Отчет по курсовой работе №7

по курсу: языки и методы программирования

студент группы : М8О-105Б-21 Козлов Егор Сергеевич, № по списку: 9

Адреса www, e-mail, jabber, skype: iamaghoulzxc@gmail.com

Работа выполнена: "24 мая 2022 г."

Преподаватель: каф. 806 В.К.Титов

Входной контроль знаний с оценкой: _____

Отчет сдан "___"_____20___г., итоговая оценка _____

Подпись преподавателя _____

1. Тема: Разреженные матрицы.

2. Цель работы: Написать программу с процедурами и/или функциями для обработки прямоугольных разреженных матриц с элементом вещественного, целого и комплексного числа на языке Си.

3. Задание (*вариант 9*): Вариант размещения матрицы: 1. Цепочка ненулевых элементов в векторе A со строчным индексированием. Вариант преобразований: Найти столбец, содержащий наибольшее количество ненулевых элементов, и напечатать его номер и произведение элементов этого столбца. Если таких столбцов несколько обработать предпоследний.

4. Оборудование(*лабораторное*):
ЭВМ _____, процессор _____, имя узла сети ____ с ОП ___ ГБ
НМД _____ ГБ. Терминал _____ адрес _____. Принтер _____
Другие устройства _____
Оборудование ПЭВМ студента, если использовалось:
Процессор Ryzen 7 5800 @ 8x 3.2 GHz , ОП 16384 МБ, НМД _____ГБ. Монитор Встроенный
Другие устройства _____

5. Программное обеспечение(*лабораторное*):
Операционная система семейства UNIX, наименование _____ версия _____
Интерпретатор команд: _____ версия _____
Система программирования: _____ версия _____
Редактор текстов: _____ версия _____
Утилиты операционной системы: _____
Прикладные системы и программы: _____
Местонахождение и имена файлов и программ данных: _____

Программное обеспечение ЭВМ студента, если использовалось:
Операционная система семейства UNIX, наименование Ubuntu версия 22.04
Интерпретатор команд: bash версия _____
Система программирования: C версия _____
Редактор текстов: Emacs версия _____
Утилиты операционной системы: _____
Прикладные системы и программы: _____
Местонахождение и имена файлов и программ данных: /usr/bin , а также /bin

6. Идея, метод, алгоритм решения задачи (в формах: словесной, псевдокода, графической [блок-схема, диаграмма, рисунок, таблица] или формальное описание с пред- и постусловиями)

На вход подается файл, который содержит необходимые данные.

- размер прямоугольной матрицы;
- количество ненулевых элементов;
- данные матрицы представленные в виде цепочки ненулевых элементов в векторе A со строчным индексированием.

После считывания всех данных программа начинает свою работу. Выводится матрица в виде цепочки ненулевых элементов в векторе A со строчным индексированием, какой она и была введена. Затем она преобразуется в прямоугольную матрицу и тоже выводится.

Выполняется преобразование. Проходимся по столбцам матрицы и считаем ненулевые элементы, фиксируя каждый раз максимальное значение и номер этого столбца. Считаем произведение элементов в столбце. Выводим значение произведения элементов столбца с максимальным количеством ненулевых элементов и номер этого столбца.

7. Сценарий выполнения работы [план работы, первоначальный текст программы в черновике (можно на отдельном листе) и тесты, либо соображения по тестированию].

kp7.cpp :

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

void printShortMatrix(int *M, float *A, int cntLn, int cntNz) {
    printf("\nShort matrix\n");
    for (int i = 0; i < cntLn; ++i) {
        printf("%3d", M[i]);
    }
    printf("\n");
    for (int i = 0; i < 2 * cntNz; ++i) {
        printf("%2.1f ", A[i]);
    }
    printf("\n");
}

void printFullMatrix(float **matrix, int n, int m) {
    printf("\nFull matrix\n");
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < m; ++j) {
            printf("%2.1f ", matrix[i][j]);
        }
        printf("\n");
    }
    printf("\n");
}

void task(float **matrix, int n, int m) {
    int maxNz = 0, nz = 0, colInd = 0, preCol = 0;
    float multiply = 1;

    for (int j = 0; j < m; ++j) {
        for (int i = 0; i < n; ++i) {
            if (matrix[i][j] != 0) {
                ++nz;
            }
        }
        if (nz >= maxNz) {
            if (nz == maxNz) {
                preCol = colInd;
            } else preCol = j;
            maxNz = nz;
            colInd = j;
        }
        nz = 0;
    }
    for (int i = 0; i < n; ++i) {
        multiply *= matrix[i][preCol];
    }
    printf("\nTask answer:\nmultiply: %2.2f\ncolumn: %d\n", multiply, preCol);
}
```

```
51
52 void shortToFull(float **matrix, int *M, float *A, int cntLn, int cntNz) {
53     int start, end, cur, i, j, lastInd;
54     for (i = 0; i < cntLn; ++i) {
55         if (M[i] == 0) continue;
56         start = M[i];
57         lastInd = i - (1 * M[i] == 0);
58         for (cur = i + 1; (cur < cntLn - 1) && (M[cur] == 0); ++cur);
59         end = M[cur];
60         for (j = start - 1; j < end - 1; ++j) {
61             matrix[i][int(A[j])] = A[j + 1];
62             ++j;
63         }
64     }
65     //last pass
66     for (j = M[lastInd]-1; j < 2 * cntNz; ++j) {
67         matrix[lastInd][int(A[j])] = A[j + 1];
68         ++j;
69     }
70 }
71
72 void fullToShort(float **matrix, int *M, float *A, int cntNz, int n, int m) {
73     for (int i = 0; i < n; ++i) {
74         M[i] = 0;
75     }
76     for (int i = 0; i < cntNz * 2; ++i) {
77         A[i] = 0;
78     }
79     int cnt = 0, zero = 0;
80     for (int i = 0; i < n; ++i) {
81         for (int j = 0; j < m; ++j) {
82             if (matrix[i][j] != 0) {
83                 if (!zero) {
84                     M[i] = cnt + 1;
85                     zero = 1;
86                 }
87                 A[cnt++] = j;
88                 A[cnt++] = matrix[i][j];
89             }
90         }
91         zero = 0;
92     }
93 }
94
95 int main() {
96     float *A, **matrix;
97     int *M, cntNz, n, m;
98
99     scanf("%d", &n);
100    scanf("%d", &m);
101    scanf("%d", &cntNz);
102
103    matrix = new float *[n];
104    for (int i = 0; i < n; ++i) {
105        matrix[i] = new float[m];
106    }
107    M = new int[n];
108    A = new float[3 * cntNz];
109
110    for (int i = 0; i < n; ++i) {
111        scanf("%d", &M[i]);
112    }
113    for (int i = 0; i < 2 * cntNz; ++i) {
114        scanf("%f", &A[i]);
115    }
116
117    printShortMatrix(M, A, n, cntNz);
118    shortToFull(matrix, M, A, n, cntNz);
119    printFullMatrix(matrix, n, m);
120    task(matrix, n, m);
121    fullToShort(matrix, M, A, cntNz, n, m);
122    printShortMatrix(M, A, n, cntNz);
123    return 0;
124 }
```

Пункты 1-7 отчѐта составляются ***строго до*** *начала лабораторной работы.*

Допущен к выполнению работы. Подпись преподавателя _____

8. Распечатка протокола (подклеить листинг окончательного варианта программы с текстовыми примерами, подписанный преподавателем).

```
isitmuse@isitmuse:~/lab/secondSem/kp7$ cat head
--------------------------------
||||||КУРСОВАЯ  РАБОТА  №7||||||
||||||РАЗРЕЖЕННЫЕ  МАТРИЦЫ||||||
|||ВЫПОЛНИЛ  СТУДЕНТ  ГРУППЫ||||
|||М8О-105Б-21  КОЗЛОВ  ЕГОР||||
\\\\\\\\\\\\\\\\||/////////////////
isitmuse@isitmuse:~/lab/secondSem/kp7$ cat kp7.cpp
#include <stdio.h>
#include <stdlib.h>

void printShortMatrix(int *M, float *A, int cntLn, int cntNz) {
    printf("\nShort matrix\n");
    for (int i = 0; i < cntLn; ++i) {
        printf("%3d", M[i]);
    }
    printf("\n");
    for (int i = 0; i < 2 * cntNz; ++i) {
        printf("%2.1f ", A[i]);
    }
    printf("\n");
}

void printFullMatrix(float **matrix, int n, int m) {
    printf("\nFull matrix\n");
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < m; ++j) {
            printf("%2.1f ", matrix[i][j]);
        }
        printf("\n");
    }
    printf("\n");
}

void task(float **matrix, int n, int m) {
    int maxNz = 0, nz = 0, colInd = 0, preCol = 0;
    float multiply = 1;

    for (int j = 0; j < m; ++j) {
        for (int i = 0; i < n; ++i) {
            if (matrix[i][j] != 0) {
                ++nz;
            }
        }
        if (nz >= maxNz) {
            if (nz == maxNz) {
                preCol = colInd;
            } else preCol = j;
            maxNz = nz;
            colInd = j;
        }
        nz = 0;
    }
    for (int i = 0; i < n; ++i) {
        multiply *= matrix[i][preCol];
    }
    printf("\nTask answer:\nmultiply: %2.2f\ncolumn: %d\n", multiply, preCol);
}

void shortToFull(float **matrix, int *M, float *A, int cntLn, int cntNz) {
```

```
    int start, end, cur, i, j, lastInd;
    for (i = 0; i < cntLn; ++i) {
        if (M[i] == 0) continue;
        start = M[i];
        lastInd = i - (1 * M[i] == 0);
        for (cur = i + 1; (cur < cntLn - 1) && (M[cur] == 0); ++cur);
        end = M[cur];
        for (j = start - 1; j < end - 1; ++j) {
            matrix[i][int(A[j])] = A[j + 1];
            ++j;
        }
    }
    //last pass
    for (j = M[lastInd]-1; j < 2 * cntNz; ++j) {
        matrix[lastInd][int(A[j])] = A[j + 1];
        ++j;
    }
}

void fullToShort(float **matrix, int *M, float *A, int cntNz, int n, int m) {
    for (int i = 0; i < n; ++i) {
        M[i] = 0;
    }
    for (int i = 0; i < cntNz * 2; ++i) {
        A[i] = 0;
    }
    int cnt = 0, zero = 0;
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < m; ++j) {
            if (matrix[i][j] != 0) {
                if (!zero) {
                    M[i] = cnt + 1;
                    zero = 1;
                }
                A[cnt++] = j;
                A[cnt++] = matrix[i][j];
            }
        }
        zero = 0;
    }
}

int main() {
    float *A, **matrix;
    int *M, cntNz, n, m;

    //printf("Enter n\n");
    scanf("%d", &n);
    //printf("Enter m\n");
    scanf("%d", &m);
    //printf("Enter cntNz\n");
    scanf("%d", &cntNz);

    matrix = new float *[n];
    for (int i = 0; i < n; ++i) {
        matrix[i] = new float[m];
    }
    M = new int[n];
    A = new float[3 * cntNz];

    //printf("Enter short matrix\n");
    for (int i = 0; i < n; ++i) {
        scanf("%d", &M[i]);
```

```
    }
    for (int i = 0; i < 2 * cntNz; ++i) {
        scanf("%f", &A[i]);
    }

    printShortMatrix(M, A, n, cntNz);
    shortToFull(matrix, M, A, n, cntNz);
    printFullMatrix(matrix, n, m);
    fullToShort(matrix, M, A, cntNz, n, m);
    printShortMatrix(M, A, n, cntNz);

    task(matrix, n, m);

    return 0;
}isitmuse@isitmuse:~/lab/secondSem/kp7$ cat test
4
8
11
1 5 11 17
0 7.3 4 6 0 3.1 3 2 4 1.1 0 1.7 4 3 5 1 0 2.9 3 1 4 1
isitmuse@isitmuse:~/lab/secondSem/kp7$ cat test2
6
7
14
0 1 13 19 23 0
0 1.2 2 3 3 4 4 7.1 5 2 6 3.4 1 1.3 2 7 6 3.1 0 2 6 7 0 3.2 1 1 2 2.8
isitmuse@isitmuse:~/lab/secondSem/kp7$ cat test4
5
5
6
0 0 1 7 11
0 7 1 10 3 3 3 1 4 1.5 1 7
isitmuse@isitmuse:~/lab/secondSem/kp7$ cat test5
6
7
11
0 1 13 19 0 0
0 1.2 2 3 3 4 4 7.1 5 2 6 3.4 1 1.3 2 7 6 3.1 0 2 6 7
isitmuse@isitmuse:~/lab/secondSem/kp7$ g++ -o kp7 kp7.cpp
isitmuse@isitmuse:~/lab/secondSem/kp7$ ./kp7 < test

Short matrix
  1  5 11 17
0.0 7.3 4.0 6.0 0.0 3.1 3.0 2.0 4.0 1.1 0.0 1.7 4.0 3.0 5.0 1.0 0.0 2.9 3.0 1.0 4.0 1.0

Full matrix
7.3 0.0 0.0 0.0 6.0 0.0 0.0 0.0
3.1 0.0 0.0 2.0 1.1 0.0 0.0 0.0
1.7 0.0 0.0 0.0 3.0 1.0 0.0 0.0
2.9 0.0 0.0 1.0 1.0 0.0 0.0 0.0


Short matrix
  1  5 11 17
0.0 7.3 4.0 6.0 0.0 3.1 3.0 2.0 4.0 1.1 0.0 1.7 4.0 3.0 5.0 1.0 0.0 2.9 3.0 1.0 4.0 1.0

Task answer:
multiply: 111.57
column: 0
isitmuse@isitmuse:~/lab/secondSem/kp7$ ./kp7 < test2

Short matrix
  0  1 13 19 23  0
```

```
0.0 1.2 2.0 3.0 3.0 4.0 4.0 7.1 5.0 2.0 6.0 3.4 1.0 1.3 2.0 7.0 6.0 3.1 0.0 2.0 6.0 7.0 0.0 3.2 1.0 1.0

Full matrix
0.0 0.0 0.0 0.0 0.0 0.0 0.0
1.2 0.0 3.0 4.0 7.1 2.0 3.4
0.0 1.3 7.0 0.0 0.0 0.0 3.1
2.0 0.0 0.0 0.0 0.0 0.0 7.0
3.2 1.0 2.8 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0


Short matrix
  0  1 13 19 23  0
0.0 1.2 2.0 3.0 3.0 4.0 4.0 7.1 5.0 2.0 6.0 3.4 1.0 1.3 2.0 7.0 6.0 3.1 0.0 2.0 6.0 7.0 0.0 3.2 1.0 1.0

Task answer:
multiply: 0.00
column: 2
isitmuse@isitmuse:~/lab/secondSem/kp7$ ./kp7 < test4

Short matrix
  0  0  1  7 11
0.0 7.0 1.0 10.0 3.0 3.0 3.0 1.0 4.0 1.5 1.0 7.0

Full matrix
0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0
7.0 10.0 0.0 3.0 0.0
0.0 0.0 0.0 1.0 1.5
0.0 7.0 0.0 0.0 0.0


Short matrix
  0  0  1  7 11
0.0 7.0 1.0 10.0 3.0 3.0 3.0 1.0 4.0 1.5 1.0 7.0

Task answer:
multiply: 0.00
column: 1
isitmuse@isitmuse:~/lab/secondSem/kp7$ ./kp7 < test5

Short matrix
  0  1 13 19  0  0
0.0 1.2 2.0 3.0 3.0 4.0 4.0 7.1 5.0 2.0 6.0 3.4 1.0 1.3 2.0 7.0 6.0 3.1 0.0 2.0 6.0 7.0

Full matrix
0.0 0.0 0.0 0.0 0.0 0.0 0.0
1.2 0.0 3.0 4.0 7.1 2.0 3.4
0.0 1.3 7.0 0.0 0.0 0.0 3.1
2.0 0.0 0.0 0.0 0.0 0.0 7.0
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


Short matrix
  0  1 13 19  0  0
0.0 1.2 2.0 3.0 3.0 4.0 4.0 7.1 5.0 2.0 6.0 3.4 1.0 1.3 2.0 7.0 6.0 3.1 0.0 2.0 6.0 7.0

Task answer:
multiply: 0.00
column: 6
isitmuse@isitmuse:~/lab/secondSem/kp7$
```

9. Дневник отладки должен содержать дату и время сеансов отладки, и основные ошибки (ошибки в сценарии и программе, не стандартные операции) и краткие комментарии к ним. В дневнике отладки приводятся сведения об использовании других ЭВМ, существенном участии преподавателя и других лиц в написании и отладке программы.

№	Лаб или дом.	Дата	Время	Событие	Действие по исправлению	Примечание

10. Замечание автора по существу работы _____

_____

11. Выводы Я научился работать с разреженными матрицами.

Недочеты, допущенные при выполнении задания, могут быть устранены следующим образом _____

_____

Подпись студента _____