МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра информатики и систем управления

Разреженные матрицы

ОТЧЕТ

по лабораторной работе 2

по дисциплине

Алгоритмы и структуры данных

РУКОВОДИТЕЛЬ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ С. Капранов

(подпись)

СТУДЕНТ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Бубнов Е.А

(подпись)

19-ивт-2

(шифр группы)

Работа защищена «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_

С оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Нижний Новгород 2020

**Задание**

Задание 9.

Даны две разреженные матрицы. Из одной матрицы вычесть другую и результат занести в разреженную матрицу CRS.

Определить: 1) создать две разреженные матрицы

2) вычесть

3) Преобразовать итоговую ленточную матрицу в формат CRS

**Код**

***main.cpp***

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* ASD II лаба \*/

/\* Автор: Бубнов Евгений 19-ивт-2 \*/

/\* Задача: Даны две разреженные ленточные матрицы.

Из одной матрицы вычесть другую и

результат занести в разреженную матрицу CRS.\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

#include "SparseMatrix.h"

//обработка ошибок

void handing\_error(const int error);

int main()

{

//N - размер квадратной матрицы

//q - верхняя ширина ленты

//p - нижняя ширина ленты

int N1, q1, p1; //данные о первой матрице

int N2, q2, p2; //данные о второй матрице

try

{

cout << "Matrix1: " << endl;

input\_data\_matrix(&N1, &q1, &p1); // заполнение данных о 1-ой матрице

cout << "Matrix2: " << endl;

input\_data\_matrix(&N2, &q2, &p2); // заполнение данных о 2-ой матрице

//создание ленточно-строчной матрицы

SS matrix1(N1, q1, p1), matrix2(N2, q2, p2);

//заполнение

matrix1.create();

matrix2.create();

//вывод на экран 2-ух матриц

cout << "Matrix 1: " << endl;

matrix1.print();

cout << endl;

cout << "Matrix 2: " << endl;

matrix2.print();

cout << endl;

//нахождение разницы двух матриц

SS matrix = matrix1 - matrix2;

matrix.print(); //вывод промежуточного результата

cout << endl;

//преобразование ленточно-строчной матрицы в разреженно-строчный формат

CRS crs\_matrix(matrix);

crs\_matrix.print(); //вывод результата

}

catch(int error)

//обработка ошибок и вывод пользователю сообщения об ошибке

{

handing\_error(error);

}

return 0;

}

void handing\_error(const int error)

//обработка ошибок

{

cout << "Error! ";

//если если ввведены некоректные данные для создания матрицы

if (error == -3)

cout << "invalid data was entered for creating the matrix" << endl;

//если ввели лишние буквы

if (error == -2)

cout << "Input int value" << endl;

//если ошиблись с адресом ячейки матрицы

if (error == -1)

cout << "Access to an element" << endl;

//если не попали в ленту

if (error == 1)

cout << "Еhe element is located outside the feed" << endl;

//если матрицы разного размера

if (error == 2)

cout << "Matrices of different sizes" << endl;

}

***SparseMatrix.h***

#include <iostream>

#include <cstdio>

#include "List.cpp" // для класса CRS

#include "InputIntValue.h" // для безопасного заполнения матрицы

using std::cout;

using std::cin;

using std::endl;

#define NULL\_STR -1

//класс ленточно-строчного формата матрицы

class SS

{

public:

int N; // размер квадратной матрицы

int q; // верхняя ширина ленты

int p; // нижняяя ширина ленты

int len; // общая длина ленты

int\*\* stip; // лента

SS() : stip(NULL) {} // конструктор по умолчанию

SS(const int \_N, const int \_q, const int \_p); // конструктор по значению

~SS(); // деструктор

//доступ к элементу

void set(const int i, const int j, const int data); // заполнение ячейки матрицы

int get(const int i, const int j) const; // получение значения ячейки матрицы

//заполнения матрицы

void create();

//вывод

void print();

//вычитание матриц

friend SS operator-(const SS& lMatrix, const SS& rMatrix);

};

//класс разреженно-строчного формата матрицы

class CRS

{

private:

List<int> value; // элементы

List<int> LI; // номер 1 элемента i строки

List<int> LJ; // номер столбца элемнта

public:

CRS(){} // конструктор по умолчанию

CRS(const SS& matrix); // конструктор копирования

~CRS(){} // деструктор

void print(); // вывод

};

***SparseMatrix.cpp***

#include "SparseMatrix.h"

SS::SS(const int \_N, const int \_q, const int \_p) : N(\_N), q(\_q), p(\_p), len(\_q + \_p + 1)

// конструктор по значению

{

//выделение памяти для ленты N-строк, len - столбцов

stip = new int\*[N];

for (int i = 0; i < N; i++)

stip[i] = new int[len];

}

SS::~SS()

// деструктор

{

//очистка памяти

if (stip != NULL)

{

for (int i = 0; i < N; i++)

delete [] stip[i];

delete stip;

}

}

void SS::set(const int i, const int j, const int data)

// заполнение ячейки матрицы

{

//если элемент находиться за границами, матрицами выдай ошибку

if (i < 0 || i >= N || j < 0 || j >= N)

throw -1;

//если элемент находиться в ленте, то запиши значение в ячейку

if ((i < j && (j - i <= q)) || (i > j && (i - j <= p)) || (i == j))

{

stip[i][j-i+p] = data;

return;

}

//элемент находиться не в ленте, выдай предупреждение

throw 1;

}

int SS::get(const int i, const int j) const

// получение значения ячейки матрицы

{

//если элемент находиться за границами, матрицами выдай ошибку

if (i < 0 || i >= N || j < 0 || j >= N)

throw -1;

//если элемент находиться в ленте, то верни значение

if ((i < j && (j - i <= q)) || (i > j && (i - j <= p)) || (i == j))

return stip[i][j-i+p];

//элемент находиться не в ленте, выдай 0

return 0;

}

void SS::create()

//заполнения матрицы

{

//заполнение

for (int i = 0; i < p; i++)

{

for (int j = 0; j <= q; j++)

{

cout << "stip[" << i << "][" << j << "] = ";

set(i, j, input\_int\_value());

}

}

for (int i = p, j = 0; i < N; i++, j++)

{

for (int j1 = j; (j1 < j+len) && (j1 < N); j1++)

{

cout << "stip[" << i << "][" << j << "] = ";

set(i, j, input\_int\_value());

}

}

}

void SS::print()

//вывод

{

//обход матрицы по строчно

for (int i = 0; i < N; i++)

{

for (int j = 0; j < N; j++)

{

printf("%3d ", get(i, j));

}

printf("\n");

}

}

SS operator-(const SS& lMatrix, const SS& rMatrix)

//вычитание матриц

{

if (lMatrix.N != rMatrix.N) throw 2;

int Q = (lMatrix.q > rMatrix.q)? lMatrix.q : rMatrix.q;

int P = (lMatrix.p > rMatrix.p)? lMatrix.p : rMatrix.p;

//создание

int cur\_value;

SS Matrix(lMatrix.N, Q, P);

//заполнение

for (int i = 0; i < P; i++)

{

for (int j = 0; j <= Q; j++)

{

cur\_value = lMatrix.get(i, j) - rMatrix.get(i, j); // нахождение разницы элемнтов

Matrix.set(i, j, cur\_value); // запись полученного значения

}

}

for (int i = P, j = 0; i < Matrix.N; i++, j++)

{

for (int j1 = j; (j1 < j+Matrix.len) && (j1 < Matrix.N); j1++)

{

cur\_value = lMatrix.get(i, j1) - rMatrix.get(i, j1); // запись полученного значения

Matrix.set(i, j1, cur\_value); // запись полученного значения

}

}

return Matrix;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*конец класса SS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*класс SRS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

CRS::CRS(const SS& matrix)

// конструктор копирования

{

int n = 0; //номер элемента

int elem; //значение эелемнта

bool have\_elem; //флаг, показывающий есть ли хотя бы 1 элемент в строке

for (int i = 0; i < matrix.p; i++)

{

have\_elem = false; //изначально элемнта нет

//запись элементов в матрицу

for (int j = 0; j <= matrix.q; j++)

{

elem = matrix.get(i, j);

if (elem != 0)

{

value.push\_back(elem);

LJ.push\_back(j);

if (!have\_elem)

LI.push\_back(n);

have\_elem = true;

n++;

}

}

//если элемента не нашлось, запиши пустую строку

if (!have\_elem)

LI.push\_back(NULL\_STR);

}

for (int i = matrix.p, j = 0; i < matrix.N; i++, j++)

{

have\_elem = false; //изначально элемнта нет

//запись элементов в матрицу

for (int j1 = j; (j1 < j+matrix.len) && (j1 < matrix.N); j1++)

{

elem = matrix.get(i, j1);

if (elem != 0)

{

value.push\_back(elem);

LJ.push\_back(j1);

if (!have\_elem)

LI.push\_back(n);

have\_elem = true;

n++;

}

}

//если элемента не нашлось, запиши пустую строку

if (!have\_elem)

LI.push\_back(NULL\_STR);

}

//запись кол-ва элементов

n+=2; //количество NZ + 1, индекс +1

LI.push\_back(n);

//обработка пустых строк LI

for (int i = 0; i < LI.size()-1; i++)

{

//записываю пустые строки

if (LI[i] == -1)

LI[i] = LI[i+1];

}

}

void CRS::print()

// вывод

{

//вывод номеров для опредления индекса

printf(" N: ");

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

printf("%3d ", i);

printf("\n");

//вывод value

printf("value: ");

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

printf("%3d ", value[i]);

printf("\n");

//вывод LJ

printf(" LJ: ");

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

printf("%3d ", LJ[i]);

printf("\n");

//вывод LI

printf(" LI: ");

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

printf("%3d ", LI[i]);

printf("\n");

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*конец класса CRS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

***InputIntValue.h***

#include <iostream>

using std::cout;

using std::cin;

using std::endl;

#define MAX\_BUFFER 255

//ф-я для безопасного ввода целого числа

int input\_int\_value();

//ф-я ввода данных для создания разреженной ленточной матрицы

void input\_data\_matrix(int\* N, int\* q, int\* p);

***InputIntValue.cpp***

#include "InputIntValue.h"

int input\_int\_value()

//ф-я для безопасного ввода целого числа

{

int value;

//ввод значения

cin >> value;

//проверка

if (cin.peek() != '\n')

{

cin.ignore(MAX\_BUFFER, '\n');

cin.clear();

throw -2;

}

return value;

}

void input\_data\_matrix(int\* N, int\* q, int\* p)

//ф-я ввода данных для создания разреженной ленточной матрицы

{

//ввод размера квадратной матрицы

cout << "Size matrix: ";

\*N = input\_int\_value();

//ввод верхнейц границы ленты

cout << "High border: ";

\*q = input\_int\_value();

//ввод нижней границы ленты

cout << "Low border: ";

\*p = input\_int\_value();

//проверка на корректность введенных парметров матрицы

if ( ( (\*N) <= 0 ) || ( (\*q) >= (\*N) ) || ( (\*q) < 0 ) || ( (\*p) >= (\*N) ) || ( (\*p) < 0 ) )

throw -3; //ошибка

}

***List.cpp***

#include <iostream>

using std::cout;

using std::cin;

using std::endl;

//список

template<typename LT>

class List

{

private:

//ячейка

template<typename NT>

class Node

{

public:

NT data; //данные

Node\* next; //указатель на следующий элемент

Node(NT \_data=NT(), Node\* \_next=NULL) : data(\_data), next(\_next) {}

~Node(){}

};

Node<LT>\* head; //указатель на говолу

Node<LT>\* end; //указатель на конец

int SIZE; //размер списка

public:

List() : head(NULL), end(NULL), SIZE(0) {}

~List();

void push\_back(const LT& \_data); // добавление в конец списка

void pop (int index); // удаление index элемнта

void print (); //вывод всего списка

LT& operator[](const int index); //дооступ к элементу

bool empty() { return (SIZE == 0)? true : false; } // проверка на пустоту списка

int size() { return SIZE; } // определение размера списка

};

template<typename LT>

List<LT>::~List()

/\*Удаляем список\*/

{

Node<LT>\* tmp;

while (head != NULL)

{

tmp = head;

head = head->next;

delete tmp;

}

}

template<typename LT>

void List<LT>::print()

/\*выводим список\*/

{

Node<LT>\* tmp = head; //создаем временнный объект для обхода

//пока не конец списка выводи элемнты

while (tmp != NULL)

{

cout << tmp->data << ' ';

tmp = tmp->next;

}

}

template<typename LT>

void List<LT>::push\_back(const LT& \_data)

/\*Добавление в конец списка\*/

{

//если элемнтов нет

if (SIZE == 0)

{

head = new Node<LT>(); //создаем елемент

head->data = \_data; //заполняем

end = head; //присваем новый хвост

}

else

{

Node<LT>\* tmp = new Node<LT>(); //создаем елемент

tmp->data = \_data; //заполняем

end->next = tmp; //говорим что n-ый элемент указывает на наш новый элемент

end = tmp; //присваем новый хвост

}

SIZE++; //увеливаем количтво элементов

}

template<typename LT>

void List<LT>::pop(int index)

/\* Удаление index элемента \*/

{

if ((head != NULL) && (index >= 0) && (index < SIZE)) // если по этому номеру что-то лежит и этот элемент внутри списка

{

// Mass - объекты, которые хранятся в списке

Node<LT> \*tmp = head, \*helping = head;

for (int i = 0; i < index; i++)

{

helping = tmp; // предыдущее значение temp

tmp = tmp->next;

}

if (tmp == head) // если элемент который надо удалить первый

{

head = tmp->next;

}

else

{

helping->next = tmp->next;

}

delete(tmp);

SIZE--; // уменьшаем размер списка

}

else

{

throw -1;

}

}

template<typename LT>

LT& List<LT>::operator[](const int index)

{

//если индекс элемента правильный

if (index >= 0 && index < SIZE)

{

//создаю временный указатель на элемент для обхода списка

Node<LT>\* tmp = head;

int count = 0; //счетчик

while (count < index) //пока не найден нужный элемент

{

tmp = tmp->next; // переходи на следующий элемент

count++;

}

return tmp->data; //возвращаю найденный данные

}

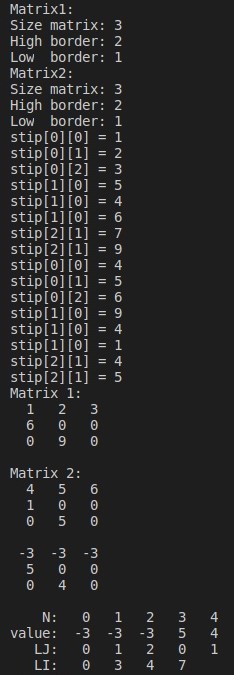
//индекс введен с ошибкой, возврат исключения

throw -1;

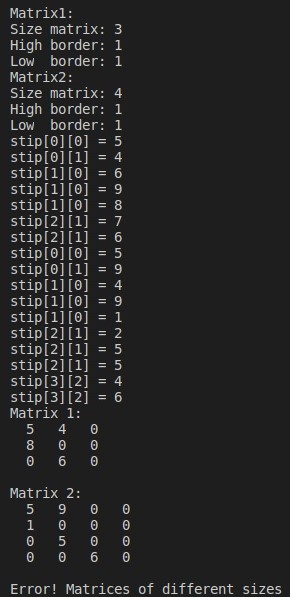
}

***Скриншоты***

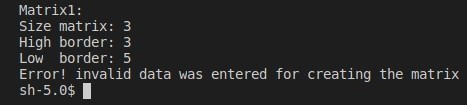
Правильная работа программы:



Если размерность матрицы не совпадает:



Если введен неверный нижний или верхний размер ленты:



Если введены ошибочные данные при заполнении матрицы:

