Отчет по курсовой работе VII

по курсу: 1 фундаментальная информатика

студента группы М80-101Б-21 Тулина Ивана, № по списку: 22

Контакты www, e-mail, icq, skype: i.tulin0107@gmail.com

Работа выполнена: «15» июня 2022г.

Преподаватель: Титов В. К. каф. 806

Входной контроль знаний с оценкой _____

Отчет сдан « » _____201 __ г., итоговая оценка _____

Подпись преподавателя _____

1. **Тема:** Разреженные матрицы

2. **Цель работы:** Составить программу на языке Си с процедурами и функциями для обработки прямоугольных разреженных матриц с элементами вещественного типа.

3. **Задание** (*вариант №* 2, 11)**:** *Вариант размещения матрицы:* ненулевому элементу соответствуют две ячейки: первая содержит номер столбца, вторая содержит значение элемента. Нуль в первой ячейке означает конец строки, а вторая ячейка содержит в этом случае номер следующей хранимой строки. Нули в обеих ячейках являются признаком конца перечня ненулевых элементов разреженной матрицы. *Вариант преобразования:* транспонировать разреженную матрицу относительно побочной диагонали. Выяснить, является ли полученная матрица кососимметрической.

4. **Оборудование** (лабораторное):
ЭВМ_____-_____, процессор_____-_____, имя узла сети _____-_____ с ОП _____-_____ Мб, НМД _____ Мб. Терминал _____ адрес _____. Принтер _____
Другие устройства _____

Оборудование ПЭВМ студента, если использовалось:
Процессор Intel Core i5-7300HQ с ОП 7,87 Мб, НМД 15360 Мб. Монитор: встроенный
Другие устройства _____

5. **Программное обеспечение (лабораторное):**
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 UNIX, наименование Ubuntu версия 20.04.3 LTS
интерпретатор команд bash версия _____.
Система программирования _____версия _____
Редактор текстов Emacs версия 3.22.30
Утилиты операционной системы -
Прикладные системы и программы -
Местонахождение и имена файлов программ и данных на домашнем компьютере -

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

Согласно тексту задания программа должна получить на вход матрицу в обычном (полном) формате, сохранить её в сокращённом виде, напечатать в полной и сокращенной вариации, провести транспонирование относительно побочной диагонали и проверить на кососимметрию, а затем вывести полученный результат.

Идея:

Первым делом проверим имя входного файла (по умолчанию входные данные будут считаны из файла «in.txt»). Затем до ввода матрицы динамически выделим память под одномерный вещественный массив **a** в котором будет храниться сокращённая форма этой матрицы, а также под двумерные вещественные массивы **mtr** и **mtrT**, в которых будут находиться полные её формы (перед транспонированием и после транспонирования, соответственно).
Первоначально на вход поступают только размеры полной матрицы **n** и **m**, а также количество ненулевых элементов **k**. Это не позволяет точно задать размеры вектора **a**. Точные размеры этого вектора должны учитывать количество строк, на которых присутствуют ненулевые элементы матрицы, так как они тоже записываются в него. Таким образом, для массива **a**, куда мы будем записывать матрицу, придется выделить объём памяти с запасом, чтобы учесть случай, когда на каждой строке матрицы будет присутствовать хотя бы один ненулевой элемент. Учитывая, что рядом с каждым ненулевым элементом должен стоять номер его столбца, а рядом с номером непустой строки должен стоять ноль, формулой для вычисления длины вектора **a** является формула ***2(k+n)***.
Массивы **mtr** и **mtrT** планируются двумерными. Для их реализации используем массивы указателей: для двумерного массива **mtr** создадим одномерный массив длины **n** инициализированный указателями на одномерные массивы длины **m**, а для двумерного массива **mtrT** – массив длины **m** инициализированный указателями на одномерные массивы длины **n**.

Ввод матрицы:
Считывание матрицы должно происходить во вложенном цикле **for**. Внешний цикл эмитирует чтение строки, внутренний – чтение столбца (считывание элементов будет происходить во внутреннем цикле). За записью элементов в массив **a** будет следить заранее созданный и инициализированный нулем счетчик **cnt**.
Каждую итерацию внутреннего цикла вместе со считыванием происходит проверка элемента. Если элемент ненулевой, необходимо сначала записать в массив номер строки этого элемента и только потом помещать туда этот элемент и последующие за ним ненулевые элементы. Поэтому номер строки (перед номером строки, естественно, должен быть записан ноль) будет записываться в начале каждой итерации внешнего цикла. Во внутреннем цикле с каждым найденным ненулевым элементом значение **cnt** будет увеличиваться на 2, а также происходить запись столбца и самого элемента матрицы по индексам ***cnt*** и ***cnt+1***. Если ненулевых элементов не будет найдено, то счётчик не изменится, и на место номера предыдущей строки будет записан номер новой строки.
Таким образом полученная на вход разреженная полная матрица запишется в одномерный вектор.

Теперь, имея введенный вектор **a**, мы можем точно посчитать длину вектора **aT**, где будет храниться сокращённая форма транспонированной матрицы, выделив необходимый и достаточный для этого объём памяти. Создадим функцию, которая считает количество ненулевых столбцов во введенной матрице (при транспонировании матрицы число ненулевых столбцов превратится в число ненулевых строк). Для этого внутри функции придется динамически выделить инициализированный нулями локальный массив **flags** длины **m**, каждая ячейка которого будет отвечать за отдельный столбец матрицы. Функция будет пробегать вектор **a** (в функцию будет передана ссылка на него) и отмечать в созданном локальном массиве столбцы, которые были ей встречены, увеличивая счетчик ненулевых столбцов. Значение этой функции (окончательное значение счетчика) уже в основной программе запишем в новую переменную, которую назовем **t**. Таким образом длина вектора **aT,** будет вычисляться по формуле ***2(k+t)***.

Печать матрицы:
Для вывода полной и сокращённой формы матрицы выделим две отдельные процедуры. Сам вывод будет производиться в циклах. Вследствие того, что мы не знаем точной длины массива **a**, вывод сокращённой формы матрицы невозможно реализовать посредством цикла по счётчику. Но мы имеем возможность воспользоваться особенностью построения самой сокращённой формы: два идущих подряд нуля являются признаком окончания массива (номера строк, столбцов матрицы и сами элементы всегда отличны от нуля). Поэтому вывод сокращенной формы матрицы будет производиться в цикле с постусловием.
Полную форму матрицы будем выводить во вложенном цикле **for**. Внешний цикл эмитирует печать строки, внутренний – печать столбца.

Трансформация сокращённой формы матрицы в полную:
Данная трансформация также будет происходить в отдельной процедуре. Фактическими параметрами для процедуры должны стать ссылка на массив с сокращённой формой матрицы, ссылка на массив, куда будет помещена полная её форма, и размеры полной матрицы. Процедура состоит из двух циклов. Первый цикл записывает во все ячейки пустого

массива, отведённого под полную форму матрицы, нули. Второй цикл должен переписывать в этот массив ненулевые элементы из сокращённой формы. В первом случае можно использовать вложенные циклы **for**, так как нам известны размеры полной формы матрицы. Во втором случае необходимо реализовать чтение сокращённой формы матрицы, поэтому придется пользоваться циклом с предусловием вложенным в такой же цикл с предусловием. Внешний цикл будет работать пока не встретит два нуля, идущие подряд. Внутренний цикл будет работать пока не встретит один ноль. Во внешнем цикле будет считываться номер строки и сохраняться в отдельную переменную **s**. Во внутреннем цикле будут считываться номер столбца и непосредственно само значение элемента, которое будет записываться по считанным координатам в полную форму матрицы. За чтением сокращенной формы будет следить заранее инициализированный нулем счётчик **i**, который будет увеличиваться на 2 при каждой итерации внешнего и внутреннего цикла.

Трансформация полной формы матрицы в сокращённую:
Процедура по сути повторяет ввод полной формы матрицы и сохранение её в сокращённом виде. Во внешнем цикле **for** записывается номер читаемой строки полной формы матрицы, во внутреннем происходит проверка элемента полной формы (отличен ли он от нуля). Если элемент ненулевой, то счетчик увеличивается и элемент с номером столбца записываются на свои места. Если ненулевых элементов найдено не было, то на место номера предыдущей строки массива записывается номер следующей.

Транспонирование полной формы матрицы относительно побочной диагонали:
Обобщённо говоря, процесс транспонирования матрицы относительно побочной диагонали состоит из замены строк и столбцов матрицы зеркальными их копиями, а затем замены строк и столбцов местами.
Поэтому процедура состоит из вложенного цикла **for**, который пробегает двумерный массив **mtr** и записывает каждый **x**, **y**-ый элемент этого массива на (***m-y)***, (***n-x)*** -ое место массива **mtrT**.

Транспонирование сокращённой формы матрицы относительно побочной диагонали:
При транспонировании сокращённой формы матрицы мы не знаем точные индексы элементов, которые надо поменять местами. Поэтому важно соблюдать порядок обхода вектора(надо идти от последнего столбца к первому и от последней строки к первой), чтобы не нарушать последовательность строк транспонированной матрицы и не перемешивать столбцы.
Первым делом надо найти наибольший ненулевой столбец матрицы, который после транспонирования станет самой первой ненулевой строкой транспонированной матрицы. Затем вектор транспонируемой сокращённой формы надо пробежать с конца в поиске строк, на пересечении с которыми у этого столбца есть ненулевые элементы. Эти строки станут столбцами первой строки транспонированной матрицы. Таким образом, мы заполняем элементами первую строку нового вектора. После заполнения строки найденный последний столбец исходного вектора необходимо сохранить в качестве верхней границы поиска новых векторов, которые займут место последующих строк в новой матрице.
Описанный алгоритм транспонирования столбцов необходимо повторять пока верхняя граница поиска не примет нулевое значение, что будет означать, что все столбцы транспонированы и полученный вектор является сокращенной формой матрицы транспонированной относительно побочной диагонали.

Таким образом процедура будет состоять из цикла **while**, внутри которого будет реализованы

1)Поиск наибольшего столбца под верхней границей (заранее инициализируем её значением MAX_INT)

2)Занесение номера столбца в качестве верхней границы в отдельную переменную

3)Проверка, не равна ли нулю верхняя граница (выход из внешнего цикла будет происходить именно здесь)

4)Обратное считывание транспонируемого вектора в поиске пересекающихся с найденным столбцом строк и, собственно, запись их в транспонированный вектор.

Проверка кососимметричности матрицы
Квадратная матрица A называется кососимметрической, если $A=-A^T$.
Первым делом нужно проверить равно ли количество строк **n** количеству столбцов **m** и четны ли эти числа.

В случае полной формы матрицы далее во вложенном цикле **for** необходимо проверить элементы матрицы с одинаковыми индексами, используя переменную **flag** инициализированную нулём. Если какой-то элемент **mtr** не равен элементу **mtrT**, взятому с противоположным знаком, то значение **flag** должно замениться на 1.

Если форма матрицы сокращённая, то проверку элементов придется реализовывать через цикл **while**. Внешний цикл проверяет равенство номеров строк, увеличивая заранее инициализированный нулем счетчик, и останавливается в случае, если встретит два идущих подряд нуля в одном из векторов. Внутренний цикл проверяет равенство номеров столбцов, сверяет элементы матриц, увеличивает счетчик и останавливается, если встретит один ноль в одном из векторов. Если номера строк, столбцов не совпадают или элемент вектора **a** не равен элементу **aT,** взятому с противоположным знаком, то значение переменной **flag** (заранее инициализированной нулем) изменится на 1.

В конце процедуры в зависимости от значения переменной **flag** выводится ответ (матрица кососимметрическая или НЕ кососимметрическая)

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

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

void prn_sh_mtr(float*);
void sh_to_full(float*, float**, int, int);
void prn_mtr(float**, int, int);
void full_to_sh(float**, int, int, float*);
void tr_full_mtr(float**, float**, int, int);
int col_counter(float*, int);
void tr_sh_mtr(float*, float*, int, int);
void kososimmetriya_full_mtr(float**, float**, int, int);
void kososimmetriya_sh_mtr(float*, float*, int, int);


int main(int argc, char *argv[]){
FILE *fi;
int m, n, k, t;
float **mtr, **mtrT, *a, *aT;

if(argc==1)
  {if(!(fi=fopen("in.txt", "r")))
    {printf("Can't input file\n"); return 0;}}
else
  {if(!(fi=fopen(argv[1], "r")))
    {printf("Can't open %s\n", argv[1]); return 0;}}

fscanf(fi, "%d %d %d", &n, &m, &k);

a = new float[(k+n)*2];

mtr= new float*[n];
for(int i=0; i<n; i++)
    mtr[i] = new float[m];

mtrT= new float*[m];
for(int i=0; i<m; i++)
    mtrT[i] = new float[n];


int cnt=0;
for(int i=0; i<n; i++)
   {float elem;
    a[cnt]=0; a[cnt+1]=i+1;
    for(int j=0; j<m; j++)
       {fscanf(fi, "%f", &elem);
        if(elem)
           {cnt+=2;
            a[cnt]=j+1;
            a[cnt+1]=elem;
           }
       }
    if(a[cnt]) cnt+=2;
}
a[cnt]=0;
a[cnt+1]=0;

t = col_counter(a, m);
aT = new float[(k+t)*2];

prn_sh_mtr(a);
sh_to_full(a, mtr, n, m);
prn_mtr(mtr, n, m);
full_to_sh(mtr, n, m, a);
prn_sh_mtr(a);

tr_full_mtr(mtr, mtrT, n, m);
prn_mtr(mtrT, m, n);
tr_sh_mtr(a, aT, n, m);
prn_sh_mtr(aT);
kososimmetriya_full_mtr(mtr, mtrT, n, m);
kososimmetriya_sh_mtr(a, aT, n, m);
return 0;}
```

```
void prn_sh_mtr(float *a){
   printf("\nСокращенная форма матрицы:\n");
   int i=-2;
   do
      {i+=2;
       if(!a[i])
          printf("  %1.0f %1.0f  ", a[i], a[i+1]);
       else
          printf("%1.0f %1.2f  ", a[i], a[i+1]);
      }
   while(a[i]||a[i+1]);
   printf("\n");
}

void sh_to_full(float *a, float **mtr, int n, int m){
   printf("\nТрансформация сокращённой формы матрицы в полную...\n");
   for(int i=0; i<n; i++)
      for(int j=0; j<m; j++)
            mtr[i][j]=0;
   int s,i=0;
   while(a[i]||a[i+1])
      {s=(int)a[i+1]-1; i+=2;
          while(a[i])
            {mtr[s][(int)a[i]-1]=a[i+1]; i+=2;
            }
      }
}

void prn_mtr(float **mtr, int n, int m){
   printf("\nПолная форма матрицы:\n");
      for(int i=0; i<n; i++)
        {for(int j=0; j<m; j++)
            printf("%6.2f", mtr[i][j]);
         printf("\n");
        }
}

void full_to_sh(float **mtr, int n, int m, float *a){
   printf("\nТрансформация полной формы матрицы в сокращённую...\n");
   int i=0;
   for(int x=0; x<n; x++)
      {a[i]=0; a[i+1]=x+1;
       for(int y=0; y<m; y++)
          if(mtr[x][y])
            {i+=2;
             a[i]=y+1;
             a[i+1]=mtr[x][y];
            }
       if(a[i]) i+=2;
      }
   a[i]=0;
   a[i+1]=0;
}

void tr_full_mtr(float **mtr, float **mtrT, int n, int m){
   printf("\nТранспонирование полной формы матрицы...\n");
   for(int y=0; y<m; y++)
      for(int x=0; x<n; x++)
         mtrT[y][x]=mtr[n-x-1][m-y-1];
}

int col_counter(float *a, int m){
   int *flags; flags=new int[m];
   int counter=0;
   for(int i=0; i<m; i++)
      flags[i]=0;
   int j=0;
   do
      {if(!flags[(int)a[j]-1]&&a[j])
         {flags[(int)a[j]-1]=1;
          counter++;
         }
       j+=2;
      }
   while(a[j]||a[j+1]);
```

```
    free(flags);
    return counter;
}

void tr_sh_mtr(float *a, float *aT, int n, int m){
    printf("\nТранспонирование сокращённой формы матрицы...\n");
    int cnt=0, up_b=INT_MAX;
    while(up_b)
      {int down_b=-1, ind, l=0;
       do
         {if(down_b<a[l]&&a[l]<up_b)
            {down_b=a[l]; ind=l;
            }
          l+=2;
         }
       while(a[l]||a[l+1]);
       down_b=(int)down_b;
       up_b=down_b;
       if(!up_b) break;
       aT[cnt]=0;  cnt++;
       aT[cnt]=m-down_b+1;
       int i=l-2;
       while(i>=ind)
         {if((int)a[i]==down_b)
            {aT[cnt+2]=a[i+1];
             while(a[i])
                i-=2;
             cnt++;
             aT[cnt]=n-a[i+1]+1;
             cnt++;
            }
          else
             i-=2;
         }
       cnt++;
      }
    aT[cnt]=0;
    aT[cnt+1]=0;
}

void kososimmetriya_full_mtr(float **mtr, float **mtrT, int n, int m){
    printf("\nПроверка полной формы матрицы...\n");
    int flag=0;
    if(m!=n||(n%2!=0))
       flag=1;
    else
    for(int i=0; i<n; i++)
      {for(int j=0; j<m; j++)
          if(mtr[i][j]!=(-mtrT[i][j]))
              {flag=1; break;}
       if(flag) break;
      }

    if(flag)
       printf(">>Матрица НЕ кососимметрическая<<\n");
    else
       printf(">>Матрица кососимметрическая<<\n");
}

void kososimmetriya_sh_mtr(float *a, float *aT, int n, int m){
    printf("\nПроверка сокращённой формы матрицы...\n");
    int flag=0;
    if(m!=n||(n%2!=0))
       flag=1;
    else
   {int cnt=0;
    while(a[cnt]||a[cnt+1])
      {if(a[cnt+1]==(aT[cnt+1]))
         {cnt+=2;
          while(a[cnt])
             if(a[cnt]==aT[cnt]&&a[cnt+1]==(-aT[cnt+1]))
                cnt+=2;
             else
               {flag=1;
                break;}
```

```
          }
        else
          {flag=1;
           break;
          }
      }
  }
   if(flag)
      printf(">>Матрица НЕ кососимметрическая<<\n");
   else
      printf(">>Матрица кососимметрическая<<\n");
}
```

Тест 1
5 4 6
0.0 2.3 1.5 6.4
0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0
5.0 7.2 0.0 0.1

Тест 2
4 4 2
0.0 0.0 0.0 0.0
0.0 -2.8 0.0 0.0
0.0 0.0 2.8 0.0
0.0 0.0 0.0 0.0

Тест 3
3 3 3
4.1 0.0 0.0
0.0 8.4 0.0
0.0 0.0 2.3

Тест 4
2 3 0
0 0 0
0 0 0

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

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

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

```
yusayu@YS:~/Рабочий стол/cppProjects$ cat head
*****************************************************************
*                  Курсовая работа VII                  *
*                  Рвзреженные матрицы                    *
*              Выполнил: Тулин Иван Денисович                  *
*                  (номер по списку: 22)                *
*                   Группа: М8О-101Б-21                   *
*****************************************************************
yusayu@YS:~/Рабочий стол/cppProjects$ cat kr7.cpp
#include<stdio.h>
#include<limits.h>
#include<stdlib.h>

void prn_sh_mtr(float*);
void sh_to_full(float*, float**, int, int);
void prn_mtr(float**, int, int);
void full_to_sh(float**, int, int, float*);
void tr_full_mtr(float**, float**, int, int);
int col_counter(float*, int);
void tr_sh_mtr(float*, float*, int, int);
void kososimmetriya_full_mtr(float**, float**, int, int);
void kososimmetriya_sh_mtr(float*, float*, int, int);


int main(int argc, char *argv[]){
FILE *fi;
int m, n, k, t;
float **mtr, **mtrT, *a, *aT;

if(argc==1)
  {if(!(fi=fopen("in.txt", "r")))
    {printf("Can't input file\n"); return 0;}}
else
  {if(!(fi=fopen(argv[1], "r")))
    {printf("Can't open %s\n", argv[1]); return 0;}}

fscanf(fi, "%d %d %d", &n, &m, &k);

a = new float[(k+n)*2];

mtr= new float*[n];
for(int i=0; i<n; i++)
    mtr[i] = new float[m];

mtrT= new float*[m];
for(int i=0; i<m; i++)
    mtrT[i] = new float[n];


int cnt=0;
for(int i=0; i<n; i++)
   {float elem;
    a[cnt]=0; a[cnt+1]=i+1;
    for(int j=0; j<m; j++)
       {fscanf(fi, "%f", &elem);
        if(elem)
           {cnt+=2;
            a[cnt]=j+1;
            a[cnt+1]=elem;
```

```
            }
        }
    if(a[cnt]) cnt+=2;
}
a[cnt]=0;
a[cnt+1]=0;

t = col_counter(a, m);
aT = new float[(k+t)*2];

prn_sh_mtr(a);
sh_to_full(a, mtr, n, m);
prn_mtr(mtr, n, m);

full_to_sh(mtr, n, m, a);
prn_sh_mtr(a);

tr_full_mtr(mtr, mtrT, n, m);
prn_mtr(mtrT, m, n);

tr_sh_mtr(a, aT, n, m);
prn_sh_mtr(aT);

kososimmetriya_full_mtr(mtr, mtrT, n, m);
kososimmetriya_sh_mtr(a, aT, n, m);
return 0;}


void prn_sh_mtr(float *a){
    printf("\nСокращенная форма матрицы:\n");
    int i=-2;
    do
      {i+=2;
       if(!a[i])
          printf("  %1.0f %1.0f  ", a[i], a[i+1]);
       else
          printf("%1.0f %1.2f  ", a[i], a[i+1]);
      }
    while(a[i]||a[i+1]);
    printf("\n");
}


void sh_to_full(float *a, float **mtr, int n, int m){
    printf("\nТрансформация сокращённой формы матрицы в полную...\n");
    for(int i=0; i<n; i++)
        for(int j=0; j<m; j++)
              mtr[i][j]=0;
    int s,i=0;
    while(a[i]||a[i+1])
      {s=(int)a[i+1]-1; i+=2;
          while(a[i])
            {mtr[s][(int)a[i]-1]=a[i+1];
             i+=2;
            }
      }
}

void prn_mtr(float **mtr, int n, int m){
    printf("\nПолная форма матрицы:\n");
       for(int i=0; i<n; i++)
```

```
          {for(int j=0; j<m; j++)
              printf("%6.2f", mtr[i][j]);
           printf("\n");
          }
}

void full_to_sh(float **mtr, int n, int m, float *a){
   printf("\nТрансформация полной формы матрицы в сокращённую...\n");
   int i=0;
   for(int x=0; x<n; x++)
      {a[i]=0; a[i+1]=x+1;
       for(int y=0; y<m; y++)
          if(mtr[x][y])
             {i+=2;
              a[i]=y+1;
              a[i+1]=mtr[x][y];
             }
        if(a[i])
          {i+=2;
          }
      }
   a[i]=0;
   a[i+1]=0;
}


void tr_full_mtr(float **mtr, float **mtrT, int n, int m){
   printf("\nТранспонирование полной формы матрицы...\n");
   for(int y=0; y<m; y++)
       for(int x=0; x<n; x++)
          mtrT[y][x]=mtr[n-x-1][m-y-1];
}


int col_counter(float *a, int m){
   int *flags; flags=new int[m];
   int counter=0;
   for(int i=0; i<m; i++)
       flags[i]=0;
   int j=0;
   do
      {if(!flags[(int)a[j]-1]&&a[j])
          {flags[(int)a[j]-1]=1;
           counter++;
          }
       j+=2;
      }
   while(a[j]||a[j+1]);
   free(flags);
   return counter;
}


void tr_sh_mtr(float *a, float *aT, int n, int m){
   printf("\nТранспонирование сокращённой формы матрицы...\n");
   int cnt=0, up_b=INT_MAX;
   while(up_b)
      {int down_b=-1, ind, l=0;
       do
          {if(down_b<a[l]&&a[l]<up_b)
              {down_b=a[l]; ind=l;
              }
```

```
            l+=2;
           }
        while(a[l]||a[l+1]);
        down_b=(int)down_b;
        up_b=down_b;
        if(!up_b) break;
        aT[cnt]=0;  cnt++;
        aT[cnt]=m-down_b+1;
        int i=l-2;
        while(i>=ind)
          {if((int)a[i]==down_b)
             {aT[cnt+2]=a[i+1];
              while(a[i])
                 i-=2;
              cnt++;
              aT[cnt]=n-a[i+1]+1;
              cnt++;
             }
           else
              i-=2;
          }
        cnt++;
       }
    aT[cnt]=0;
    aT[cnt+1]=0;
}


void kososimmetriya_full_mtr(float **mtr, float **mtrT, int n, int m){
    printf("\nПроверка полной формы матрицы...\n");
    int flag=0;
    if(m!=n||(n%2!=0))
        flag=1;
    else
    for(int i=0; i<n; i++)
       {for(int j=0; j<m; j++)
           if(mtr[i][j]!=(-mtrT[i][j]))
              {flag=1; break;}
        if(flag) break;
       }

    if(flag)
        printf(">>Матрица НЕ кососимметрическая<<\n");
    else
        printf(">>Матрица кососимметрическая<<\n");
}


void kososimmetriya_sh_mtr(float *a, float *aT, int n, int m){
    printf("\nПроверка сокращённой формы матрицы...\n");
    int flag=0;
    if(m!=n||(n%2!=0))
        flag=1;
    else
   {int cnt=0;
    while(a[cnt]||a[cnt+1])
       {if(a[cnt+1]==(aT[cnt+1]))
          {cnt+=2;
           while(a[cnt])
              if(a[cnt]==aT[cnt]&&a[cnt+1]==(-aT[cnt+1]))
                 cnt+=2;
              else
```

```
                {flag=1;
                 break;
                }
          }
        else
          {flag=1;
           break;
          }
       }
   }
    if(flag)
        printf(">>Матрица НЕ кососимметрическая<<\n");
    else
        printf(">>Матрица кососимметрическая<<\n");
}


yusayu@YS:~/Рабочий стол/cppProjects$ cat in.txt
5 4 6
0.0  2.3  1.5  6.4
0.0  0.0  0.0  0.0
0.0  0.0  0.0  0.0
0.0  0.0  0.0  0.0
5.0  7.2  0.0  0.1
yusayu@YS:~/Рабочий стол/cppProjects$ cat in1.txt
4 4 2
0.0  0.0  0.0  0.0
0.0 -2.8  0.0  0.0
0.0  0.0  2.8  0.0
0.0  0.0  0.0  0.0
yusayu@YS:~/Рабочий стол/cppProjects$ cat in2.txt
3 3 3
4.1  0.0  0.0
0.0  8.4  0.0
0.0  0.0  2.3
yusayu@YS:~/Рабочий стол/cppProjects$ cat in3.txt
2 3 0
0 0 0
0 0 0
yusayu@YS:~/Рабочий стол/cppProjects$ c++ kr7.cpp
yusayu@YS:~/Рабочий стол/cppProjects$ ./a.out

Сокращенная форма матрицы:
  0 1  2 2.30  3 1.50  4 6.40    0 5  1 5.00  2 7.20  4 0.10    0 0

Трансформация сокращённой формы матрицы в полную...

Полная форма матрицы:
  0.00  2.30  1.50  6.40
  0.00  0.00  0.00  0.00
  0.00  0.00  0.00  0.00
  0.00  0.00  0.00  0.00
  5.00  7.20  0.00  0.10

Трансформация полной формы матрицы в сокращённую...

Сокращенная форма матрицы:
  0 1  2 2.30  3 1.50  4 6.40    0 5  1 5.00  2 7.20  4 0.10    0 0

Транспонирование полной формы матрицы...

Полная форма матрицы:
```

```
  0.10  0.00  0.00  0.00  6.40
  0.00  0.00  0.00  0.00  1.50
  7.20  0.00  0.00  0.00  2.30
  5.00  0.00  0.00  0.00  0.00

Транспонирование сокращённой формы матрицы...

Сокращенная форма матрицы:
  0 1  1 0.10  5 6.40    0 2  5 1.50    0 3  1 7.20  5 2.30    0 4  1 5.00    0 0

Проверка полной формы матрицы...
>>Матрица НЕ кососимметрическая<<

Проверка сокращённой формы матрицы...
>>Матрица НЕ кососимметрическая<<
yusayu@YS:~/Рабочий стол/cppProjects$ ./a.out in1.txt

Сокращенная форма матрицы:
  0 2  2 -2.80    0 3  3 2.80    0 0

Трансформация сокращённой формы матрицы в полную...

Полная форма матрицы:
  0.00  0.00  0.00  0.00
  0.00 -2.80  0.00  0.00
  0.00  0.00  2.80  0.00
  0.00  0.00  0.00  0.00

Трансформация полной формы матрицы в сокращённую...

Сокращенная форма матрицы:
  0 2  2 -2.80    0 3  3 2.80    0 0

Транспонирование полной формы матрицы...

Полная форма матрицы:
  0.00  0.00  0.00  0.00
  0.00  2.80  0.00  0.00
  0.00  0.00 -2.80  0.00
  0.00  0.00  0.00  0.00

Транспонирование сокращённой формы матрицы...

Сокращенная форма матрицы:
  0 2  2 2.80    0 3  3 -2.80    0 0

Проверка полной формы матрицы...
>>Матрица кососимметрическая<<

Проверка сокращённой формы матрицы...
>>Матрица кососимметрическая<<
yusayu@YS:~/Рабочий стол/cppProjects$ ./a.out in2.txt

Сокращенная форма матрицы:
  0 1  1 4.10    0 2  2 8.40    0 3  3 2.30    0 0

Трансформация сокращённой формы матрицы в полную...

Полная форма матрицы:
  4.10  0.00  0.00
  0.00  8.40  0.00
```

```
  0.00  0.00  2.30

Трансформация полной формы матрицы в сокращённую...

Сокращенная форма матрицы:
  0 1  1 4.10    0 2  2 8.40    0 3  3 2.30    0 0

Транспонирование полной формы матрицы...

Полная форма матрицы:
  2.30  0.00  0.00
  0.00  8.40  0.00
  0.00  0.00  4.10

Транспонирование сокращённой формы матрицы...

Сокращенная форма матрицы:
  0 1  1 2.30    0 2  2 8.40    0 3  3 4.10    0 0

Проверка полной формы матрицы...
>>Матрица НЕ кососимметрическая<<

Проверка сокращённой формы матрицы...
>>Матрица НЕ кососимметрическая<<
yusayu@YS:~/Рабочий стол/cppProjects$ ./a.out in3.txt

Сокращенная форма матрицы:
  0 0

Трансформация сокращённой формы матрицы в полную...

Полная форма матрицы:
  0.00  0.00  0.00
  0.00  0.00  0.00

Трансформация полной формы матрицы в сокращённую...

Сокращенная форма матрицы:
  0 0

Транспонирование полной формы матрицы...

Полная форма матрицы:
  0.00  0.00
  0.00  0.00
  0.00  0.00

Транспонирование сокращённой формы матрицы...

Сокращенная форма матрицы:
  0 0

Проверка полной формы матрицы...
>>Матрица НЕ кососимметрическая<<

Проверка сокращённой формы матрицы...
>>Матрица НЕ кососимметрическая<<
```

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

10. **Замечания автора** по существу работы _____

11. **Выводы**

В ходе работы я научился составлять алгоритмы, оперирующие с разреженными матрицами и их сокращёнными формами

Недочёты при выполнении задания могут быть устранены следующим образом: _____

Подпись студента _____