Московский Авиационный Институт

(Национальный Исследовательский Университет)

Кафедра вычислительной математики и программирования

Курсовая работа

По курсу «Языки и методы программирования»

II семестр

Задание №7 «Разреженные матрицы»

Выполнил студент

1-го курса, 105-ой группы

Махмудов О. С.

_____

(подпись)

Научный руководитель

Доцент кафедры 806

Сластушенский Ю. В.

_____

(подпись)

Работа защищена

«__»_____ 2019

Оценка_____

ОГЛАВЛЕНИЕ

ЦЕЛЬ

Составить программу на языке Си с процедурами и/или функциями для обработки *прямоугольных* разреженных матриц с элементами вещественного типа, 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 (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 и в обычном виде.

В процедурах и функциях предусмотреть проверки и печать сообщений в случаях ошибок в задании параметров. Для отладки использовать матрицы, содержащие 5-10% ненулевых элементов с максимальным числом элементов 100.

Вариант схемы размещения матрицы определяется по формуле $((N + 3) \% 4) + 1$, где N — номер студента по списку в группе. Вариант преобразования определяется по формуле $((N - 1) \% 11) + 1$. Вариант физического представления (1 — отображение на массив, 2 — отображение на динамические структуры) определяются по формуле $([1.5 * ((3 + M) \% 9)] + N) \% 2 + 1$, где M— номер группы. В случае использования динамических структур индексы заменяются соответствующими ссылками.

Номер по списку N=11:

Вариант схемы размещения матрицы:

3. Три вектора:

Вариант преобразования №11:

Транспонировать разреженную матрицу относительно побочной диагонали. Выяснить, является ли полученная матрица кососимметрической.

Вариант физического представления №2:

Отображение на динамические структуры

ПРОГРАММА

Функции

void print(node *head)	Работает в выводе матрицы в векторном виде
void printt(node *head)	Работает в выоде матрицы в векторном виде
int enter()	Считывает матрицу из файла и преобразует ее
int output()	Выводит матрицу в обычном и векторном виде
int transform()	Выполняет транспонирование матрицы
void menu()	Выводит меню

Код программы

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

void error1() { printf("Matrix is not square\n"); };
void error2() { printf("Can't open file\n"); };
void error3() { printf("No matrix anymore\n\n"); };

typedef struct node {
	struct node *next;
	double value;
}node;

struct node* init(double a) {
	node *lst = (node*)malloc(sizeof(node));
	lst->value = a;
	lst->next = NULL;
	return(lst);
};

struct node * push(node *lst, double num) {
	node *tmp = (node*)malloc(sizeof(node));
	lst->next = tmp;
	tmp->value = num;
	tmp->next = NULL;
	return(tmp);
};

void print(node *head) {
	if (head == NULL)printf("empty");
	else {
		while (head) {
			printf("%-5.1lf ", head->value);
			head = head->next;
		}
	}
	printf("\n");
}
void printt(node *head) {
	if (head == NULL)printf("empty");
	else {
		while (head) {
			printf("%-5.0lf ", head->value);
			head = head->next;
		}
	}
	printf("\n");
}

typedef struct matrix {
	node **cip;
	node *pi;
	node *ye;
```

```
}matrix;

matrix m, mm;
int stlb = 0, str = 0, *b = NULL, res;
int *bb = NULL, count2;
FILE* file;

int enter() {
	double elem;
	int column = 1, i = 0, check = 1, count = 0;
	stlb = 0, str = 0;
	char s, s2;
	m.cip = (node**)malloc(i * sizeof(node*));
	node *p1, *p2;
	if (feof(file) != 0) {
		error3();
		return 1;
	}
	while (fscanf(file, "%lf", &elem) != EOF) {
		if (elem != 0) {
			if (count == 0) {
				m.ye = init(elem);
				m.pi = init(column);
				p1 = m.ye;
				p2 = m.pi;
			}
			if (check) {
				m.cip = (node**)realloc(m.cip, (i + 1) * sizeof(node*));
				b = (int*)realloc(b, (i + 1) * sizeof(int));
				*(m.cip + i) = p1;
				*(b + i) = count;
				i++;
			}
			if (count != 0) {
				p1 = push(p1, elem);
				p2 = push(p2, column);
			}
			count++;
			check = 0;
		}
		column++;
		fscanf(file, "%c", &s);
		if (s == '\n') {
			stlb = column - 1; column = 1; check = 1; str++;
			if (i < str) {
				m.cip = (node**)realloc(m.cip, (i + 1) * sizeof(node*));
				b = (int*)realloc(b, (i + 1) * sizeof(int));
				*(m.cip + i) = NULL;
				*(b + i) = -1;
				i++;
			}
			fscanf(file, "%c", &s2);
			if (s2 == '\n')break;
		}
	}
	return 0;
}
```

```
int output() {
	int i = 0, n = 1, c = -1;
	int *bbb;
	node *p1, *p2;
	p1 = m.ye; p2 = m.pi; bbb = b;
	printf("Matrix in the usual form :");
	for (int k = 0; k < str; k++) {
		n = 1;
		printf("\n"); c++;
		while (n <= stlb) {
			if (p2 && n == p2->value && *(bbb + c) != -1) {
				printf("%5.1lf ", p1->value);
				p1 = p1->next;
				p2 = p2->next;
			}
			else {
				printf("%5.1lf ", 0.0);
			}
			n++;
		}
	}
	for (int k = n; k < stlb + 1; k++)printf("%5.1lf ", 0.0);
	printf("\n");
	printf("\n");
	printf("Matrix in internal view : \n");
	if (num == 2) {
		if (m.ye == NULL)printf("All matrix elements are zero\n");
		printf("CIP : ");
		int k = 0;
		while (printf("%d  ", *(b + k)) > 0 && k < str - 1)k++;
		printf("\n");
		printf("PI  : "); printt(m.pi);
		printf("YE  : "); print(m.ye);
	}
	return 0;
}

int transform() {
	int i = str, j = stlb;
	if (i != j) { error1(); return 1; }
	if (g == 0) {
		mm.pi = init(ed);
		pp1 = mm.ye;
		pp2 = mm.pi;
		mm.cip = (node**)realloc(mm.cip, (ii + 1) * sizeof(node*));
		*(mm.cip + ii) = pp1;
		bb = (int*)realloc(bb, (ii + 1) * sizeof(int));
		*(bb + ii) = count2;
		ii++; g++;
	}
	int i = 0, n = 1, c = -1;
	int *bbb;
	node *p1, *p2;
	if (num == 2) { p1 = m.ye; p2 = m.pi; bbb = b; }
	else { p1 = mm.ye; p2 = mm.pi; bbb = bb; }
	p2 = NULL;
```

```
	for (int k = 0; k < j; k++) {
		n = 1;
		c++;
		while (n < stlb)){
		for (int l = i; l > 0; l--) {
			if (p2 && n == p2->value && *(bbb + c) != -1) {
				pp2 = 0;
				while (pp2->ppp2 != p2) pp2 = pp2->ppp2;
				swap(p1, p2); p1 = p1->next; p2->value = pp2;
			}
		}
		if (num == 2) { p1 = m.ye; p2 = m.pi; bbb = b; }
		else { p1 = mm.ye; p2 = mm.pi; bbb = bb; }
		double a[x][y]; double b[x][y];
		for (int x = 0; x < str; x++) {
			for (int y = 0; y < stlb; y++) {
				n = 1;
				printf("\n"); c++;
				while (n <= stlb) {
					if (p2 && n == p2->value && *(bbb + c) != -1) {
						a[x][y] = p1->value; b[y][x] = p1->value;
						p1 = p1->next;
						p2 = p2->next;
					}
					else {
						a[x][y] = 0.0; b[y][x] = 0.0;
					}
					n++;
				}
			}
			x = 0; y = 0; int prvrk1 = 0; while (x <= str && y <= stlb) { if (a[x][y] != 0) prvrk
= -1; x++; y++; }
			int prvrk2 = 0; for (int x = 0; x < str; x++) { for (int y = 0; y < stlb; y++) { if
(a[x][y] != -b[x][y]) prvrk2 = -1; } }
			n++;
		}
		}
	}
	return 0;
}


 void menu() {
		printf("=========================================\n");
		printf("||  1-Enter actual matrix            ||\n");
		printf("||  2-Output matrix in different types ||\n");
		printf("||  3-Tranform the actual matrix      ||\n");
		printf("||  4-Menu                            ||\n");
		printf("||  0-End                             ||\n");
		printf("=========================================\n");
		printf("\n");
 }

 int main() {
		int n, chk = 1;
		char ch = '9';
		file = fopen("matr.txt", "r");
```

```
	if (!file) {
		error2();
		return 1;
	}
	menu();
	while (ch != '0') {
		printf("=> ");
		scanf(" %c", &ch);
		switch (ch) {
		case '1':
			if (!enter())
				printf("The matrix was successfully read\n\n");
			chk = 0;
			break;
		case '2':
			output();
			printf("\n");
			break;
		case '3':
			if (!transform())
				printf("The matrix was successfully transformed\n\n");
			break;
		case '4':
			menu();
			break;
		}
	}
	return 0;
}
```

Тесты

Admin@LAPTOP-Q5U6S2UH:/mnt/c/Users/Admin/Desktop/Все для вуза$ vim matr.txt

Admin@LAPTOP-Q5U6S2UH:/mnt/c/Users/Admin/Desktop/Все для вуза$ cat matr.txt

0 0 3 0 0

2 0 0 0 0

0 0 0 2 0

0 4 0 0 0

0 0 0 0 5

```
Admin@LAPTOP-Q5U6S2UH:/mnt/c/Users/Admin/Desktop/Все для вуза$ gcc kursach7.c
Admin@LAPTOP-Q5U6S2UH:/mnt/c/Users/Admin/Desktop/Все для вуза$ ./a.out
========================================
||  1-Enter actual matrix            ||
||  2-Output matrix in different types ||
||  3-Tranform the actual matrix      ||
||  4-Menu                           ||
||  0-End                            ||
========================================
=> 1
The matrix was successfully read
=> 2
Matrix in the usual form :
  0.0   0.0   3.0   0.0   0.0
  2.0   0.0   0.0   0.0   0.0
  0.0   0.0   0.0   2.0   0.0
  0.0   4.0   0.0   0.0   0.0
  0.0   0.0   0.0   0.0   5.0
Matrix in internal view :
CIP : 0  1  2  3  4
PI  : 2    0    3    1    4
YE  : 3.0   2.0   2.0   4.0   5.0
=> 3
The matrix was successfully transformed
=> 2
Matrix in the usual form :
  5.0   0.0   0.0   0.0   0.0
  0.0   0.0   2.0   0.0   0.0
  0.0   0.0   0.0   0.0   3.0
  0.0   4.0   0.0   0.0   0.0
  0.0   0.0   0.0   2.0   0.0
```

```
Matrix in internal view :
CIP : 0  1  2  3  4
PI  : 0    2    4    1    3
YE  : 5.0   2.0   3.0   4.0   5.0
=> 0
Admin@LAPTOP-Q5U6S2UH:/mnt/c/Users/Admin/Desktop/Все для вуза$ vim matr.txt
Admin@LAPTOP-Q5U6S2UH:/mnt/c/Users/Admin/Desktop/Все для вуза$ cat matr.txt
0 0 3 0
0 2 0 0
0 0 2 0
0 4 0 0
Admin@LAPTOP-Q5U6S2UH:/mnt/c/Users/Admin/Desktop/Все для вуза$ ./a.out
==========================================
||   1-Enter actual matrix              ||
||   2-Output matrix in different types ||
||   3-Tranform the actual matrix       ||
||   4-Menu                             ||
||   0-End                              ||
==========================================
=> 1
The matrix was successfully read
=> 2
Matrix in the usual form :
  0.0   0.0   3.0   0.0
  0.0   2.0   0.0   0.0
  0.0   0.0   2.0   0.0
  0.0   4.0   0.0   0.0
Matrix in internal view :
CIP : 0  1  2  3
PI  : 3     1     2     1
YE  : 3.0   2.0   2.0   4.0
=> 3
The matrix was successfully transformed
```

```
=> 2
Matrix in the usual form :
  0.0   0.0   0.0   0.0
  0.0   2.0   0.0   3.0
  4.0   0.0   2.0   0.0
  0.0   0.0   0.0   0.0
Matrix in internal view :
CIP : -1  1  2 -1
PI  : 1    3     0     2
YE  : 2.0   3.0   4.0   2.0
=> 0
Admin@LAPTOP-Q5U6S2UH:/mnt/c/Users/Admin/Desktop/Все для вуза$ vim matr.txt
Admin@LAPTOP-Q5U6S2UH:/mnt/c/Users/Admin/Desktop/Все для вуза$ cat matr.txt
0 0 0 0
0 0 0 0
0 0 0 0
0 4 0 0
Admin@LAPTOP-Q5U6S2UH:/mnt/c/Users/Admin/Desktop/Все для вуза$ ./a.out
========================================
||   1-Enter actual matrix             ||
||   2-Output matrix in different types ||
||   3-Tranform the actual matrix      ||
||   4-Menu                            ||
||   0-End                             ||
========================================
=> 1
The matrix was successfully read
=> 2
Matrix in the usual form :
  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   4.0   0.0   0.0
```

Matrix in internal view :

CIP : -1 -1 -1 0

PI : 1

YE : 4.0

=>3

The matrix was successfully transformed

=> 2

Matrix in the usual form :

 0.0   0.0   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

Matrix in internal view :

CIP : -1 0 -1 -1

PI : 0

YE : 4.0

=> 0

Admin@LAPTOP-Q5U6S2UH:/mnt/c/Users/Admin/Desktop/Все для вуза$ vim matr.txt

Admin@LAPTOP-Q5U6S2UH:/mnt/c/Users/Admin/Desktop/Все для вуза$ cat matr.txt

0 0 0

0 0 0

0 0 0

./a.out

==========

|| 1-Enter actual matrix ||

|| 2-Output matrix in different types ||

|| 3-Tranform the actual matrix ||

|| 4-Menu ||

|| 0-End ||

==========

=> 1

The matrix was successfully read

=> 2

Matrix in the usual form :

0.0 0.0 0.0

0.0 0.0 0.0

0.0 0.0 0.0

Matrix in internal view :

All matrix elements are zero

CIP : -1 -1 -1

PI : empty

YE : empty

ЗАКЛЮЧЕНИЕ

Благодаря данному заданию курсового проекта я научился обрабатывать разреженные прямоугольные матрицы с элементами вещественного типа и размещать ненулевые элементы этих матриц в соответствии с различными схемами внутреннего представления разреженных матриц. Также я научился реализовывать односвязные списки и такому варианту физического представления, как отображение на динамические структуры.