МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

КУРСОВАЯ РАБОТА

по дисциплине «Программирование»

ТЕМА: Обработка матриц в файле.

Студент гр. 6304 _____ Иванов В.С.

Преподаватель _____ Берленко Т.А.

Санкт-Петербург
2017

ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ

Студент: Иванов В.С.

Группа 6304

Тема работы: Обработка матриц в файле

Содержание пояснительной записки:

- Аннотация
- Содержание
- Введение (цель работы, формулировка задачи)
- Описания функций, использованных в проекте
- Тестирование работоспособности программы
- Разбиение на файлы и работа с Makefile
- Заключение
- Список использованных источников
- Приложение

Предполагаемый объем пояснительной записки:
Не менее 20 страниц.

Дата выдачи задания: _____

Дата сдачи реферата: _____

Дата защиты реферата: _____

Студент _____ Иванов В.С.

Преподаватель _____ Берленко Т.А.

Аннотация

В данной работе описывается процесс создания программы для обработки квадратной матрицы (определение её размеров, приведение к ступенчатому виду, нахождение её ранга), полученной из текстового файла, и записи её ранга в новый текстовый файл. В документе приведён полный исходный код, разбитый на несколько частей, проекта на языке Си, а также описание тестирования работоспособности программы.

Содержание

Введение

Требуется написать программу, на вход которой подается квадратная матрица. Программа получает параметры из входного потока (проверять на корректность не требуется). Программа должна находить ранг матрицы и выводить результат на консоль. Также программа должна проверять, что матрица является квадратной, и в ином случае выводить сообщение об ошибке и не создавать выходного файла.

Цель данной работы заключается в изучении работ с файлами и с матрицами.

Исходя из цели, можно составить перечень задач, которые необходимо решить:

1. Изучение базовых функций работы с файлами;
2. Выбор метода нахождения ранга для реализации заданной программы;
3. Разбиение проекта на несколько файлов;
4. Тестирование работоспособности.

Содержание

1. Создание отдельных элементов проекта

1.1. Реализация функций перестановки

Для написания программы потребуются три функции перестановок.

Первая – функция перестановки значений элементов:

```
void swap(float* prevEl, float* newEl){
   float c=*prevEl;
   *prevEl=*newEl;
   *newEl=c;
}
```

Вторая – функция, меняющая значения элементов одной строки с соответствующими элементами другой строки:

```
void swapstr(float* mass[], int prevStr, int newStr, int col){
   int i;
```

Выполняется обмен значений элементов строки, используя предыдущую функцию.

```
   for(i=0;i<col;i++){
      swap(&mass[prevStr][i],&mass[newStr][i]);
   }
}
```

Третья – функция, аналогичная предыдущей, но совершающая несколько проверок:

```
int FinalSwap(float** mass,int i,int newi, int str){
 int shift=0,j;
```

Обмен значений элементов строк выполняется, если выбранный элемент равен нулю.

```
 if(newi<str && mass[i][newi]==0 ){
  for(j=i+1;j<str;j++)
   if(mass[j][newi]!=0){
    swapstr(mass,i,j,str);
   }
 }
```

В случае, если перестановка строк не помогла избавиться от нуля, а это возможно, если все элементы ниже выбранного, тоже нулевые, то увеличивается значение сдвига и рекурсивно запускается эта же функция, но с соседним (слева от выбранного) элементом.

```
 if(mass[i][newi]==0){
  shift++;
  return shift+FinalSwap(mass,i,newi+1,str);
 }
```

По окончанию работы функция возвращает величину сдвига до ближайшего ненулевого элемента слева либо число, сумма которого с выбранной изначально строкой будет больше числа столбцов.

```
 return shift;
}
```

1.2. Нахождение ранга матрицы

Для нахождения ранга матрицы используется прямой ход метода Гаусса с небольшим изменением (вычислительные операции производятся, начиная с последнего элемента строки).

Также используются две дополнительные функции:

Первая – запись элементов из строки, содержащую матрицу, в двумерный динамический массив для вычислительных действий с её элементами:

```
void creation_matrix(char* s, float** mass,int str){
   int i=0,j=0;
```

С помощью функции стандартной библиотеки *strtok* происходит разбиение строки (при встречи пробелов и символов перевода строки) на лексемы, преобразования их к типу *int* и запись в двумерный массив на соответствующую позицию.

```
   char *k=strtok(s," \n");
   while(k){
      if(j==str){
         i++;
         j=0;
      }
      mass[i][j]=atoi(k);
      j++;
      k=strtok(NULL," \n");
   }

}
```

Вторая – подсчёт ранга матрицы, приведённой методом Гаусса к ступенчатому виду:

```
int find_rang(int str, int *rank, float **mass){
   int i,j;
```

Происходит проверка каждого элемента строки матрицы ступенчатого вида, если он является единицей (т.к. это ступенчатая матрица), то счётчик ранга матрицы увеличивается и начинается проверка следующей строки.

```
   for(i=0;i<str;i++)
      for(j=0;j<str;j++)
         if (mass[i][j]==1){
            (*rank)++;
            j=str;
      }
}
```

Реализация функции, выполняющей приведение матрицы к ступенчатому виду и возвращающей значение его ранга:

```
int calc_rank(char *s, int str, int col){
```

Выполняется проверка, если матрица не квадратная, то возвращается «-1», которая даст понять, что матрица не является квадратной.

```
   if(str!=col)
      return -1;

   int i=0,j=0,p,rank=0, shift;
```

Выделение памяти под матрицу.

```
   float **mass=(float**)malloc(str*sizeof(float*));
   for(p=0;p<str;p++)
      mass[p]=(float*)malloc(col*sizeof(float));
```

Вызов функции для записи элементов в матрицу.

```
   creation_matrix(s,mass,str);
```

Приведение матрицы к ступенчатому виду методом Гаусса.

```
   for(i=0;i<str;i++){
```

Вызов функции обмена значений строк, если это необходимо, и определение значения сдвига, нужного для правильного выполнения

программы.

```
shift=FinalSwap(mass,i,i,str);
```

Деление каждого элемента строки, начиная с конца, на первый ненулевой элемент i-ой строки или же отсутствие действий, если строка полностью нулевая.

```
for(j=1;j<str+1;j++){
   if(i+shift<str)
      mass[i][col-j]=mass[i][col-j]/mass[i][i+shift];
}
```

Сложение элементов i-ой строки, умноженных на элементы следующих строк того же столбца, с соотв. элементами других строк или же отсутствие действий, если строка полностью нулевая.

```
for(j=i+1;j<str;j++){
   for(p=i;p<str;p++){
      if(i+shift<str)
         mass[j][col-p-1+i]= mass[j][col-p-1+i]-
                  (mass[i][col-p-1+i]*mass[j][i+shift]);
   }
}
}
```

Вызов функции подсчёт количества ненулевых строк, число которых равно рангу матрицы.

```
find_rang(str,&rank,mass);
```

Очистка памяти.

```
for(i=0;i<str;i++)
   free(mass[i]);
free(mass);
return rank;
}
```

1.3. Функция main

Функция main включает в себя: открытие и чтение входного файла, считывания из его матрицы в строку, определение размера матрицы, а также создание выходного файла с записанным в него рангом полученной матрицы или же вывод сообщения об ошибке.

Для определения размера матрицы создана отдельная функция

```
void Size(char *s, int *str, int *col){
  int newstr=0,i=0,all=0;
```

Подсчёт количества всех элементов в матрице и количества строк: если встречается пробел, то увеличивается счётчик элементов, в матрице, если символ перевода строки, то увеличиваются оба счётчика.

```
  while(s[i]!='\0'){
    if(s[i]==' ')
      all++;
    if (s[i]=='\n'){
      all++;
      newstr++;
    }
    i++;
  }
```

Увеличивается число строк и всех элементов на 1, потому что последняя строка и символ не включены из-за отсутствия символа перевода строки после них.

```
  *str=newstr+1;
```

Данный метод вычисления количества столбцов верный, т.к. матрица по условию корректна.

```
  *col=(all+1)/(newstr+1);
```

```
}
```

Сама функция main:

```
int main(){
    char s[100];
    char symb;
    int i=0,str=0,col=0,rank;
```

Открытие входного файла для чтения и посимвольные считывание и запись его содержимого в строку.

```
    FILE *input_file=fopen("./input_file.txt","r");
    while(fscanf(input_file,"%c",&symb)>0)
        s[i++]=symb;
```

Вызов функции, определяющей размер матрицы.

```
    Size(s,&str,&col);
```

Вызов функции, проверяющей равенство строк и столбцов матрицы, высчитывающей ранг матрицы и возвращающей либо значение ранга матрицы, либо «-1», которая соответствует ошибке.

```
    rank=calc_rank(s,str,col);
```

Если матрица не является квадратной, то выводится сообщение об ошибке, происходит закрытие входного файла и не создаётся выходной файл.

```
    if (rank==-1){
        printf("Fail\n");
        fclose(input_file);
        return 0;
    }
```

Если матрица является квадратной, то создаётся выходной файл, в который записывается значение ранга матрицы, после чего происходит закрытие открытых потоков.

```
    FILE *output_file=fopen("./output_file.txt","w");
    fprintf(output_file,"%d",rank);
    fclose(input_file);
    fclose(output_file);
    return 0;
}
```

2. Работа с Makefile

Для удобства работы с проектом совершено разбиение его на несколько файлов:

- main.c – основная функция;
- Swaps.c – реализация нужных функций перестановок;
- Swaps.h – объявление прототипов функций из “Swaps.c”;
- Gauss.c – реализация метода Гаусса и нахождение ранга;
- Gauss.h – объявление прототипов функций из “Gauss.h”.

Для упрощения и ускорения компиляции проекта, создан Makefile:

```
rank: main.o Gauss.o Swaps.o
	gcc main.o -o rank Gauss.o Swaps.o
	rm *.o

main.o: main.c Gauss.h
	gcc -c main.c

Gauss.o: Gauss.c Gauss.h Swaps.h
	gcc -c Gauss.c

Swaps.o: Swaps.c Swaps.h
	gcc -c Swaps.c
```

3. Тестирование программы

Рассмотрим различные варианты матриц и результат выполнения программы:

- Неквадратная матрица, у которой больше столбцов:

Input:

```
1 0 0 1
0 1 0 2
```

Output:

```
~/workspace/Kurs_2_Sem/ $ ./rank
Fail
```

- Неквадратная матрица, у которой больше строк:

Input:

```
1 0
0 1
3 2
5 1
```

Output:

```
~/workspace/Kurs_2_Sem/ $ ./rank
Fail
```

- Квадратная ступенчатая матрица(единичная) (ранг 4):

Input:

```
1 0 0 0
0 1 0 0
0 0 1 0
0 0 0 1
```

Output:

```
4
```

- Обычная квадратная матрица (ранг 4):

Input:

```
1 0 3 2
0 1 1 5
3 2 4 5
5 1 1 7
```

Output:

```
4
```

- Квадратная матрица из задания к данной работе (ранг 3):

Input:

```
1 2 3
3 4 5
4 6 10
```

Output:

```
3
```

- Квадратная матрица с линейно зависимой строкой (ранг 3):

Input:

```
1 0 3 2
1 0 3 2
3 2 4 5
5 1 1 7
```

Output:

```
3
```

- Квадратная матрица, в которой явно задействован сдвиг (ранг 2):

Input:

```
1 0 0 0
0 0 0 1
0 0 0 5
0 0 0 7
```

Output:

```
2
```

- Нулевая квадратная матрица (ранг 0):

Input:

```
0
```

Output:

```
0
```

Заключение

В ходе выполнения данной работы, были закреплены на практике знания о работе с файловой системой и двумерными массивами. Это было выполнено на примере программы, получающей на вход матрицу в текстовом файле, и выводящей её ранг в новый файл или же сообщения об ошибке на консоль. Помимо этого, было совершено разбиение проекта на отдельные файлы для удобства использования и простоты добавления новых функций.

Список использованных источников

1. Шилдт Г. – Полный справочник по С.-М.: Вильямс, 2004. -752 с.
2. Керниган Б., Ритчи Д. Язык программирования Си.\Пер. с англ., 3-е изд., испр. — СПб.: "Невский Диалект", 2001. - 352 с.
3. Wikipedia. – Метод Гаусса.
   URL: https://ru.wikipedia.org/wiki/Метод_Гаусса.

Приложение А. Исходный код

Файл *main.c*:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "Gauss.h"

void Size(char *s, int *str, int *col){
    int newstr=0,i=0,all=0;

    while(s[i]!='\0'){
        if(s[i]==' ')
            all++;
        if (s[i]=='\n'){
            all++;
            newstr++;
        }
        i++;
    }
    *str=newstr+1;
    *col=(all+1)/(newstr+1);
}


int main(){
    char s[100];
    char symb;
    int i=0,str=0,col=0,rank;

    FILE *input_file=fopen("./input_file.txt","r");
    while(fscanf(input_file,"%c",&symb)>0)
        s[i++]=symb;
    Size(s,&str,&col);

    rank=calc_rank(s,str,col);
    if (rank==-1){
        printf("Fail\n");
        fclose(input_file);
        return 0;
    }

    FILE *output_file=fopen("./output_file.txt","w");
    fprintf(output_file,"%d",rank);
    fclose(input_file);
    fclose(output_file);
```

```
    return 0;
}
```

Файл *Gauss.c*:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "Swaps.h"
#include "Gauss.h"

int find_rang(int str, int *rank, float **mass){
    int i,j;
    for(i=0;i<str;i++)
        for(j=0;j<str;j++)
            if (mass[i][j]==1){
                (*rank)++;
                j=str;
        }
}

void creation_matrix(char* s, float** mass,int str){
    int i=0,j=0;
    char *k=strtok(s," \n");
    while(k){
        if(j==str){
            i++;
            j=0;
        }
        mass[i][j]=atoi(k);
        j++;
        k=strtok(NULL," \n");
    }

}

int calc_rank(char *s, int str, int col){
    if(str!=col)
        return -1;
    int i=0,j=0,p,rank=0, shift;

    float **mass=(float**)malloc(str*sizeof(float*));
    for(p=0;p<str;p++)
        mass[p]=(float*)malloc(col*sizeof(float));

    creation_matrix(s,mass,str);
```

```
    for(i=0;i<str;i++){

        shift=FinalSwap(mass,i,i,str);

        for(j=1;j<str+1;j++){
            if(i+shift<str)
                mass[i][col-j]=mass[i][col-j]/mass[i][i+shift];
        }

        for(j=i+1;j<str;j++){
            for(p=i;p<str;p++){
                if(i+shift<str)
                    mass[j][col-p-1+i]= mass[j][col-p-1+i]-(mass[i][col-p-
1+i]*mass[j][i+shift]);
            }
        }
    }

    find_rang(str,&rank,mass);
    for(i=0;i<str;i++)
        free(mass[i]);
    free(mass);
    return rank;
}
```

Файл *Gauss.h*:

```
int find_rang(int str, int *rank, float **mass);
void creation_matrix(char* s, float** mass,int str);
int calc_rank(char *s, int str, int col);
```

Файл *Swaps.c*:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "Swaps.h"

void swap(float* prevEl, float* newEl){
    float c=*prevEl;
    *prevEl=*newEl;
    *newEl=c;
}
void swapstr(float* mass[], int prevStr, int newStr, int col){
    int i;
```

```
    for(i=0;i<col;i++){
        swap(&mass[prevStr][i],&mass[newStr][i]);
    }
}
int FinalSwap(float** mass,int i,int newi, int str){
  int shift=0,j;
  if(newi<str && mass[i][newi]==0 ){
    for(j=i+1;j<str;j++)
      if(mass[j][newi]!=0){
        swapstr(mass,i,j,str);
      }
  }
  if(mass[i][newi]==0){
    shift++;
    return shift+FinalSwap(mass,i,newi+1,str);
  }
  return shift;
}
```

Файл *Swaps.h*:

```
void swap(float* prevEl, float* newEl);
void swapstr(float* mass[], int prevStr, int newStr, int col);
int FinalSwap(float** mass,int i,int newi, int str);
```

Файл *Makefile*:

```
rank: main.o Gauss.o Swaps.o
	gcc main.o -o rank Gauss.o Swaps.o
	rm *.o
main.o: main.c Gauss.h
	gcc -c main.c
Gauss.o: Gauss.c Gauss.h Swaps.h
	gcc -c Gauss.c
Swaps.o: Swaps.c Swaps.h
	gcc -c Swaps.c
```