Московский авиационный институт
(Национальный исследовательский университет)

Факультет прикладной математики и физики
Кафедра вычислительной математики и программирования

Задание VII по курсовому проекту

«Разреженные матрицы»

Руководитель: Никулин С.П.

_____
(оценка)

_____ _____
(подпись) (дата)

Студент группы 8О-104Б: Дюсекеев Алишер

Москва 2018

Задание VII

Разреженные матрицы

Постановка задачи:

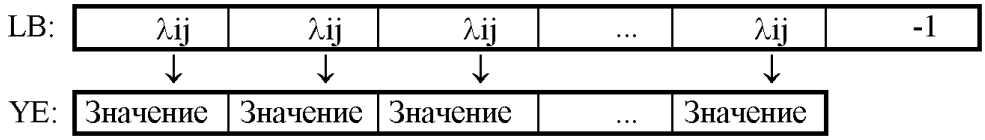
Составить программу на языке Си с функциями для обработки прямоугольных разреженных матриц с элементами вещественного типа, которая:

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.

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: все матрицы *m* x *n* хранятся по строкам, в порядке возрастания индексов ненулевых элементов.

4. Два вектора:



где $\lambda_{ij} = (i - 1) \times n + j - 1$

Вариант преобразования:

9. Найти столбец, содержащий наибольшее количество ненулевых элементов, и напечатать его номер и произведение элементов этого столбца. Если таких столбцов несколько обработать предпоследний.

Вариант физического представления:

1. Отображение на массив.

Общий метод решения:

Два вектора.

Каждому ненулевому элементу разреженной матрицы однозначно ставится в соответствие целое число λ_{ij} вида

$$\lambda_{ij} = (i - 1)*N + j,\ a_{ij} \neq 0.$$

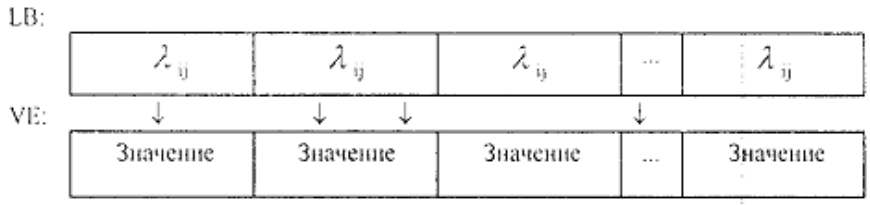
Статическое выделение памяти.

Хранение K ненулевых элементов обеспечивается двумя массивами:

YE – массив значений ненулевых элементов, расположенных по строкам;

LB – целочисленный массив коэффициентов λ_{ij}.

Оба массива содержат по K элементов и имеется однозначное соответствие между их элементами: в LB[m] находится λ_{ij}, соответствующее a_{ij} из YE[m], где m = 1, 2,…,K. Для хранения матрицы A по данной схеме требуется объем памяти 2*K ячеек.



Например, матрица A[5][5] будет хранится в виде

	a_{11}	a_{33}	a_{34}	a_{35}	a_{44}	a_{54}
YE = [	1.1,	2.2,	3.3,	4.4,	5.5,	6.6]
LB = [	1,	13,	14,	15,	19,	24]

При хранении по этой схеме исходная матрица может быть восстановлена следующим образом. В соответствии с данным выше определением λ_{ij} можно найти индекс строки i как наименьшее целое число, большее или равное отношению λ_{ij}, а индекс столбца по формуле

$$j = \lambda_{ij} - (i - 1)*N$$

kp7.exe

При запуске программы требуется ввести имя текстового файла, в котором содержится исходная прямоугольная матрица (также ее размер, записанный в первой строке). Далее отображается меню, в котором есть 4 опции:

1.Вывести матрицу в обычном виде на экран.

Вызывается вспомогательная функция printmatrix. Для отображения матрицы на экран нужно обработать каждый элемент матрицы. Для этого используется цикл, обрабатывающий матрицу построчно, со вложенным циклом, обрабатывающим по

столбцам. Если первый элемент массива LB равен i*n + j, т.е. код каждого элемента матрицы сравнивается с λ_{ij} до тех пор, пока не найдется совпадение, то печатается соответствующий элемент массива YE и далее сравнивается уже второй элемент LB с последующими элементами матрицы и т.д. В ином случае печатается ноль.

2.Вывести матрицу согласно схеме размещения.

Вызывается вспомогательная функция printvectors. Поочередно выводятся на экран с помощью циклов массивы YE, LB.

3.Выполнить задание.

Вызывается вспомогательная функция task. Создается вспомогательный массив columns целочисленного типа, который заполнен нулями.

Первый цикл восстанавливает координаты элементов, закодированных в массиве LB. Восстановленный индекс j сопоставляется массиву columns[j], после чего значение columns[j] увеличивается на единицу. Таким образом в columns каждый его индекс соответствует столбцу матрицы, и в каждом значении массива хранится число элементов столбца исходной матрицы.

Далее среди элементов columns находим максимальный элемент – max, и предшествующий максимальному – premax. Это и будут столбцы матрицы, содержащие наибольшее число элементов. По условию задачи, если в матрице находится несколько столбцов с одинаковым числом элементов, то выбирается предпоследний – premax. В обратном случае – max.

Произведение элементов выбранного столбца находится при помощи цикла, который пробегает все значения массива LB. Для этого создадим внутренний цикл который работает пока i*n + premax < LB[k=0,1,2,…]. Если во время выполнения внутреннего цикла выполняется равенство i*n + premax = LB[k=0,1,2,…], то подсчитывается произведение mult*=Y=LB[k], увеличивается значение k на единицу и значение флага меняется на единицу, что говорит о том, что в i-той строке столбца premax нашелся ненулевой элемент. Если же равенство не выполняется (об этом говорит значение флага равное нулю), то это значит в выбранном столбце есть нулевой элемент, и соответственно произведение элементов равно нулю.

4.Ввести матрицу.

Вызывается вспомогательная функция inputmatrix. Работа производится с текстовыми файлами. Программа запрашивает имя файла, которое при помощи fopen открывается на чтение. С помощью функции fscanf считываем первую строку исходного файла, в ней записан размер матрицы, где переменной m присваивается число строк, n – число столбцов. Далее считываем оставшиеся строки файла, пользуясь полученными ранее

размерами матрицы. Если элемент матрицы ненулевой, то записываем его и сопоставленное ему число в массивы LB, YE.

Общие сведения о программе:

Программное и аппаратное обеспечение для запуска данной программы на ПК не ограничено в выборе. Операционная система семейства Windows NT – Windows 10. Интерпретатор команд – cmd.exe. Интегрированная среда разработки – Visual Studio Express 2013. Язык программирования – C. Строк в программе – 159. Местонахождение текстового файла: C:\Users\Алишер\Desktop\прога\kp7. Имя файлов: kp7.c. Программа запускается через Visual Studio или вручную, после компиляции через командную строку через приложения kp7.exe.

Описание логической структуры:

1. Создаем несколько матриц, которые хранятся в текстовых файлах.
2. Запускаем приложение kp7.exe.
3. Вводим имя матрицы, с которой будет производится работа.
4. С матрицей можно произвести 4 действия:
   4.1. Вывести матрицу в обычном виде на экран.
   4.2. Вывести матрицу согласно схеме размещения.
   4.3. Выполнить задание.
   4.4. Ввести новую матрицу.

Таблица переменных и констант:

Название переменной	**Тип переменной**	**Назначение переменной**
m	int	Число строк в матрице
n	int	Число столбцов в матрице
LB	int[]	Целочисленный массив коэффициентов λ
YE	float[]	Массив ненулевых элементов матрицы
k	int	Размер массивов LB, YE
f	FILE*	Файл, в котором содержится матрица
num	float	Переменная, которой присваиваются значения матрицы при считывании файла

filename	char[]	Имя файла
i, j, s	int	Счетчики итератора
columns	int[]	Массив числа ненулевых элементов по столбцам матрицы
max	int	Номер столбца с максимальным числом элементов
premax	int	Номер столбца с предшествующим максимальному числу элементов
flag	int	Флаг
mult	float	Произведение элементов столбца матриц

Дополнительные функции:

- void printvectors(int* LB, float* YE, int k);
  Функция, которая выводит матрицу на экран согласно схеме размещения.
- void printmatrix(int n, int m, int* LB, float* YE)
  Функция, которая выводит матрицу в обычном виде на экран.
- void task(int* LB, float* YE, int k, int m, int n)
  Функция, которая находит столбец, содержащий наибольшее количество ненулевых элементов, и печатет его номер и произведение элементов этого столбца. Если таких столбцов несколько обрабатывается предпоследний.
- int inputmartix(int* LB, float* YE, int* k, int* m, int* n)
  Функция, которая обрабатывает текстовый файл и записывает матрицу в массивы согласно схеме размещения.
- int menu()
  Функция, которая выводит меню на экран и проверяет корректность выбора опции.

Входные данные:

Входные данные задаются в виде текстовых файлов. Поля матрицы разделяются табуляцией. В первой строке матрицы записан ее размер (1 поле – количество строк, n – количество столбцов). В качестве исходных файлов заданы файлы ниже.

matrix1.txt

5 4

0.1 1.8 0 0

0 0 0 7.77

0 0 0 0

1.34 0 3.84 0

matrix2.txt

3 6

9.1 0 0 0 2.23 0

3.33 0 1.6 0 4.9 0

5.54 0 0 0 2.61 0

matrix3.txt

5 5

1.2 0 0 0 0

0 1.3 0 5 0

0 0 1.3 0 0

0 0 0 1.4 0

0 0 0 0 1.5

Выходные данные:

Выходные данные зависят от выбора опции в меню, если выбрано 1 или 2, то на экран выводится матрица в обычном виде или согласно схеме размещения. Если выбрана опция 3, то на экран выводится сообщение, в котором содержится номер максимального столбца, причем отсчет столбцов матрицы начинается с единицы, а не с нуля, и произведение элементов этого столбца.

Программа на Си:

kp7.c

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <Windows.h>

int menu();
int inputmartix(int* LB, float* YE, int* k, int* m, int* n);
void printvectors(int* LB, float* YE, int k);
void printmatrix(int n, int m, int* LB, float* YE);
void task(int* LB, float* YE, int k, int m, int n);
int main(int argc, char* argv[])
{
	SetConsoleCP(1251);
	SetConsoleOutputCP(1251);
	int m, n, LB[15], k = 0;
	float YE[15];
	do{
```

```
	} while (inputmartix(LB, YE, &k, &m, &n));
	while (1)//цикл вечен
	{
		switch (menu())//на выбор какое-либо действие над базой данных из меню
		{
		case 0: return 0;//закрыли программу
		case 1: printmatrix(n, m, LB, YE); break;
		case 2: printvectors(LB, YE, k); break;
		case 3: task(LB, YE, k, m, n); break;
		case 4: inputmartix(LB, YE, &k, &m, &n); break;
		}
	}
	return 0;
}
int inputmartix(int* LB, float* YE, int* k, int* m, int* n)
{
	char filename[15];
	int i, j;
	float num;
	*k = 0;
	printf("\nВведите имя файла:");
	scanf("%s", filename);
	FILE *f = fopen(filename, "r");
	if (!f)
	{
		perror("Can't open file");
		return 1;
	}
	fscanf(f, "%d\t%d\n", m, n);
	if ((*m) * (*n) > 100)
	{
		printf("Недопустимый размер");
		return 3;
	}
	for (i = 0; i < *m; i++)
	{
		for (j = 0; j < *n; j++)
		{
			fscanf(f, "%f", &num);
			if (num != 0)
			{
				YE[*k] = num;
				LB[*k] = i*(*n) + j;
				(*k)++;
			}
		}
	}
	fclose(f);
	return 0;
}
void printvectors(int* LB, float* YE, int k)
{
	int i;
	printf("\nLB:");
	for (i = 0; i < k; i++)
	{
		printf("%d\t", LB[i]);
	}
	printf("\nYE:");
	for (i = 0; i < k; i++)
	{
		printf("%.2f\t", YE[i]);
	}
	printf("\n\n");
}
```

```
void printmatrix(int n, int m, int* LB, float* YE)
{
	int i, k, j;
	putchar('\n');
	for (i = 0, k = 0; i < m; i++)
	{
		for (j = 0; j < n; j++)
		{
			if (LB[k] == i*n + j)
			{
				printf("%.2f\t", YE[k++]);
			}
			else
				printf("0\t");
		}
		putchar('\n');
	}
	putchar('\n');
}
void task(int* LB, float* YE, int k, int m, int n)
{
	int columns[10], i, j, s, max = 0, premax = -1, flag = 0;
	float mult = 1;
	memset(columns, 0, 10 * sizeof(int));
	for (s = 0; s < k; s++)
	{
		i = LB[s] / n;
		j = LB[s] - i*n;
		columns[j]++;
	}
	for (s = 0; s < n; s++)
	{
		if (columns[s] >= columns[max])
		{
			premax = max;
			max = s;
		}
	}
	premax = (columns[premax] >= columns[max]) ? premax : max;
	for (i = 0, k = 0; i < m; i++)
	{
		while (i*n + premax >= LB[k])
		{
			if (i*n + premax == LB[k]){
				mult *= YE[k];
				flag = 1;
				k++;
			}
			else
				k++;
		}
		if (flag == 0)
		{
			mult = 0;
			break;
		}
		flag = 0;
	}
	printf("\nСтолбец, содержащий наибольшее кол-во ненулевых элементов -
%d\nПроизведение элементов этого столбца - %.2f\n\n", premax + 1, mult);
}
int menu()
{
	int k;
	printf("0:Закончить работу\n");
```

```
	printf("1:Вывести матрицу в обычном виде на экран\n");
	printf("2:Вывести матрицу согласно схеме размещения\n");
	printf("3:Выполнить задание\n");
	printf("4:Ввести матрицу\n");
	printf(":>");
	scanf("%d", &k);
	while (k < 0 || k>4)//пока переменная k за пределами допустимого...
	{
		printf("Ошибка. Вы ввели номер действия, которого не существует\n"//выдаем
ошибку
			"Попробуйте ввести снова\n:>");
		scanf("%d", &k);//вводим еще раз
	}
	return k;//возвращаем номер нужного  действия
}
```

Демонстрация работы программы:

```
C:\Users\Антон\Desktop>kp7.exe

Введите имя файла:matrix1.txt
0:Закончить работу
1:Вывести матрицу в обычном виде на экран
2:Вывести матрицу согласно схеме размещения
3:Выполнить задание
4:Ввести матрицу
:>1

0.10    1.80    0       0
0       0       0       7.77
0       0       0       0
1.34    0       3.84    0
0       2.00    0       0

0:Закончить работу
1:Вывести матрицу в обычном виде на экран
2:Вывести матрицу согласно схеме размещения
3:Выполнить задание
4:Ввести матрицу
:>2
```

```
LB:0    1       7       12      14      17
YE:0.10 1.80    7.77    1.34    3.84    2.00

0:Закончить работу
1:Вывести матрицу в обычном виде на экран
2:Вывести матрицу согласно схеме размещения
3:Выполнить задание
4:Ввести матрицу
:>3

Столбец, содержащий наибольшее кол-во ненулевых элементов - 1
Произведение элементов этого столбца - 0.00

0:Закончить работу
1:Вывести матрицу в обычном виде на экран
2:Вывести матрицу согласно схеме размещения
3:Выполнить задание
4:Ввести матрицу
:>4

Введите имя файла:matrix2.txt
0:Закончить работу
1:Вывести матрицу в обычном виде на экран
2:Вывести матрицу согласно схеме размещения
3:Выполнить задание
4:Ввести матрицу
:>1

9.10    0       0       0       2.23    0
3.33    0       1.60    0       4.90    0
5.54    0       0       0       2.61    0
```

```
0:Закончить работу
1:Вывести матрицу в обычном виде на экран
2:Вывести матрицу согласно схеме размещения
3:Выполнить задание
4:Ввести матрицу
:>2

LB:0    4       6       8       10      12      16
YE:9.10 2.23    3.33    1.60    4.90    5.54    2.61

0:Закончить работу
1:Вывести матрицу в обычном виде на экран
2:Вывести матрицу согласно схеме размещения
3:Выполнить задание
4:Ввести матрицу
:>3

Столбец, содержащий наибольшее кол-во ненулевых элементов - 1
Произведение элементов этого столбца - 167.88

0:Закончить работу
1:Вывести матрицу в обычном виде на экран
2:Вывести матрицу согласно схеме размещения
3:Выполнить задание
4:Ввести матрицу
:>4

Введите имя файла:matrix3.txt
0:Закончить работу
1:Вывести матрицу в обычном виде на экран
2:Вывести матрицу согласно схеме размещения
```

```
3:Выполнить задание
4:Ввести матрицу
:>1

1.20    0       0       0       0
0       1.30    0       5.00    0
0       0       1.30    0       0
0       0       0       1.40    0
0       0       0       0       1.50

0:Закончить работу
1:Вывести матрицу в обычном виде на экран
2:Вывести матрицу согласно схеме размещения
3:Выполнить задание
4:Ввести матрицу
:>2

LB:0    6       8       12      18      24
YE:1.20 1.30    5.00    1.30    1.40    1.50

0:Закончить работу
1:Вывести матрицу в обычном виде на экран
2:Вывести матрицу согласно схеме размещения
3:Выполнить задание
4:Ввести матрицу
:>3

Столбец, содержащий наибольшее кол-во ненулевых элементов - 4
Произведение элементов этого столбца - 0.00

0:Закончить работу
1:Вывести матрицу в обычном виде на экран
```

```
2:Вывести матрицу согласно схеме размещения
3:Выполнить задание
4:Ввести матрицу
:>7
Ошибка. Вы ввели номер действия, которого не существует
Попробуйте ввести снова
:>4

Введите имя файла:matrix
Can't open file: No such file or directory
0:Закончить работу
1:Вывести матрицу в обычном виде на экран
2:Вывести матрицу согласно схеме размещения
3:Выполнить задание
4:Ввести матрицу
:>0
```

Вывод:

Разреженные матрицы используются для хранения сравнительно небольшого объема данных, которые располагаются в большой области данных. Реализация разреженных матриц связана со значительными издержками, и это делает их все более непрактичными по мере заполнении области данных значимыми величинами, и в тоже время при большей разреженности эти структуры становятся более эффективными.

Были разобраны 4 варианта представления разреженных матриц:

1. Цепочка ненулевых элементов в векторе со строчным индексированием.
2. Один вектор
3. Три вектора
4. Два вектора

На мой взгляд, наиболее удобным и экономичным является 4 метод. Как показала практика, с ним очень легко работать, и он самый эффективный по использованию памяти.