Отчёт по заданию №7 курсового проекта

Черыгова Елизавета
Группа 8О-104Б

Оглавление

Цель работы..... 2
Алгоритм решения задачи..... 4
Код программы 5
Заключение..... 12

Цель работы

Составить программу на языке Си с функциями для обработки прямоугольных разреженных матриц с элементами целого типа, которая:

1. Вводит матрицы различного размера с одновременным размещением ненулевых элементов в разреженной матрице в соответствии с заданной схемой;
2. Печатает введенные матрицы во внутреннем представлении и в обычном виде;
3. Выполняет необходимые преобразования разреженных матриц (или вычисления над ними) путем обращения к соответствующим функциям;
4. Печатает результат преобразования во внутреннем представлении и в обычном виде.

Входные данные

На первой строке входного файла — числа M и N — размеры матрицы, затем следуют MxN элементов матрицы. Далее возможно (в зависимости от вариантов) указание параметров (чисел, элементов вектора-столбца или вектора-строки).

Выходные данные

Матрица во внутреннем представлении, матрица в обычном виде, матрица во внутреннем представлении после преобразования, матрица в обычном виде после преобразования, результат вычисления функции или предиката (присутствует в некоторых вариантах). Варианты внутреннего представления матрицы

Все матрицы хранятся по строкам, в порядке возрастания индексов ненулевых элементов.

Задание:

Найти элемент матрицы, ближайший к заданному значению и разделить на него все элементы строки и столбца, в которых он расположен.

Один вектор.

Отображение на массив.

Алгоритм решения задачи

Задание выполняется в три этапа:

- ищем ближайший к введённому элемент;
- ищем все такие элементы;
- выполняем задание для каждого элемента.

Код программы

```
// Номер по списку 14: 3. Найти элемент матрицы, ближайший к заданному значению и
// разделить на него все элементы строки и столбца, в которых он расположен.
// Один вектор.
// Отображеине на массив.

#include <stdio.h>
#include <string.h>
#include "function7.h"

int main(){
	mat a;
	double p;
	char com[10];
	init(&a);

	help();

	while(strcmp(com, "exit")!=0){
		printf("=>");
		scanf("%s", com);

		if(strcmp(com, "read")==0){
			read(&a);
		}
		else if(strcmp(com, "print")==0){
			puts("=====+print+======");
			print(a);
			puts("==================");
		}
		else if(strcmp(com, "inprint")==0){
			puts("====+inprint+=====");
			inprint(a);
			puts("==================");
		}
		else if(strcmp(com, "sol")==0){
			solution(&a);
		}
		else if(strcmp(com, "exit")==0){
			;
		}
		else puts("Unknown command");
	}
	return 0;
}
```

```
#ifndef FUNCTION7_H
#define FUNCTION7_H
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <limits.h>

#define EPS 1e-12
#define MEM_EPS 10
#define V_BASE_SIZE 10

/*******Дескриптор матрицы*****/
typedef struct{
	int m;                                                  // Матрица
MxN, M - количесво строк, N - столбцов.
	int n;
	float* vec;                                             // Вектор,
хранящий информацию о ненулевых элементах.
	int mem_inf;                                        // Количество
выделенной памяти.
}mat;
/*****************************/


/*******************Объявление функций**************/
void help();

float absl(float);                                          // Абсолютоное
значение для float/

void init(mat*);                                            // Инициализирует
матрицу 0x0.
void resize(mat*, int);                                     // Перевыделить
вектору память.

void read(mat*);                                            // Считывает
матрицу.
void print(mat);                                            // Печатает матрицу.
void inprint(mat);                                          // Печатает матрицу
во внутреннем представлении.

void solution(mat*);                                    // Выполняет поставленную
задачу.
/************************************************/
```

```
/******* Функции ********************/

void help(){
	puts("===============help=====================");
	puts("read <m> <n> <m*n matrix element>...read matrix.");
	puts("print.............................print matrix normal way.");
	puts("inprint...........................print matrix in an internal representation");
	puts("sol <parameter>.....................make a solution of a problem.");
	puts("exit..............................end program.");
	puts("=========================================");
}

float absl(float a){
	return (a>0 ? a : -a);
}

void init(mat* A){
	A->m=A->n=0;
	A->vec=(float*)malloc(V_BASE_SIZE*sizeof(float));
	A->mem_inf=V_BASE_SIZE;
	A->vec[0]=0;
}

void resize(mat* A, int mem){
	A->vec=realloc(A->vec, mem*sizeof(float));
	A->mem_inf=mem;
}

void read(mat* A){ //++
	int k,i,j, flag;                                        // k - индекс вектора, i,j - строка/столбец элемента.
	float val;                                              // значение элемента
	scanf("%d %d", &A->m, &A->n);
	k=1;
	for(i=0;i<A->m;i++){
		flag=1;
		for(j=0;j<A->n;j++){
			scanf("%f", &val);
			if(absl(val)>EPS){
				if((A->mem_inf - k) < MEM_EPS) resize(A, 2*A->mem_inf);
				if(flag){                                   // Если в строке встретился ненулевой эдемент - ставим номер строки в вектор.
					A->vec[k++]=i+1;
					flag=0;
```

```
                        }
                        A->vec[k++]=j+1;                    // Ставим номер
столбца в вектор.
                        A->vec[k++]=val;                    // Ставим значение в
вектор.
                    }
                }
                if(!flag){
                    A->vec[k++]=0;                          // Завершение
строки.
                }
        }
        A->vec[k++]=0;                                      // Конец
вектора.
        resize(A, k+1);
}

void inprint(mat A){  //++
        int i;
        for(i=0;i<A.mem_inf-1;i++){
                printf("%-2d. %.2f\n", i, A.vec[i]);
        }
        return;
}

void print(mat A){      //++
        int i, j, k=1, flag;
        for(i=0;i<A.m;i++){
                flag=0;                                     // Флаг
показывает, есть в i-й строке элементы, или она нулевая.
                if(absl(A.vec[k]-i-1)<EPS){                 // Проверяем, есть ли
эта строка в векторе.
                    k++;
                    flag=1;                                 // Если
есть, печатаем элементы этой строки.
                }
                for(j=0;j <A.n;j++){
                    if(absl(A.vec[k]-j-1)<EPS && flag){     // Если номер столбца совпадает со
столбцом эл-та и в этой строке есть ненулевы элементы, печатаем.
                        k++;
                        printf("%.2f  ",A.vec[k]);
                        k++;
                    }
                    else{                                   // Если
эл-та с такими индексами нет, печатаем 0.
                        printf("0.00  ");
```

```
                }
            }
            if(flag) k++;
            printf("\n");
        }
}

void solution(mat* A){

    // Решение ищется в три этапа: В первом ищем ближайший к введенному элемент, во втором ищем все такие элементы,
    // в третьем выполняем задание для каждого элемента.

    int m=0, k, t;
    float i, j, val, min=INT_MAX, i_min[10], j_min[10], ex_val;
    // i, j - строка/столбец. min - минимальная разность между значением и val
    // val - заданное значение, i_min[]/j_min[]- список строк/столбцов с минимальным эл-том. ex_val-значение мин. эл-та.

    scanf("%f", &val);

    k=1;
    while(absl(A->vec[k])>EPS){                    // Идем по вектору, пока не видим 0 - конец вектора.
        i=A->vec[k++];
        while(absl(A->vec[k])>EPS){                // Идем по вектору, пока не видим 0 - конец строки.
            j=A->vec[k++];
            if((absl(A->vec[k]-val))<min){         // Если разность между значением и val меньше текущей, значит значение эл-та ближе к val.
                min=absl(A->vec[k]-val);           // Разность равна |значение-val|
                i_min[0]=i;
                j_min[0]=j;
                ex_val=A->vec[k];
            }
            k++;                                   // Не зависимо от того, подходит ли эл-т, сдвигаем указатель. В ячейке будет либо 0 -конец строки, либо номер столбца.
        }
        k++;                                       // Строка закончилась, проскакиваем 0. Индекс будет указывать на след. строку или на 0 -т.е. конец вектора.
    }

    // В этом цикле ищем все эл-ты, для которых нужно выполнить задание.
```

```
	m=0;
	k=1;
	while(absl(A->vec[k])>EPS){
		i=A->vec[k++];                                          // См. пред. цикл.
		while(absl(A->vec[k])>EPS){
			j=A->vec[k++];                                  // См. пред. цикл.
			if(absl(A->vec[k]-ex_val)<EPS){                 // Значение=ex_val => для этого эл-та нужно выполнить задание.
				i_min[m]=i;                             // Запоминаем его строку и столбец.
				j_min[m]=j;
				m++;
			}
			k++;
		}
		k++;
	}
	m--;


	for(t=0;t<m+1;t++){
	// Проходим этот цикл столько раз, сколько значений попадает под условие.
		k=1;
		while(absl(A->vec[k])>EPS){
			i=A->vec[k++];
			while(absl(A->vec[k])>EPS){
				j=A->vec[k++];
				if(absl(j-j_min[t])<EPS || absl(i-i_min[t])<EPS){       // Если строка или столбец совпадают с с/ст эл-та, для которого нужно выполнить задание, то
					A->vec[k]= A->vec[k]/ex_val;
	// Делим этот элемент на значение эл-та, для которого нужно выполнить задание..
				}
				k++;
			}
			k++;
		}
	}
}




#endif
```

Вывод программы

```
==============help=====================
read <m> <n> <m*n matrix element>...read matrix.
print..............................print matrix normal way.
inprint............................print matrix in an internal
representation
sol <parameter>.....................make a solution of a problem.
exit...............................end program.
========================================
=>read 2 2 4 4 4 2
=>print
=====+print+======
4.00  4.00
4.00  2.00
==================
=>sol 2
=>print
=====+print+======
4.00  2.00
2.00  1.00
==================
=>read 3 3 1 1 1 1 3 1 1 1 1
=>print
=====+print+======
1.00  1.00  1.00
1.00  3.00  1.00
1.00  1.00  1.00
==================
=>sol 3
=>print
=====+print+======
1.00  0.33  1.00
0.33  1.00  0.33
1.00  0.33  1.00
==================
=>read 4 4 0 0 5 4 0 5 6 0 1 1 9 8 0 4 5 7
=>print
=====+print+======
0.00  0.00  5.00  4.00
0.00  5.00  6.00  0.00
1.00  1.00  9.00  8.00
0.00  4.00  5.00  7.00
==================
=>sol 5
=>print
=====+print+======
0.00  0.00  0.20  0.80
0.00  1.00  0.05  0.00
1.00  0.20  0.36  8.00
0.00  0.16  0.20  1.40
==================
```

Заключение

Благодаря, этому заданию, я смогла поближе познакомиться и разобраться с представлением в Си с функциями для обработки прямоугольных разреженных матриц с элементами целого типа.