**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский Авиационный Институт»**

**(Национальный Исследовательский Университет)**

**Институт №8 «Компьютерные науки и прикладная математика»**

# **Кафедра 806 «Вычислительная математика и программирование»**

**Курсовой проект 7**

**Автор работы:**

студент 1 курса, группа М8О-102Б-21

Яценко Александр Владимирович

**Руководитель проекта:**

Доцент кафедры 806 Никулин С.П.

**Дата сдачи:**

**Цель работы:**

Составить программу на языке Си с процедурами и/или функциями для обработки прямоугольных разреженных матриц с элементами вещественного типов, которая:

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

**Вариант преобразований:**

1. **Вариант физического представления**: отображение на динамические структуры.

**Вариант схемы размещения**: два вектора

**Вариант преобразования:** найти столбец, содержащий наибольшее количество ненулевых элементов, и напечатать его номер и произведение элементов этого столбца. Если таких будет несколько обработать предпоследний из них.

**Программа:**

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

typedef struct Node

{

double lyambda, znach;

struct Node\* next;

} Node;

typedef struct List

{

Node\* head;

} List;

List\* create\_list()

{

return (List\*)calloc(1, sizeof(List));

}

Node\* create\_node(double lya, double zn)

{

Node\* node = (Node\*)malloc(sizeof(Node));

node->lyambda = lya;

node->znach = zn;

node->next = NULL;

return node;

}

void list\_append(List\* list, Node\* new\_elem)

{

if (list->head == NULL)

{

list->head = new\_elem;

new\_elem->next = NULL;

return;

}

Node\* now = list->head;

while (now->next != NULL)

now = now->next;

now->next = new\_elem;

new\_elem->next = NULL;

}

Node \*get\_i (List \*list, size\_t index)

{

Node\* now = list->head;

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

now = now->next;

return now;

}

void list\_delete(List \*list) {

Node \*ptr = list->head, \*ptr\_prev;

while (ptr) {

ptr\_prev = ptr;

ptr = ptr->next;

free(ptr\_prev);

}

free(list);

}

void printMatrix(List \*list, int m, int n){

Node \*now = list->head;

int l = 0, kolvo = 0, count = 0,count1 = 0, new\_str = 0;

double i, j, zn, lya, b;

while(now->next != NULL){

zn = now->znach;

lya = now->lyambda;

b = lya / (double)n + 1;

i = (int)b;

j = lya - (i - 1) \* n;

count = j + n \* i - count1;

for(int k = 0; k < count; k++){

printf("0 \t");

new\_str++;

if(new\_str == n){

new\_str = 0;

printf("\n");

}

}

printf("%.4lf \t", zn);

new\_str++;

if(new\_str == n){

new\_str = 0;

printf("\n");

}

count1 += count + 1;

now = now->next;

}

zn = now->znach;

lya= now->lyambda;

b = lya / (double)n + 1;

i = (int)b;

j = lya - (i - 1) \* n;

count = j + n \* i - count1;

for(int k = 0; k < count; k++){

printf("0 \t");

new\_str++;

if(new\_str == n){

new\_str = 0;

printf("\n");

}

}

printf("%.4lf \t", zn);

count1 += count + 1;

for(int k = count1; k < m \* n; k++){

printf("0\t");

new\_str++;

if(new\_str == n){

new\_str = 0;

printf("\n");

}

}

printf("\n");

}

void PrintnewView(List \*list){

double lya, zn;

Node \*now = list->head;

while(now->next != NULL){

printf("%.4lf %.4lf \t\n", now->lyambda, now->znach);

now = now->next;

}

printf("%.4lf %.4lf \t\n", now->lyambda, now->znach);

}

void WorkMatrix(List \*list, double\* a, int n){

double max, lya, zn, i, j, b;

int cheker = 0,check = 0, k = 1;

Node \*now = list->head;

while(now->next != NULL){

if(check == 0){

check = 1;

max = abs(now->znach);

lya = now->lyambda;

b = lya / (double)n + 1;

i = (int)b;

a[0] = i;

}

if(abs(now->znach) == max){

cheker = 0;

lya = now->lyambda;

b = lya / (double)n + 1;

i = (int)b;

for(int p = 0; p < k; p++){

if(a[p] == i){

cheker = 1;

break;

}

}

if(cheker == 0){

k++;

a = (double\*)realloc(a, k \* sizeof(double));

a[k-1] = i;

}

}

else{

if(abs(now->znach) > max){

max = abs(now->znach);

for(int p = 0; p < k; p++)

a[p] = 0;

lya = now->lyambda;

b = lya / (double)n + 1;

i = (int)b;

a[0] = i;

k = 1;

}

}

now = now->next;

}

if(abs(now->znach) == max){

cheker = 0;

lya = now->lyambda;

b = lya / (double)n + 1;

i = (int)b;

for(int p = 0; p < k; p++){

if(a[p] == i){

cheker = 1;

break;

}

}

if(cheker == 0){

k++;

a = (double\*)realloc(a, k \* sizeof(double));

a[k-1] = i;

}

}

else{

if(abs(now->znach) > max){

max = abs(now->znach);

for(int p = 0; p < k; p++)

a[p] = 0;

lya = now->lyambda;

b = lya / (double)n + 1;

i = (int)b;

a[0] = i;

k = 1;

}

}

Node\* naw = list->head;

while(naw->next != NULL){

zn = naw->znach;

lya = naw->lyambda;

b = lya / (double)n + 1;

i = (int)b;

for(int p = 0; p < k; p++){

if(i == a[p]){

naw->znach = zn / max;

}

}

naw = naw->next;

}

zn = naw->znach;

lya = naw->lyambda;

b = lya / (double)n + 1;

i = (int)b;

for(int p = 0; p < k; p++){

if(i == a[p]){

naw->znach = zn / max;

}

}

printf("Номер столбца: %d\n", k);

printf("Произведение: %f.4\n", &max);

}

int menu() {

printf("%s\n", "1. Вывести оригинальную матрицу");

printf("%s\n", "2. Вывести матрицу во внутреннем представлении");

printf("%s\n", "3. Выволнить функцию");

printf("%s\n", "4. Выход");

int k;

scanf("%d", &k);

return k;

}

void without\_filename()

{

printf("Error.\nUsage: [INT] FILE.\n");

}

void file\_not\_exist(char\* file) //file not exist

{

printf("File \"%s\" not exist.\nUsage: [INT] FILE.\n",file);

}

int main(int argc, char\* argv[]) {

int m, n;

double \*a;

a = (double\*)malloc(sizeof(double));

double lyambda, znach;

List\* list = create\_list();

if(argc<1) {

without\_filename();

return 1;

}

FILE\* f\_in;

f\_in = 0;

f\_in = fopen(argv[1],"r");

if(!f\_in) {

file\_not\_exist(argv[1]);

return 2;

}

fscanf(f\_in,"%d %d", &m, &n);

for (int i = 0; i < m; i++)

{

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

{

double element;

fscanf(f\_in, "%lf", &element);

if (element != 0) {

lyambda = (i-1) \* n + j;

znach = element;

list\_append(list, create\_node(lyambda, znach));

}

}

}

int k = 0;

while (k != 4) {

k = menu();

switch (k)

{

case 1:

{

printMatrix(list, m, n);

break;

}

case 2:

{

PrintnewView(list);

break;

}

case 3:

{

WorkMatrix(list,a, n);

break;

}

case 4:

{

break;

}

default:

printf("%s\n", "Попробуйте снова");

}

}

list\_delete(list);

fclose(f\_in);

return 0;

}

**Работа программы:**

[root@fedora ~]# gcc 7kurs.c -o 7.out

[root@fedora ~]# ./7.out in.txt

1. Вывести оригинальную матрицу

2. Вывести матрицу во внутреннем представлении

3. Выволнить функцию

4. Выход

1

7.0000 0 0 0

8.0000 0 0 4.0000

8.0000 0 0 0

1. Вывести оригинальную матрицу

2. Вывести матрицу во внутреннем представлении

3. Выволнить функцию

4. Выход

2

-4.0000 7.0000

0.0000 8.0000

3.0000 4.0000

4.0000 8.0000

1. Вывести оригинальную матрицу

2. Вывести матрицу во внутреннем представлении

3. Выволнить функцию

4. Выход

3

Номер столбца: 1

Произведение: 448.0000

1. Вывести оригинальную матрицу

2. Вывести матрицу во внутреннем представлении

3. Выволнить функцию

4. Выход

4

[root@fedora ~]# ./7.out in2.txt

1. Вывести оригинальную матрицу

2. Вывести матрицу во внутреннем представлении

3. Выволнить функцию

4. Выход

1

0 2.0000 4.0000 1.0000

0 0 2.0000 3.0000

1.0000 0 0 0

1. Вывести оригинальную матрицу

2. Вывести матрицу во внутреннем представлении

3. Выволнить функцию

4. Выход

2

-3.0000 2.0000

-2.0000 4.0000

-1.0000 1.0000

2.0000 2.0000

3.0000 3.0000

4.0000 1.0000

1. Вывести оригинальную матрицу

2. Вывести матрицу во внутреннем представлении

3. Выволнить функцию

4. Выход

3

Номер столбца: 3

Произведение: 8.0000

1. Вывести оригинальную матрицу

2. Вывести матрицу во внутреннем представлении

3. Выволнить функцию

4. Выход

4

[root@fedora ~]# ./7.out in3.txt

1. Вывести оригинальную матрицу

2. Вывести матрицу во внутреннем представлении

3. Выволнить функцию

4. Выход

1

1.0000 0 0 -3.0000

0 0 4.0000 5.0000

6.0000 0 7.0000 9.0000

0 2.0000 0 0

1. Вывести оригинальную матрицу

2. Вывести матрицу во внутреннем представлении

3. Выволнить функцию

4. Выход

2

-4.0000 1.0000

-1.0000 -3.0000

2.0000 4.0000

3.0000 5.0000

4.0000 6.0000

6.0000 7.0000

7.0000 9.0000

9.0000 2.0000

1. Вывести оригинальную матрицу

2. Вывести матрицу во внутреннем представлении

3. Выволнить функцию

4. Выход

3

Номер столбца: 4

Произведение: -135.000000

1. Вывести оригинальную матрицу

2. Вывести матрицу во внутреннем представлении

3. Выволнить функцию

4. Выход

4

[root@fedora ~]#

**Заключение:**