}

return m2;

}

Matrix\* transpose\_by\_side\_diagonal(Matrix \*m){

Matrix \*m2 = (Matrix \*)malloc(sizeof(Matrix));

m2->rows = m->cols;

m2->cols = m->rows;

m2->end = m2->start = NULL;

for (int i = 0; i < m2->cols; ++i) {

push\_back(m2, 0);

push\_back(m2, i+1);

}

push\_back(m2, 0);

push\_back(m2, 0);

Node \*cur\_index = next(next(m->start));

Node \*cur\_data = next(cur\_index);

int x, y = 1;

while(!(cur\_index->data == 0 && cur\_data->data == 0)){

if(cur\_index->data == 0){

y = cur\_data->data;

goto renew;

}

x = cur\_index->data;

int res\_x = m->rows - y + 1;

int res\_y = m->cols - x + 1;

insert\_xy(m2, res\_x, res\_y, res\_x, cur\_data->data);

renew:

cur\_index = next(cur\_data);

cur\_data = next(cur\_index);

}

return m2;

}

bool matrices\_equal(Matrix \*m1, Matrix \*m2){

if(m1->rows != m2->rows || m1->cols != m2->cols) return false;

Node \*flag1 = m1->start, \*data1 = next(flag1);

Node \*flag2 = m2->start, \*data2 = next(flag2);

while(data1 != m1->end && data2 != m2->end){

if(flag1->data != flag2->data || data1->data != data2->data) return false;

flag1 = next(data1);

data1 = next(flag1);

flag2 = next(data2);

data2 = next(flag2);

}

if(data1 == m1->end && data2 == m2->end)

return true;

return false;

}

Matrix\* negative\_matrix(Matrix \*m){

Matrix \*nm = (Matrix\*)malloc(sizeof(Matrix));

nm->cols = m->cols;

nm->rows = m->rows;

Node \*flag = m->start, \*data = next(flag);

while(data != m->end){

push\_back(nm, flag->data);

if(flag->data > 0)

push\_back(nm, -data->data);

else

push\_back(nm, data->data);

flag = next(data);

data = next(flag);

}

push\_back(nm, flag->data);

push\_back(nm, -data->data);

return nm;

}

sakost@sakost-pc  ~/projects/CLionProjects/lab/cmake-build-debug/cw7  cat ../../cw7/matrix.h

//

// Created by sakost on 22.04.2020.

//

#ifndef LAB\_MATRIX\_H

#define LAB\_MATRIX\_H

#include <stdbool.h>

typedef struct Node{

int data;

struct Node\* next, \*prev;

} Node;

typedef struct{

int cols, rows;

Node\* start, \*end;

} Matrix;

Matrix\* init();

void push\_back(Matrix\* m, int element);

void enter(Matrix\* m, char\* filename);

void print(Matrix \*m);

void pop\_back(Matrix \*m);

void print\_inner(Matrix \*m);

bool skew\_symmetric(Matrix \*m);

Matrix\* transpose(Matrix \*m);

Matrix\* transpose\_by\_side\_diagonal(Matrix \*m);

bool matrices\_equal(Matrix \*m1, Matrix \*m2);

Matrix\* negative\_matrix(Matrix \*m);

#endif //LAB\_MATRIX\_H

sakost@sakost-pc  ~/projects/CLionProjects/lab/cmake-build-debug/cw7 

# **Заключение**

Я смог написать программу на языке Си с помощью динамических структур, которая считывает разреженную матрицу из текстового файла и производит над ней некоторые операции по ее выводу или преобразованию с учётом возможности выбора действий для пользователя. А также научился работе с разреженными матрицами и комплексными операциями над ними.

# **Список используемых источников**

1. Функция fseek [Электронный ресурс]. URL: <http://www.c-cpp.ru/content/fseek> (Дата обращения: 23.04.2020).
2. Односвязный линейный список [Электронный ресурс]. URL: <https://prog-cpp.ru/data-ols/> (Дата обращения: 23.04.2020).
3. Функция fgetc [Электронный ресурс]. URL: <http://www.c-cpp.ru/content/fgetc-fgetchar> (Дата обращения: 23.04.2020).
4. Фунция feof [Электронный ресурс]. URL: <http://www.cpluplus.com/reference/cstdio/feof> (Дата обращения 23.04.2020)
5. Функция fscanf [Электронный ресурс]. URL: <http://www.cplusplus.com/reference/cstdio/fscanf/> (Дата обращения 23.04.2020)