**Московский Авиационный Институт**

(Национальный Исследовательский Университет)

Институт №8 «Информационные технологии и прикладная математика»

Кафедра 806 «Вычислительная математика и программирование»

Курсовой проект

по дисциплине «Практикум программирования»

2 семестр

Задание 7.

Тема: «Разреженные матрицы»

|  |  |
| --- | --- |
| Студент: | Доянова В. А. |
| Группа: | М8О-111Б-22 |
| Преподаватель: | Аносова Н. П. |
| Подпись: |  |
| Оценка: |  |

Москва, 2023

Содержание:

**Задание3**

Формулировка3

**Работа4**

Код программы4

Описание алгоритма6

Протокол выполнения программы7

**Вывод8**

**Задание**

**Формулировка**

Составить программу на языке Си с процедурами и/или функциями для обработки прямоугольных разреженных матриц с элементами целого типа, которая:

1. 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;

2. печатает введенные матрицы во внутреннем представлении согласно заданной схеме размещения и в обычном (естественном) виде;

3. выполняет необходимые преобразования разреженных матриц (или вычисления над ними) путем обращения к соответствующим процедурам и/или функциям;

4. печатает результат преобразования (вычисления) согласно заданной схеме размещения и в обычном виде.

Схема размещения матрицы: один вектор: 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.

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.

**Работа**

**Код программы**

matrix.cpp: #pragma once

#include <string>

#include <vector>

using namespace std;

struct Matrix {

int n;

int m;

vector<int> col\_index;

vector<int> values;

vector<int> row\_index;

};

Matrix read(const string& filename);

void print(const Matrix& matrix);

void max\_abs\_el(Matrix& matrix);

functions.cpp: #include "matrix.h"

#include <fstream>

#include <iostream>

#include <sstream>

using namespace std;

Matrix read(const string& filename) {

ifstream input(filename);

Matrix matrix;

input >> matrix.n >> matrix.m;

matrix.row\_index.resize(matrix.n + 1, 0);

int cur\_row = 1;

while (!input.eof()) {

string line;

getline(input, line);

if (line.empty()) continue;

istringstream iss(line);

int row, col, val;

iss >> row;

if (row != cur\_row) {

for (int i = cur\_row; i < row; ++i) {

matrix.row\_index[i] = matrix.col\_index.size();

}

cur\_row = row;

}

matrix.row\_index[cur\_row] = matrix.col\_index.size();

while (iss >> col >> val) {

matrix.col\_index.push\_back(col - 1);

matrix.values.push\_back(val);

}

}

for (int i = cur\_row; i <= matrix.n; ++i) {

matrix.row\_index[i] = matrix.col\_index.size();

}

return matrix;

}

void print(const Matrix& matrix) {

for (int row = 0; row < matrix.n; ++row) {

for (int col = 0; col < matrix.m; ++col) {

bool found = false;

for (int i = matrix.row\_index[row]; i < matrix.row\_index[row + 1]; ++i) {

if (matrix.col\_index[i] == col) {

cout << matrix.values[i] << '\t';

found = true;

break;

}

}

if (!found) {

cout << "0\t";

}

}

cout << '\n';

}

}

void max\_abs\_el(Matrix& matrix) {

double max\_abs = 0;

double abs\_val = 0;

int max\_col = -1;

vector<int> max\_cols;

for (int col = 0; col < matrix.m; ++col) {

for (int i = matrix.row\_index[0]; i < matrix.row\_index[matrix.n]; ++i) {

if (matrix.col\_index[i] == col) {

double abs\_val = abs(matrix.values[i]);

if (abs\_val > max\_abs) {

max\_abs = abs\_val;

max\_col = col;

}

}

}

}

for (int col = 0; col < matrix.m; ++col) {

for (int i = matrix.row\_index[0]; i < matrix.row\_index[matrix.n]; ++i) {

if (matrix.col\_index[i] == col) {

double abs\_val = abs(matrix.values[i]);

if (abs\_val == max\_abs) {

max\_cols.push\_back(col);

}

break;

}

}

if (max\_cols.size() >= 2) {

max\_col = max\_cols[max\_cols.size() - 2];

}

}

if (max\_col >= 0) {

for (int i = matrix.row\_index[0]; i < matrix.row\_index[matrix.n]; ++i) {

if (matrix.col\_index[i] == max\_col) {

matrix.values[i] /= max\_abs;

}

}

}

}

main.cpp: #include "matrix.h"

#include <iostream>

using namespace std;

int main() {

setlocale(LC\_ALL, "Russian");

Matrix matrix = read("input.txt");

cout << "Введенная матрица:\n";

print(matrix);

max\_abs\_el(matrix);

cout << "\nМатрица после преобразования:\n";

print(matrix);

return 0;

}

**Описание алгоритма**

matrix.h: создание структуры матрицы, объявление функций программы.

* n – количество строк матрицы;
* m – количество столбцов матрицы;
* col\_index – массив с номерами столбцов матрицы, где есть ненулевые значения;
* values – массив с ненулевыми значениями матрицы;
* row\_index – массив с номерами ненулевых строк матрицы.

functions.cpp:

* read – функция считывает из файла размерность матрицы, затем создает массив соответствующего размера. Далее в цикле функция считывает строки файла и получает номер строки матрицы, номер столбца и значение и заполняет массив соответствующе.
* print – функция идет по двум циклам исходя из размерности матрицы. Она идет по созданным массивам и выводит элементы на соответствующих им местах.
* max\_abs\_el – функция проходит по ненулевым элементам матрицы и ищет максимальный элемент. Далее она еще раз проходит по матрице и запоминает номера столбцов, в которых есть найденный максимальный элемент. Если столбцов несколько, то функция делит на максимальный элемент именно предпоследний столбец из найденных.

main.cpp: происходит вызов описанных ранее функций.

**Протокол выполнения программы**

1. Входные данные:

5 7

1 1 2634 3 7463 5 1 0 0

4 2 7463 0 0

5 1 1 2 2634 5 5870 7 9 0 0

Вывод программы:

Введенная матрица:

2634 0 7463 0 1 0 0

0 0 0 0 0 0 0

0 0 0 0 0 0 0

0 7463 0 0 0 0 0

1 2634 0 0 5870 0 9

Матрица после преобразования:

2634 0 7463 0 1 0 0

0 0 0 0 0 0 0

0 0 0 0 0 0 0

0 1 0 0 0 0 0

1 0 0 0 5870 0 9

1. Входные данные:

4 6

1 1 26 3 46 5 1 0 0

2 2 21 0 0

4 1 1 2 24 5 3 6 9 0 0

Вывод программы:

Введенная матрица:

26 0 46 0 1 0

0 21 0 0 0 0

0 0 0 0 0 0

1 24 0 0 3 9

Матрица после преобразования:

26 0 1 0 1 0

0 21 0 0 0 0

0 0 0 0 0 0

1 24 0 0 3 9

**Вывод**

Во время работы над этой курсовой работой я разобралась с вводом разреженной матрицы, выводом её, а также с работой с ней. Я научилась находить максимальное значение и использовать его для работы над целым столбцом матрицы.