Отчет по лабораторной работе №7 по курсу

Языки и методы программирования _____

Студент группы М8О-101Б-21 Постнов Александр Вячеславович, № по списку 17

Контакты www, e-mail: 61pav03@mail.ru

Работа выполнена: «» 202 2г.

Преподаватель: ___ каф. 806 _____Титов В.К._____

Входной контроль знаний с оценкой _____

Отчет сдан « » _____ 2022_ г., итоговая оценка _____

Подпись преподавателя _____

1. **Тема:** Разреженные матрицы_____
   _____
2. **Цель работы:** Составить программу на языке Си с процедурами и/или функциями для обработки прямоугольных разреженных матриц_с элементами целого типа(по согласованию с преподавателем)_____
   _____
   _____
3. **Задание** (*вариант схемы размещения матрицы №((17 + 3) % 4 + 1 = 1), вариант преобразования №((17 - 1) % 11 + 1 = 6)*)**:**
   6. Вычислить сумму двух разреженных матриц. Проверить, не является ли полученная матрица симметричной.

Варианты схемы размещения матрицы: все матрицы $m \times n$ хранятся *по строкам*, в порядке возрастания индексов ненулевых элементов.

1. Цепочка ненулевых элементов в векторе *A* со строчным индексированием (индексы в массиве *M* равны 0, если соответствующая строка матрицы содержит только нули)

M:	Индекс начала 1-ой строки в массиве A	Индекс начала 2-й строки	...	Индекс начала N-ой Строки

A:	Номер столбца	Значение	Индекс следующего ненулевого элемента этой строки (или 0)	Номер столбца	Значение	Индекс следующего ненулевого элемента этой строки (или 0)	...

Индекс, равный нулю, означает отсутствие ненулевых элементов в строке (или в ее остатке).
Если матрицы не изменяются программой, возможна экономия памяти за счет отказа от хранения в массиве *A* индексов следующего элемента столбца (когда элементы идут подряд). Вставка и удаление при этом способе возможны, но чересчур дороги: число перестановок элементов составит $O(N)$ вместо $O(1)$.

4. **Оборудование(лабораторное)**:
   ЭВМ -, процессор -, имя узла сети - с ОП - ГБ,
   НМД - ГБ, терминал- адрес -, принтер -
   Другие устройства -

   Оборудование ПЭВМ студента, если использовалось:
   Процессор AMD Ryzen 5 4500U, с ОП 8 ГБ
   Другие устройства -

5. **Программное обеспечение:**
   Операционная система семейства -, наименование - версия -
   интерпретатор команд - версия
   Система программирования - версия -
   Редактор текстов - версия -
   Утилиты операционной системы -
   Прикладные системы и программы -
   Местонахождение и имена файлов программ и данных -

Программное обеспечение ЭВМ студента, если использовалось:

Операционная система семейства GNU/Linux, наименование Manjaro версия 5-13-12-1

интерпретатор команд GNOME Terminal версия 3.38.2.

Система программирования _____версия_____

Редактор текстов emacs версия 3.27.20

Утилиты операционной системы

Прикладные системы и программы -

Местонахождение и имена файлов программ и данных -

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

Предложенную схему размещения матрицы я чуть-чуть изменил:

1) В массиве M я храню не индекс начала новой строки, а кол-во ненулевых элементов в строке(0 - в строке
2) Я не храню индекс следующего элемента(потому что я не изменяю матрицу, а записываю результат в новую матрицу)
3) Я не записываю индекс столбца и значение в один массив, а разделяю на 2 массива (A - индекс столбцов, а V - значение столбца)

В целом модель не изменилась, так как содержит такую же идею. Но ячейке в строке могут быть описаны в разном порядке(не обязательно по возрастанию). Это не влияет на саму матрицу, так как все однозначно идентифицируется(матрица).

Программа принимает значения 2 переменных n и m - размеры двух матриц, которые нужно сложить(складывать матрицы можно только одинаковых размеров)

Дальше записываем массивы M(содержит информацию о кол-ве ненулевых элементов в строке), A(индексы столбцов ненулевых элементов), V(значения столбцов)

Я реализовал функции для работы с этой схемой размещения:

1) **print_razr(int *M, int *A, int *V, int n, int m)** - просто вывожу массивы M, A, V(каждый с новой строки)
2) **to_normal(int **matrix, int *M, int *A, int *V, int n, int m)** –преобразования из разреженной матрицы в полную.

   Алгоритм: прохожу 1 циклом по массиву M, 2 циклом по значению текущей ячейки M, и просто записываю в столбцы значения(индексы столбцов беру из массива A, значения столбца из массива V, а строка - номер текущей ячейки в массиве M)
3) **semetr_razr(int *M, int *A, int *V, int n, int m)** - проверка на то, что разреженная матрица является симметричной. Если бы матрица была бы полной, я бы в цикле делал проверку вот так : “matrix[i][j] == matrix[j][i]”, т.е. номер строки и номер столбца поменял бы местами, и проверил их на одинаковое значение. С разреженными матрицами проверка будет идентична. Сначала прохожу по матрице M, потом прохожу по текущей ячейке матрицы M, и на любом шаге у меня информация о номере строки(текущий номер ячейки), номер столбца(текущее значение элемента массива A), значение в ячейке(текущее значение элемента массива V). Что нужно делать дальше? Мне нужно найти элемент в этой матрице, но необходимо поменять номер строки и номер столбца местами. Поэтому делаю опять обход по этим массивам и ищу этот элемент. Если такого элемента не существует(т.е он нулевый), или значение не совпадает, значит матрица не симметрична. Возвращаю false. В конце возвращаю true(т.к. все проверки прошли успешно). **Очевидно, что на проверку будут идти только квадратные матрицы(иначе они не могут быть симметричными)**
4) **summ_razr(int *&M, int *&A, int *&V, int *M1, int *A1, int *V1, int *M2, int *A2, int *V2, int n, int m)** - сложение двух разреженных матриц(запись в новые массивы M, A, V). Сначала я нахожу кол-во ненулевых элементов(складываю ячейки M1 и M2). Кол-во ненулевых элементов в новой матрице не больше, чем кол-во ненулевых элементов в матрицах, которые складываются. В массиве M кол-во элементов не изменится, так как при сложении матриц кол-во строк не изменяется, а вот значения элементов изменятся(а могут и не изменится, если сложим нулевую матрицу). Поэтому создадим временные массивы A_temp, V_temp, в них кол–во элементов будет сумма кол-ва ненулевых элементов матриц. Дальше проходим по строкам каждой из матриц. На каждом шагу изменения строки заводим массив temp, в котором я буду записывать рассмотренные столбцы. Сначала запускаю двойной цикл относительно 1 матрицы по 2 матрице(по текущему номеру строки). Буду искать одинаковые столбцы на

строке и складывать значения, записывать данные во временные массивы A_temp, V_temp, увеличивать счетчик ненулевых элементов итоговой матрицы(эти все действия проводятся, если значение ячейки не нулевое). Запускаю двойной цикл относительно 2 матрицы по 1 матрице. Произвожу те же действия, но проверяю, что этот столбец я не рассматривал(проверяю, что столбец не встречался в массиве temp).

Выделяю память(кол-во ненулевых элементов в итоговой матрице) для массивов A, V(итоговые массивы)

Записываю информацию из A_temp, V_temp. Действие алгоритма окончено.

Также для сравнение результатов сделал такие же функции(реализуются значительно проще) для обработки обычных(полных) матриц.

Опишу, что происходит в главной функции main:

1) Сначала записываю значения переменных n, m - размеры 2 матриц.
2) Записываю 2 матрицы в разреженном виде. (массивы M_1, A_1, V_1, M_2, A_2, V_2)
3) Создаю 2 матрицы в полном виде с помощью функции to_normal(описана выше)
4) Вывожу эти матрицы сначала в разреженном виде, потом в полном
5) складываю в разреженном виде, перевожу в полную форму, вывожу их, пишу симметричная матрица или нет, используя алгоритм для разреженной матрицы
6) складываю в полном виде, перевожу в разреженную форму, вывожу, пишу симметричная матрица или нет, используя алгоритм для полной матрицы.

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

```
#include <stdio.h>


void print_razr(int *M, int *A, int *V, int n, int m) { //Выводим разреженную матрицу
   int NOZERO = 0;
   for (int i = 0; i < n; i++){
      printf("%d ", M[i]);
      NOZERO += M[i];
   }
   printf("\n");
   for (int i = 0; i < NOZERO; i++){
      printf("%d ", A[i]);
   }
   printf("\n");
   for (int i = 0; i < NOZERO; i++){
      printf("%d ", V[i]);
   }
   printf("\n");
}

void print_normal(int **matrix, int n, int m) { //Выводим нормальную матрицу
   for (int i = 0; i < n; i++) {
      for (int j = 0; j < m; j++){
         printf("%d ", matrix[i][j]);
      }
      printf("\n");
   }
```

```
}

void to_razr(int **matrix, int *&M, int *&A, int *&V, int n, int m){ //преобразуем нормальную матрицу в разреженную
   M = new int[n]; //информация про строки -- кол-во ненулевых элементов в строке
   int NOZERO = 0; //Кол-во ненулевых элементов в матрице
   for (int i = 0; i < n; i++){
      for (int j = 0; j < m; j++){
         if (matrix[i][j] != 0) {
            M[i] += 1;
            NOZERO += 1;
         }
      }
   }
   A = new int[NOZERO];
   V = new int[NOZERO];
   int counter = 0;
   for (int i = 0; i < n; i++){
      for (int j = 0; j < m; j++){
         if (matrix[i][j] != 0) {
            A[counter] = j;
            V[counter] = matrix[i][j];
            counter++;
         }
      }
   }
}

void to_normal(int **matrix, int *M, int *A, int *V, int n, int m){ //пребразование из разреженной в полную
   int counter = 0; //указатель на элемент массива A, V
   for (int i = 0; i < n; i++){
      for (int j = 0; j < M[i]; j++){
         int stolbik = A[counter];
         int value = V[counter];
         matrix[i][stolbik] = value;
         counter++;
      }
   }
}

void summ_razr(int *&M, int *&A, int *&V, int *M1, int *A1, int *V1, int *M2, int *A2, int *V2, int n, int m) { //Сложение двух razr матриц
   int *A_temp, *V_temp;
   M = new int[n];

   int nozero_1 = 0;
   int nozero_2 = 0;

   for (int i = 0; i < n; i++) {
      nozero_1 += M1[i];
      nozero_2 += M2[i];
   }

   int all_nozero = nozero_1 + nozero_2; //максимальное возможное кол-во ненулевых элементов
   //printf("Кол-во ненулевых: %d\n", all_nozero);
   A_temp = new int[all_nozero];
```

```
V_temp = new int[all_nozero];
int counter_1 = 0;
int counter_2 = 0;
int counter = 0;
for (int k = 0; k < n; k++) {
  int *temp;
  temp = new int[M1[k] + M2[k]]; //текущая информация о рассмотренных столбиках строки
  for (int i = 0; i < M1[k] + M2[k]; i++) {
    temp[i] = -1;
  }
  int counter_temp = 0;
  int no_zero_temp = 0;
  for (int i = counter_1; i < M1[k] + counter_1; i++) {
    int a_temp = A1[i];
    int v_temp = V1[i];
    for (int j = counter_2; j < M2[k] + counter_2; j++) {
      if (a_temp == A2[j]) {
        v_temp += V2[j];
      }
    }
    if (v_temp != 0) {
      A_temp[counter] = a_temp;
      V_temp[counter] = v_temp;
      temp[counter_temp] = a_temp;
      counter++;
      no_zero_temp++;
      counter_temp++;
    }
  }
  for (int i = counter_2; i < M2[k] + counter_2; i++) {
    int a_temp = A2[i];
    int v_temp = V2[i];
    int flag = 0; //нужно сделать проверку, что этот ненулевый столбик не проверяли в текущей строке
    for (int i = 0; i < M1[k] + M2[k]; i++) {
      if (a_temp == temp[i]) { //уже рассматривали этот столбик
        flag = 1;
        break;
      }
    }
    if (flag == 0){ //дальше как обычно
      for (int j = counter_1; j < M1[k] + counter_1; j++) {
        if (a_temp == A1[j]) {
          v_temp += V1[j];
        }
      }
      if (v_temp != 0) {
        A_temp[counter] = a_temp;
        V_temp[counter] = v_temp;
        counter++;
        no_zero_temp++;
      }
    }

  }
```

```
      counter_1 += M1[k];
      counter_2 += M2[k];
      M[k] = no_zero_temp;
      delete [] temp;
    }

    A = new int[counter];
    V = new int[counter];
    for (int i = 0; i < counter; i++) {
      A[i] = A_temp[i];
      V[i] = V_temp[i];
    }
    delete [] A_temp;
    delete [] V_temp;
}

bool semetr_razr(int *M, int *A, int *V, int n, int m) {
    //проверка в норм форме -- должно быть a[i][j] = a[j][i]
    int temp_1 = 0;
    for (int num_1 = 0; num_1 < n; num_1++) { //рассматриваем все строки
      for (int i = 0; i < M[num_1]; i++) { //рассматриваем ненулевые элементы строки
        int a_temp_i = A[temp_1 + i];
        int v_temp_i = V[temp_1 + i];
        int flag = 0;
        int temp_2 = 0;
        for (int num_2 = 0; num_2 < n; num_2++) {
          for (int j = 0; j < M[num_2]; j++) {
            int a_temp_j = A[temp_2 + j];
            int v_temp_j = V[temp_2 + j];
            if (num_2 == a_temp_i && a_temp_j == num_1 && v_temp_j == v_temp_i) {
              flag = 1;
            }
          }
          temp_2 += M[num_2];
        }
        if (!flag) {
          return false;
        }

      }
      temp_1 += M[num_1];
    }
    return true;
}

void summ_normal(int **matrix, int **matrix1, int **matrix2, int n, int m) {
    for (int i = 0; i < n; i++){
      for (int j = 0; j < m; j++){
        matrix[i][j] += (matrix1[i][j] + matrix2[i][j]);
      }
    }
}

bool semetr_normal(int **matrix, int n, int m) {
```

```
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            if (matrix[i][j] != matrix[j][i]) {
                return false;
            }
        }
    }
    return true;
}

int main() {
    int **matrix_1, *M_1, *A_1, *V_1;
    int **matrix_2, *M_2, *A_2, *V_2;
    int n, m;
    scanf("%d", &n);
    scanf("%d", &m);
    matrix_1 = new int*[n];
    matrix_2 = new int*[n];

    for (int i = 0; i < n; i++){
        matrix_1[i] = new int[m];
        matrix_2[i] = new int[m];
    }

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            matrix_1[i][j] = 0;
            matrix_2[i][j] = 0;
        }
    }
    M_1 = new int[n];
    M_2 = new int[n];
    int nozero1 = 0;
    int nozero2 = 0;
    for (int i = 0; i < n; i++) {
        scanf("%d", &M_1[i]);
        nozero1 += M_1[i];
    }
    A_1 = new int[nozero1];
    V_1 = new int[nozero1];
    for (int i = 0; i < nozero1; i++) {
        scanf("%d", &A_1[i]);
    }
    for (int i = 0; i < nozero1; i++) {
        scanf("%d", &V_1[i]);
    }

    for (int i = 0; i < n; i++) {
        scanf("%d", &M_2[i]);
        nozero2 += M_2[i];
    }
    A_2 = new int[nozero2];
    V_2 = new int[nozero2];
    for (int i = 0; i < nozero2; i++) {
```

```
   scanf("%d", &A_2[i]);
}

for (int i = 0; i < nozero2; i++) {
   scanf("%d", &V_2[i]);
}

to_normal(matrix_1, M_1, A_1, V_1, n, m);
to_normal(matrix_2, M_2, A_2, V_2, n, m);

printf("RAZR MATRIX_1:\n");
print_razr(M_1, A_1, V_1, n, m);
printf("\n");

printf("RAZR MATRIX_2:\n");
print_razr(M_2, A_2, V_2, n, m);
printf("\n");

printf("FULL ARRAY_1 FROM RAZR_1:\n");
print_normal(matrix_1, n, m);
printf("\n");

printf("FULL ARRAY_2 FROM RAZR_2:\n");
print_normal(matrix_2, n, m);
printf("\n");


int *M, *A, *V; //сюда будет записана сумма
printf("SUMMA RAZR MATRIX AND IN NORMAL FORM: \n");
summ_razr(M, A, V, M_1, A_1, V_1, M_2, A_2, V_2, n, m);
print_razr(M, A, V, n, m);
int **matrix;
matrix = new int*[n];
for (int i = 0; i < n; i++) {
   matrix[i] = new int[m];
}
for (int i = 0; i < n; i++) {
   for (int j = 0; j < m; j++) {
      matrix[i][j] = 0;
   }
}
printf("------------------------------\n");
to_normal(matrix, M, A, V, n, m);
print_normal(matrix, n, m);
if (n == m) {
   if (semetr_razr(M, A, V, n, m)) {
      printf("the matrix is symmetric\n");
   }
   else {
      printf("the matrix is not symmetric\n");
   }
}
else {
   printf("the matrix is not symmetric\n");
```

```
    }

    printf("\n");
    printf("SUMMA NORMAL MATRIX: \n");
    int **matrix_t;
    matrix_t = new int*[n];
    for (int i = 0; i < n; i++) {
        matrix_t[i] = new int[m];
    }
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            matrix_t[i][j] = 0;
        }
    }
    summ_normal(matrix_t, matrix_1, matrix_2, n, m);
    print_normal(matrix_t, n, m);
    int *M_t, *A_t, *V_t;
    to_razr(matrix_t, M_t, A_t, V_t, n, m);
    printf("------------------------------\n");
    print_razr(M_t, A_t, V_t, n, m);
    if (n == m) {
        if (semetr_normal(matrix_t, n, m)) {
            printf("the matrix is symmetric\n");
        }
        else {
            printf("the matrix is not symmetric\n");
        }
    }
    else {
        printf("the matrix is not symmetric\n");
    }
    return 0;
}
```

Тестирование:
1 тест:

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

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

Ответ:

```
1 0 2 3 0
0 0 1 0 3 4
2 4 8 10 12 4
the matrix is not symmetric
```

2 тест:

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

2 0 0 0 1
0 1 0
5 6 1

Ответ:

2 1 1 0 4
0 1 1 1 0 2 3 4
6 6 1 1 1 4 5 6
the matrix is not symmetric

3 тест:

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

1 0 1 0 0 1
0 2 3
-1 7 8

Ответ:

0 1 2 0 1 1
1 1 2 4 3
1 1 7 1 8
the matrix is not symmetric

4 тест:

1 1
1
0
-1
1
0

Ответ:

0

the matrix is symmetric

5 тест:

4 4
3 1 0 0
0 1 2 1
1 -1 1 1

0 1 1 1
0 0 3
-1 1 5

Ответ:
3 2 1 1
0 1 2 0 1 0 3
1 -1 1 -1 1 1 5
the matrix is symmetric

Пункты 1-7 отчета составляются строго до начала лабораторной работы.

Допущен к выполнению работы. **Подпись преподавателя _____**

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

```
[alex@fedora 7(?)]$ cat head.txt
-------------------------------------------------------
|          Лабораторная работа №7                     |
|          Разреженные матрицы                        |
|          Выполнил:  студент группы М8О-101Б-21      |
|          Постнов Александр Вячеславович             |
-------------------------------------------------------
[alex@fedora 7(?)]$ g++ main.cpp -o main
[alex@fedora 7(?)]$ cat test.txt
5 5
1 0 2 3 0
0 0 1 0 3 4
1 2 4 5 6 2

1 0 2 3 0
0 0 1 0 3 4
1 2 4 5 6 2
[alex@fedora 7(?)]$ ./main <test.txt
RAZR MATRIX_1:
1 0 2 3 0
0 0 1 0 3 4
1 2 4 5 6 2
```

```
RAZR MATRIX_2:
1 0 2 3 0
0 0 1 0 3 4
1 2 4 5 6 2

FULL ARRAY_1 FROM RAZR_1:
1 0 0 0 0
0 0 0 0 0
2 4 0 0 0
5 0 0 6 2
0 0 0 0 0

FULL ARRAY_2 FROM RAZR_2:
1 0 0 0 0
0 0 0 0 0
2 4 0 0 0
5 0 0 6 2
0 0 0 0 0

SUMMA RAZR MATRIX AND IN NORMAL FORM:
1 0 2 3 0
0 0 1 0 3 4
2 4 8 10 12 4
------------------------------
2 0 0 0 0
0 0 0 0 0
4 8 0 0 0
10 0 0 12 4
0 0 0 0 0
the matrix is not symmetric

SUMMA NORMAL MATRIX:
2 0 0 0 0
0 0 0 0 0
4 8 0 0 0
10 0 0 12 4
0 0 0 0 0
------------------------------
1 0 2 3 0
0 0 1 0 3 4
2 4 8 10 12 4
the matrix is not symmetric
[alex@fedora 7(?)]$ cat test1.txt
5 5
1 1 1 0 3
0 1 1 2 3 4
1 1 1 4 5 6

2 0 0 0 1
0 1 0
5 6 1
[alex@fedora 7(?)]$ ./main <test1.txt
RAZR MATRIX_1:
```

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

RAZR MATRIX_2:
2 0 0 0 1
0 1 0
5 6 1

FULL ARRAY_1 FROM RAZR_1:
1 0 0 0 0
0 1 0 0 0
0 1 0 0 0
0 0 0 0 0
0 0 4 5 6

FULL ARRAY_2 FROM RAZR_2:
5 6 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
1 0 0 0 0

SUMMA RAZR MATRIX AND IN NORMAL FORM:
2 1 1 0 4
0 1 1 1 2 3 4 0
6 6 1 1 4 5 6 1
---
6 6 0 0 0
0 1 0 0 0
0 1 0 0 0
0 0 0 0 0
1 0 4 5 6
the matrix is not symmetric

SUMMA NORMAL MATRIX:
6 6 0 0 0
0 1 0 0 0
0 1 0 0 0
0 0 0 0 0
1 0 4 5 6
---
2 1 1 0 4
0 1 1 1 0 2 3 4
6 6 1 1 1 4 5 6
the matrix is not symmetric
[alex@fedora 7(?)]$ cat test2.txt
6 5
1 1 1 0 1 0
0 1 1 4
1 1 1 1

1 0 1 0 0 1
0 2 3

```
-1 7 8
[alex@fedora 7(?)]$ ./main <test2.txt
RAZR MATRIX_1:
1 1 1 0 1 0
0 1 1 4
1 1 1 1

RAZR MATRIX_2:
1 0 1 0 0 1
0 2 3
-1 7 8

FULL ARRAY_1 FROM RAZR_1:
1 0 0 0 0
0 1 0 0 0
0 1 0 0 0
0 0 0 0 0
0 0 0 0 1
0 0 0 0 0

FULL ARRAY_2 FROM RAZR_2:
-1 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 8 0

SUMMA RAZR MATRIX AND IN NORMAL FORM:
0 1 2 0 1 1
1 1 2 4 3
1 1 7 1 8
------------------------------
0 0 0 0 0
0 1 0 0 0
0 1 7 0 0
0 0 0 0 0
0 0 0 0 1
0 0 0 8 0
the matrix is not symmetric

SUMMA NORMAL MATRIX:
0 0 0 0 0
0 1 0 0 0
0 1 7 0 0
0 0 0 0 0
0 0 0 0 1
0 0 0 8 0
------------------------------
0 1 2 0 1 1
1 1 2 4 3
1 1 7 1 8
the matrix is not symmetric
[alex@fedora 7(?)]$ cat test3.txt
```

```
1 1
1
0
-1
1
0
1
[alex@fedora 7(?)]$ ./main <test3.txt
RAZR MATRIX_1:
1
0
-1

RAZR MATRIX_2:
1
0
1

FULL ARRAY_1 FROM RAZR_1:
-1

FULL ARRAY_2 FROM RAZR_2:
1

SUMMA RAZR MATRIX AND IN NORMAL FORM:
0


------------------------------
0
the matrix is symmetric

SUMMA NORMAL MATRIX:
0
------------------------------
0


the matrix is symmetric
[alex@fedora 7(?)]$ cat test4.txt
4 4
3 1 0 0
0 1 2 1
1 -1 1 1

0 1 1 1
0 0 3
-1 1 5
[alex@fedora 7(?)]$ ./main <test4.txt
RAZR MATRIX_1:
3 1 0 0
0 1 2 1
1 -1 1 1
```

RAZR MATRIX_2:
0 1 1 1
0 0 3
-1 1 5

FULL ARRAY_1 FROM RAZR_1:
1 -1 1 0
0 1 0 0
0 0 0 0
0 0 0 0

FULL ARRAY_2 FROM RAZR_2:
0 0 0 0
-1 0 0 0
1 0 0 0
0 0 0 5

SUMMA RAZR MATRIX AND IN NORMAL FORM:
3 2 1 1
0 1 2 1 0 0 3
1 -1 1 1 -1 1 5
---
1 -1 1 0
-1 1 0 0
1 0 0 0
0 0 0 5
the matrix is symmetric

SUMMA NORMAL MATRIX:
1 -1 1 0
-1 1 0 0
1 0 0 0
0 0 0 5
---
3 2 1 1
0 1 2 0 1 0 3
1 -1 1 -1 1 1 5
the matrix is symmetric

9. Дневник отладки должен содержать дату и время сеансов отладки и основные события (ошибки в сценарии и программе, нестандартные ситуации) 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
1	дом	20.04.2	9:00	Сравнивал матрицу с числом	Сравнивал элемент матрицы с числом	Это произошло из-за похожего названия переменных(V_temp,

		022				v_temp)

10. Замечания автора

11.**Выводы**

В ходе лабораторной работы изучил способы задавания разреженных матриц, они бывают очень полезные, если матрицы большого размера с большим кол-вом нулевых элементов Это позволяет значительно экономить память и производить меньше действий!

Недочёты при выполнении задания могут быть устранены следующим образом:

Подпись студента ____Постнов_____