Задание 7: Разреженные матрицы

Цель работы

Составить программу на языке Си для обработки прямоугольных разряженных матриц, которая:

1. вводит матрицы различного размера, представленные во входном текстовом файле в обычном формате (по строкам);
2. печатает введенные матрицы во внутреннем представлении, согласно заданной схеме размещения и в обычном (естественном) виде;
3. выполняет необходимые преобразования (вычисления) разреженных матриц, путем обращения к соответствующим процедурам и функциям;
4. печатает результат преобразования (вычисленния) согласно заданной схеме оазмещения и в обычном виде

Задание

Вариант схемы размещения матрицы №1: Цепочка ненулевых элементов в векторе A со строчным индексированием. Массив M, содержащий индексы начал строк матрицы в массиве A (индексы в массиве M равны 0, если соответстующая строка содержит только нули).

Вариант преобразования №6: Вычислить сумму двух разреженных матриц. Проверить, является ли полученная матрица симметричной.

Алгоритм работы программы

Ввод и вывод матрицы:

Сначала у пользователя запрашивается размер матрицы. Затем запрашивается ввод всей матрицы в стандартном виде. Матрица считывается в двойном цикле. Если введённое число не ноль, то в массив A записываются два числа: номер столбца матрицы, в котором находится этот элемент, (соответствует значению счётчика во вложенном цикле) и значение элемента. Ячейка массива M, соответствующая текущей строке (то есть индекс которой равен значению счётчика внешнего цикла), перед входом во вложенный цикл приравнивается к текущему положению в массиве A. Если после прохождения вложенного цикла положение в массиве A не изменилось, то строка нулевая и соответствующую ячейку массива M необходимо обнулить. Если матрица нулевая, то единственным значением в массиве A будет 0.

При выводе матрицы совершается поиск её элеметов по индексам с помощью массивов A и M. Положение нужной строчки в массиве A находится из массива M. Номер первого ненулевого столбца данной строчки в массиве A находится в ячейке, находящейся перед той, на которую указывает соответствующая ячейка массива M. Каждый следующий номер столбца находится через одну ячейку от текущего. Если на

месте номера столбца стоит 0, это означает, что строка закончилась. Если элемент с необходимыми индексами найден в массиве A, то он выводится на экран, иначе выводится 0.

Нахождение суммы и проверка симметричности:

При нахождении суммы матрицы суммируются построчно. Из массивов M слогаемых матриц находим положения нужных строк в массивах A. Проходим оба массива A по номерам столбцов элементов матриц, если индексы совпадают, это означает, что элементы в обоих матрицах стоят на одном месте, тогда, если их сумма не ноль, то она записывается в массив для матрицы суммы после соответствующего номера столбца. Если номера столбцов не совпадают, то в одном из массивов мы зашли дальше, чем в другом, тогда мы продолжаем обрабатывать оставшийся массив, пока не догоним другой. При этом значения элементов и номера их столбцов просто переносятся в массив суммы. Если в одном из массивов мы дошли до конца строки, то мы обрабатываем только оставшийся массив до конца строки. Когда в обоих массивах был достигнут конец строки, осуществляется переход к следующей строке.

Симметричность означает, что для каждого ненулевого элемента a_{ij} можно найти равный ему ненулевой элемент a_{ji}. Таким образом совершается обход массива A, во время которого для каждого элемента совершается поиск симметричного с помощью вспомогательной функции. В этой функции для элемета a_{ij} мы проходим часть массива со строкой j и ищем в ней i-ый элемент, если строка кончилась, а элемент так и не был найден, то матрица не симметрична.

Функции и структуры

Матрица хранится в структуре matrix, содержащей целые числа m – количество строк, b – количество столбцов, a длина массива A; целочисленные указатели A и M для хранения массивов согласно схеме размещения.

void printmatrix(matrix m)	Печать матирцы на экран
void readmatrix(matrix *M)	Чтение матрицы
matrix sum(matrix m1, matrix m2)	Вычисление суммы матриц
int SymVal(int v, int i, int j, matrix *m)	Нахождение симметричного элемента
int sym(matrix *m)	Проверка симметричности матрицы

Код программы

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

typedef struct matrix matrix;


struct matrix{
	int *M, *A, m, a, b;
```

```
};

void printmatrix(matrix m){
	int l=0;
	for(int i=0; i<m.m; i++){
		l=m.M[i]-1;
		for(int j=0; j<m.b; j++){
			if((l>=0)){
				if(m.A[l]==j+1){
				printf("%d ", m.A[l+1]);
				l+=2;
				}
				else printf("0 ");
			}
			else printf("0 ");
		}
		printf("\n");
	}
}

void readmatrix(matrix *M){
	int a, k=1, c;
	(*M).a=0;
	if((*M).m<0){
		printf("Enter number of lines: ");
		scanf("%d", &(*M).m);
		printf("Enter number of columns: ");
		scanf("%d",&(*M).b);
	}
	(*M).M=(int *)malloc(sizeof(int)*(*M).m);
	(*M).M[0]=1;
	(*M).A=(int *)malloc(sizeof(int)*k);
	printf("Enter matrix:\n");
	for(int i=0; i<(*M).m; i++){
		(*M).M[i]=(*M).a+1;
		for(int l=0; l<(*M).b; l++){
			scanf("%d",&c);
			if(c!=0){
				if(k-(*M).a<=2){
					(*M).A=(int *)realloc((*M).A, (k*2)*sizeof(int));
					k*=2;
				}
				(*M).A[(*M).a]=l+1;
				(*M).A[(*M).a+1]=c;
				(*M).a+=2;
			}
			if(l==(*M).b-1){
				if((*M).M[i]==(*M).a+1) (*M).M[i]=0;
				if(((*M).A[(*M).a-1]!=0)&&((*M).a>0)) {
					(*M).A[(*M).a]=0;
					(*M).a++;
				}
				if((i==(*M).m-1)&&((*M).a==0)) (*M).A[0]=0;
			}
		}
		if((*M).a==0){
			(*M).a=1;
			(*M).A[0]=0;
		}
	}
	(*M).A=(int *)realloc((*M).A, sizeof(int)*(*M).a);
}
```

```
matrix sum(matrix m1, matrix m2){
	matrix m;
	m.b=m2.b;
	m.m=m2.m;
	int i1=0, i2=0, k=1;
	m.M=(int *)malloc(sizeof(int)*m.m);
	m.A=(int *)malloc(sizeof(int)*k);
	m.a=0;
	for(int l=0; l<m.m; l++){
		i1=m1.M[l]-1;
		i2=m2.M[l]-1;
		m.M[l]=m.a+1;
		while((i1!=-1)||(i2!=-1)){
			if(k-m.a<=2){
				k*=2;
				m.A=(int *)realloc(m.A, sizeof(int)*k);
			}
			if (i1>=0) if (m1.A[i1]==0) i1=-1;
			if (i2>=0) if (m2.A[i2]==0) i2=-1;
			if((i1!=-1)){
				if(i2!=-1){
					if(m1.A[i1]==m2.A[i2]){
						if(m1.A[i1+1]+m2.A[i2+1]!=0){
							m.A[m.a]=m1.A[i1];

	m.A[m.a+1]=m1.A[i1+1]+m2.A[i2+1];
							m.a+=2;
							i1+=2;
							i2+=2;
						}
						else{
							i1+=2;
							i2+=2;
						}
					}
					else if(m1.A[i1]<m2.A[i2]){
						m.A[m.a]=m1.A[i1];
						m.A[m.a+1]=m1.A[i1+1];
						m.a+=2;
						i1+=2;
					}
					else if(m2.A[i2]<m1.A[i1]){
						m.A[m.a]=m2.A[i2];
						m.A[m.a+1]=m2.A[i2+1];
						m.a+=2;
						i2+=2;
					}
				}
				else{
					m.A[m.a]=m1.A[i1];
					m.A[m.a+1]=m1.A[i1+1];
					m.a+=2;
					i1+=2;
				}
			}
			else if(i2!=-1){
				m.A[m.a]=m2.A[i2];
				m.A[m.a+1]=m2.A[i2+1];
				m.a+=2;
				i2+=2;
			}
```

```
                                        else{
                                                if(m.a>0){
                                                        if(m.A[m.a-1]!=0){
                                                                m.A[m.a]=0;
                                                                m.a++;
                                                        }
                                                }

                                        }
                                }
                                if(m.M[l]==m.a+1) m.M[l]=0;
                        }
                        if(m.a==0){
                                m.a=1;
                                m.A[0]=0;
                        }
                        m.A=(int*)realloc(m.A, sizeof(int)*m.a);
                        return m;
}

int SymVal(int v, int i, int j, matrix *m){
                        int a=1,l;
                        l=m->M[j-1]-1;
                        if(l==-1) a=0;
                        else{
                                while(m->A[l]!=0){
                                        if(m->A[l]==i){
                                                if(m->A[l+1]!=v){
                                                        a=0;
                                                }
                                                break;
                                        }
                                        else if(m->A[l]>i){
                                                a=0;
                                                break;
                                        }
                                        l+=2;
                                }
                        }
                        return a;
}

int sym(matrix *m){
                        int s=1, l;
                        for(int i=0; i<m->m; i++){
                                l=m->M[i]-1;
                                if(l!=-1){
                                        while(m->A[l]!=0){
                                                s=SymVal(m->A[l+1], i+1, m->A[l], m);
                                                l+=2;
                                                if(s==0) break;
                                        }
                                        if(s==0) break;
                                }
                        }
                        return s;
}



int main(){
                        //Âûâîä èíôîðìàöèè î ìàòðèöàõ
```

```
        int *a, *b;
        matrix m1, m2, m3;
        m1.m=-1;
        readmatrix(&m1);
        printf("\nArray A:\n");
        for(int i=0; i<m1.a; i++) printf("%d ", m1.A[i]);
        printf("\nArray M:\n");
        for(int i=0; i<m1.m; i++) printf("%d ", m1.M[i]);
        printf("\n");
        printf("\n");
        printf("\n\n");
        m2.m=m1.m;
        m2.b=m1.b;
        readmatrix(&m2);
        printf("\nArray A:\n");
        for(int i=0; i<m2.a; i++) printf("%d ", m2.A[i]);
        printf("\nArray M:\n");
        for(int i=0; i<m2.m; i++) printf("%d ", m2.M[i]);
        //Ðàñïå÷àòêà â ñòàíäàðòíîì âèäå
        printf("\n\nFirst matrix:\n");
        printmatrix(m1);
        printf("\n\nSecond matrix:\n");
        printmatrix(m2);
        //Ñóììà ìàòðèö è âûâîä èíôîðìàöèè î íåé, ðàñïå÷àòêà
        printf("\n\nSum:\n\n");
        m3=sum(m1, m2);
        printmatrix(m3);
        printf("\n\nArray A:\n");
        for(int i=0; i<m3.a; i++) printf("%d ", m3.A[i]);
        printf("\n\nArray M:\n");
        for(int i=0; i<m3.m; i++) printf("%d ", m3.M[i]);
        //Ïðîâåðêà ñèììåòðè÷íîñòè
        printf("\n\n");
        if(sym(&m3)) printf("Sum is symmetrical\n");
        else printf("Sum isn't symmetrical\n");
        free(m1.A);
        free(m1.M);
        free(m2.A);
        free(m2.M);
        free(m3.A);
        free(m3.M);
}
```

Вывод программы

```
Enter number of lines: 10

Enter number of columns: 10

Enter matrix:
1 2 0 0 0 3 4 0 0 0
3 0 0 4 0 0 5 0 0 6
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
1 2 0 0 0 0 0 0 0 0
4 0 5 0 0 0 4 0 5 6
0 0 0 0 0 0 5 0 0 0
4 0 0 0 0 5 0 0 0 0
4 5 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 1
```

Array A:
1 1 2 2 6 3 7 4 0 1 3 4 4 7 5 10 6 0 1 1 2 2 0 1 4 3 5 7 4 9 5 10 6 0 7 5 0 1 4 6 5 0 1 4
2 5 0 1 1 10 1 0
Array M:
1 10 0 0 19 24 35 38 43 48

Enter matrix:
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0

Array A:
0
Array M:
0 0 0 0 0 0 0 0 0 0

First matrix:
1 2 0 0 0 3 4 0 0 0
3 0 0 4 0 0 5 0 0 6
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
1 2 0 0 0 0 0 0 0 0
4 0 5 0 0 0 4 0 5 6
0 0 0 0 0 0 5 0 0 0
4 0 0 0 0 5 0 0 0 0
4 5 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 1

Second matrix:
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0

Sum:

1 2 0 0 0 3 4 0 0 0
3 0 0 4 0 0 5 0 0 6

0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
1 2 0 0 0 0 0 0 0 0
4 0 5 0 0 0 4 0 5 6
0 0 0 0 0 0 5 0 0 0
4 0 0 0 0 5 0 0 0 0
4 5 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 1

Array A:
1 1 2 2 6 3 7 4 0 1 3 4 4 7 5 10 6 0 1 1 2 2 0 1 4 3 5 7 4 9 5 10 6 0 7 5 0 1 4 6 5 0 1 4 2 5 0 1 1 10 1 0

Array M:
1 10 0 0 19 24 35 38 43 48

Sum isn't symmetrical

Enter number of lines: 10

Enter number of columns: 10
Enter matrix:
1 2 0 0 0 3 4 0 0 0
3 0 0 4 0 0 5 0 0 6
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
1 2 0 0 0 0 0 0 0 0
4 0 5 0 0 0 4 0 5 6
0 0 0 0 0 0 5 0 0 0
4 0 0 0 0 5 0 0 0 0
4 5 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 1

Array A:
1 1 2 2 6 3 7 4 0 1 3 4 4 7 5 10 6 0 1 1 2 2 0 1 4 3 5 7 4 9 5 10 6 0 7 5 0 1 4 6 5 0 1 4 2 5 0 1 1 10 1 0
Array M:
1 10 0 0 19 24 35 38 43 48

Enter matrix:
0 1 0 0 1 1 0 4 4 1
0 0 0 0 2 0 0 0 5 0
0 0 0 0 0 5 0 0 0 0
0 4 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 5 0 0
4 5 0 0 0 4 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 5 0 0 0 0
0 6 0 0 0 6 0 0 0 0

Array A:
2 1 5 1 6 1 8 4 9 4 10 1 0 5 2 9 5 0 6 5 0 2 4 0 8 5 0 1 4 2 5 6 4 0 6 5 0 2 6 6 6 0
Array M:
1 14 19 22 0 25 28 0 35 38

First matrix:
1 2 0 0 0 3 4 0 0 0
3 0 0 4 0 0 5 0 0 6
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
1 2 0 0 0 0 0 0 0 0
4 0 5 0 0 0 4 0 5 6
0 0 0 0 0 0 5 0 0 0
4 0 0 0 0 5 0 0 0 0
4 5 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 1

Second matrix:
0 1 0 0 1 1 0 4 4 1
0 0 0 0 2 0 0 0 5 0
0 0 0 0 0 5 0 0 0 0
0 4 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 5 0 0
4 5 0 0 0 4 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 5 0 0 0 0
0 6 0 0 0 6 0 0 0 0

Sum:

1 3 0 0 1 4 4 4 4 1
3 0 0 4 2 0 5 0 5 6
0 0 0 0 0 5 0 0 0 0
0 4 0 0 0 0 0 0 0 0
1 2 0 0 0 0 0 0 0 0
4 0 5 0 0 0 4 5 5 6
4 5 0 0 0 4 5 0 0 0
4 0 0 0 0 5 0 0 0 0
4 5 0 0 0 5 0 0 0 0
1 6 0 0 0 6 0 0 0 1

Array A:
1 1 2 3 5 1 6 4 7 4 8 4 9 4 10 1 0 1 3 4 4 5 2 7 5 9 5 10 6 0 6 5 0 2 4 0 1 1 2 2 0 1 4 3
5 7 4 8 5 9 5 10 6 0 1 4 2 5 6 4 7 5 0 1 4 6 5 0 1 4 2 5 6 5 0 1 1 2 6 6 6 10 1 0

Array M:
1 18 31 34 37 42 55 64 69 76

Sum is symmetrical

Заключение

В ходе данной работы были реализованы считывание и хранение разреженных матриц, вычисление суммы двух матриц и определение симмметричности разреженной матрицы.