Московский Авиационный Институт

(Национальный исследовательский университет)

**Институт №8 “Информационных технологий и прикладной математики”**

**Курсовой проект**

**по курсу**

**«Практикум программирования»**

**II семестр**

**Задание 7**

Студент: Калуцкий М.В.

Группа: М8О-107Б-22

Руководитель: Аносова Н.П.

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_\_

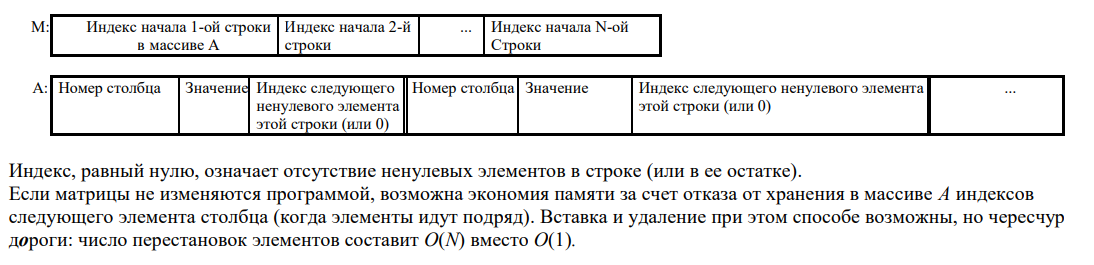
Подпись преподавателя: \_\_\_\_\_\_\_\_\_\_\_\_

* 1. **Тема:** Разряженные матрицы.

1. **Цель работы:**  Составить программу на языке Си++ с процедурами и/или функциями для обработки прямоугольных разреженных матриц с элементами целого типа, которая:

* вводит матрицы различного размера, представленные во входном текстовом файле в обычном формате (по строкам), с одновременным размещением ненулевых элементов в разреженной матрице в соответствии с заданной схемой;
* печатает введенные матрицы во внутреннем представлении согласно заданной схеме размещения и в обычном (естественном) виде;
* выполняет необходимые преобразования разреженных матриц (или вычисления над ними) путем обращения к соответствующим процедурам и/или функциям;
* печатает результат преобразования (вычисления) согласно заданной схеме размещения и в обычном виде.

1. **Задание:** Представить матрицу в соответствии со структурой:

****

Реализовать преобразование: Определить максимальный по модулю элемент матрицы и разделить на него все элементы столбца, в котором он находится. Если таких элементов несколько, обработать предпоследний столбец, содержащий такой элемент.

1. **Оборудование** (лабораторное):

*Оборудование ПЭВМ студента, если использовалось:*

Процессор Intel Core i3-4005U CPU с ОП 8096 Мб, НМД \_ 131072\_ Мб. Монитор lenovo

1. **Программное обеспечение (лабораторное):**

*Программное обеспечение ЭВМ студента, если использовалось:*

Операционная система семейства \_\_\_Unix\_\_\_\_, наименование \_\_ Ubuntu \_\_\_\_\_\_\_\_\_\_\_\_ версия 4.4.2

интерпретатор команд \_\_\_\_bash\_\_\_\_\_\_\_\_ версия \_\_5.0.17\_\_\_\_.

Система программирования \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_версия \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Редактор текстов \_\_\_\_\_nano\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ версия \_25.2.2 \_\_\_\_\_\_\_\_\_

Местонахождение и имена файлов программ и данных на домашнем компьютере \_\_\_\_\_home/jekich228\_\_\_\_\_\_\_\_\_

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

Вся программа состоит из 2-х файлов, первый – “main.cpp”, второй – текстовый файл “matrix.txt”, в котором содержится размерность матрицы и сама матрица.

Считывание матрицы будет осуществляться через перенаправление потока ввода.

Для представления матрицы в соответствующей структуре были созданы 2 вектора:

vector<int> M;

vector<a> A;

Вектор “A” содержит в себе объекты структуры “a”:

struct a{

    int number\_column;

    float value;

    int index\_next;

};

Заполнение векторов происходит при считывании по такому алгоритму: сначала считывается размерность матрицы, далее поочередно считываются её строки, причем для каждой строки создается флаг, который отображает, есть ли в строке ненулевые элементы или нет. Если их нет, то флаг остается неизменным и, следовательно, в вектор “М” идёт нуль, если же ненулевые элементы в строке есть, то флаг меняет своё значение, после чего в вектор “М” записывается номер этой строки. Вектор “A” заполняется более сложным образом: для каждой строки я создаю вспомогательный вектор, состоящий из структур “a”. Если я встречаю в строке число отличное от нуля, то создаю экземпляр структуры, в поле значения которого кладу считанное ненулевое число, в поле столбца кладу переменную j+1, которая пробегается по всем индексам столбцов, а поле индекса следующего числа заполняю тогда, когда нахожу очередное ненулевое число в строке. Причем следует учитывать, что у последнего элемента в векторе “A” поле следующего ненулевого элемента в строке заполнено нулём.

Для вывода матрицы в стандартном виде заводим счетчик “k”, который будет отображать, сколько элементов вектора “А” уже было отработано, и далее пробегаемся по вектору “M”:

* Если элемент в векторе равен нулю, то выводим строку из n нулей, так как в данной строке матрицы нет ненулевых элементов;
* Если элемент не равен нулю, то рассматриваются элементы вектора “A”, начиная с k-ого до того момента, пока не был встречен элемент со значением нуль в поле индекса следующего элемента, причем при каждом проходе элемента вектора “A”, счетчик “k” увеличивается на 1. Сами же элементы вектора “A” обрабатываются так: зная номер столбца элемента, мы сначала выводим недостающее кол-во нулей в строке, после чего выводим само значение в узле и переходим к следующей итерации.

Для выполнения самого задания с матрицей, а именно нахождение максимального элемента в матрице и деление всех элементов столбца матрицы, в котором находится максимальный, я сделал следующее: создал свою структуру “pr”:

struct pr

    {

        int x;

        int column = -1;

    };

В этой структуре есть поле значения и поле номера столбца, после чего я создаю 2 экземпляра данного класса, так как по условию задачи, если максимальных элементов будет несколько, то нужно будет обработать тот столбец, который является предпоследним из тех, куда входит максимальное число. Далее я нахожу элемент в массиве “A”, который имеет наибольшее значение в поле “value”, причем вместе с этим я храню и элемент, который является предпоследним по этой величине.

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

Файл “main.cpp”:

#include<bits/stdc++.h>

using namespace std;

struct a{

    int number\_column;

    float value;

    int index\_next;

};

vector<int> M;

vector<a> A;

int m,n;

ifstream in("matrix.txt");

struct pr

    {

        int x;

        int column = -1;

    };

void foo(){

    float integer\_number;

    for(int i = 0;i<m;++i){

        bool flag = 1;

        vector<a>lol;

        for(int j = 0;j<n;++j){

            in >> integer\_number;

            if(integer\_number != 0){

                flag = 0;

                if(lol.size()!=0){

                    lol[lol.size()-1].index\_next = j+1;

                }

                a kek;

                kek.number\_column = j+1;

                kek.value = integer\_number;

                lol.push\_back(kek);

            }

        }

        if(flag == 1) M.push\_back(0);

        else{

            M.push\_back(i+1);

            lol[lol.size()-1].index\_next = 0;

            for(a i : lol){

                A.push\_back(i);

            }

        }

    }

}

void print\_A(){

    cout<<endl;

    cout<<"Array A: [ ";

    for(a i :A){

        cout<<"{"<<i.number\_column<<", "<<i.value<<", "<<i.index\_next<<"}"<<" ";

    }

    cout<<"]"<<endl;

}

void print\_M(){

    cout<<endl;

    cout<<"Array M: [ ";

    for(int i : M) cout<<i<<" ";

    cout<<"]"<<endl;

}

void print\_matrix(){

    int k = 0;

    for(int i = 0; i < m; ++i){

        if(M[i] == 0){

            for(int j = 0; j < n; ++j){

                cout<<0<<" ";

            }

            cout<<endl;

        }

        else{

            if(A[k].index\_next == 0){

                int ll = 0;

                int iter = A[k].number\_column-ll-1;

                    for(int i = 0;i<iter;++i){

                        cout<<0<<" ";

                        ll++;

                    }

                cout<<A[k].value<<" ";

                ll++;

                for(int i = 0; i< (n-ll);++i){

                    cout<<0<<" ";

                }

                k++;

                cout<<endl;

            }

            else{

                int l = 0;

                while(A[k].index\_next != 0){

                    int iter = A[k].number\_column-l-1;

                    for(int i = 0;i<iter;++i){

                        cout<<0<<" ";

                        l++;

                    }

                    cout<<A[k].value<<" ";

                    l++;

                    k++;

                }

                int iter = A[k].number\_column-l-1;

                for(int i = 0;i<iter;++i){

                        cout<<0<<" ";

                        l++;

                    }

                cout<<A[k].value<<" ";

                    l++;

                    k++;

                for(int i = 0; i< (n-l);++i){

                    cout<<0<<" ";

                }

                cout<<endl;

            }

        }

    }

}

void max\_modul(){

    pr x1,x2;

    x1.x = -100000;

    x2.x = -200000;

    for(a i:A){

        if(i.value >= x2.x){

            if(x1.x == i.value){

                if(i.number\_column<x1.column){

                    x2.column = x1.column;

                    x2.x = x1.x;

                    x1.column = i.number\_column;

                    continue;

                }

            }

            x1.x = x2.x;

            x2.x = i.value;

            x1.column = x2.column;

            x2.column = i.number\_column;

        }

    }

    if(x1.x == x2.x){

        for(int i = 0;i<A.size();++i){

            if(A[i].number\_column == x1.column){

                A[i].value = A[i].value/(x1.x);

            }

        }

        return;

    }

    for(int i = 0;i<A.size();++i){

            if(A[i].number\_column == x2.column){

                A[i].value = (A[i].value)/(x2.x);

            }

        }

}

int main(){

    in>>m>>n;

    foo();

    print\_A();

    print\_M();

    print\_matrix();

    cout<<endl;

    max\_modul();

    cout<<endl;

    print\_matrix();

}

1. **Распечатка протокола** (подклеить листинг окончательного варианта программы с тестовыми примерами,подписанныйпреподавателем).

**Введённая матрица:**

3 4

0 1 2 7

1 7 3 7

1 1 0 2

**Результат работы программы**:

Array A: [ {2, 1, 3} {3, 2, 4} {4, 7, 0} {1, 1, 2} {2, 7, 3} {3, 3, 4} {4, 7, 0} {1, 1, 2} {2, 1, 4} {4, 2, 0} ]

Array M: [ 1 2 3 ]

0 1 2 7

1 7 3 7

1 1 0 2

0 0.142857 2 7

1 1 3 7

1 0.142857 0 2

1. **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. | Дата | Время | Событие | Действие по исправлению | Примечание |
|  | или |  |  |  |  |  |
|  | дом. |  |  |  |  |  |
|  |  |  |  |  |  |  |

1. **Замечания автора** по существу работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
2. **Выводы**: Проделав данную работу, я научился составлять программы на языке Си++ с процедурами и/или функциями для обработки прямоугольных разреженных матриц с элементами целого типа, которые:

* вводят матрицы различного размера, представленные во входном текстовом файле в обычном формате (по строкам), с одновременным размещением ненулевых элементов в разреженной матрице в соответствии с заданной схемой;
* печатают введенные матрицы во внутреннем представлении согласно заданной схеме размещения и в обычном (естественном) виде;
* выполняют необходимые преобразования разреженных матриц (или вычисления над ними) путем обращения к соответствующим процедурам и/или функциям;
* печатают результат преобразования (вычисления) согласно заданной схеме размещения и в обычном виде.

Недочёты при выполнении задания могут быть устранены следующим образом: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись студента \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_