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

Отчет по лабораторной работе №2

Разреженные Матрицы

по дисциплине

Алгоритмы и структуры данных

РУКОВОДИТЕЛЬ:

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

СТУДЕНТ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Сухоруков В.А.

19-ИВТ-3

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

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

Нижний Новгород 2020

**Задача 21**

Дана разреженная структурно симметричная матрица. Найти количество её различных элементов и вывести их на экран.

**Входные данные**

Программе через консоль передаются количество строк и столбцов в матрице и её элементы.

**Результат работы программы**

Результатом является список уникальных элементов в матрице и их количество.

**Решение задачи**

Поскольку массивы AD, AU, Al, LI, и LJ являются динамическими структурами, то в реализации программы они были представлены как структуры типа список. В ходе работы над первой лабораторной работой мной был реализован шаблонный класс MyList. Для использования его во второй и последующих лабораторных работах класс MyList был записан в библиотеку MyList.h.

**Код библиотеки MyList.h**

#pragma once

#ifndef \_MyList\_H\_

#define \_MyList\_H\_

template <typename T> // создание шаблонного класса list

class List {

private: //приватная секция класса

template <typename T> //создание внутреннего класса для хранения

class Node { //элемента списка

public:

T value; //значение элемента

Node\* Next; //указатель на следующий элемент

//конструктор с параметрами

Node(T val = T(), Node\* n = nullptr) {

value = val;

Next = n;

}

};

int size; //Количество элементов в списке

public: //публичная секция класса

Node<T>\* head; //указатель на первый элемент

List() { // конструктор по умолчанию

head = nullptr;

size = 0;

}

~List() { //деструктор

delete[] head;

}

//метод добавления элемента в конец списка

void push\_back(T data) {

//если список пустой, то выделяем память и //добавляем элемент

if (head == nullptr) {

head = new Node<T>(data);

}

else { //иначе доходим до последнего элемента,

//выделяем память и добавляем элемент

Node<T>\* cur = head;

while (cur->Next != nullptr) {

cur = cur->Next;

}

cur->Next = new Node <T>(data);

}

size++;

}

T& operator[](const int index) { //перегрузка оператора, для

//обращения к элементам списка

int counter = 0;

Node <T>\* cur = head;

while (cur != nullptr) {

if (counter == index) {return cur->value;}

cur = cur->Next;

counter++;

}

}

int get\_size() { return size; } //Возвращает размер списка

};

#endif

**Код программы**

#include "MyList.h"

#include<iostream>

using namespace std;

int main() {

setlocale(LC\_ALL, "ru"); //Включение русского языка

int col=0,str=0; //Количество строк и стобцов в матрице

List<int> AD, AU, AL, LJ, LI; //Списки, отвечающие за хранение данных

//об элементах матрицы

List <int>LJ\_2; //Список для проверки того, что матрица

//является структурно симметричной

int element; //Элемент матрицы

List<int> different; //Список для хранения Различных

//элементов матрицы

cout << "Введите количество строк"<<endl;

cin >> str;

cout << "Введите количество столбцов" << endl;

cin >> col;

//Обработка введённых данных

if (str == 0 || col == 0) {

cout<<"Неверно заданы размеры матрицы."<<endl;

exit(-1);

}

if (str!=col){

cout << "Количетство строк не совпадает с количеством столбцов"

"Матрица не явлется структурно-симметричной" << endl;

exit(-1);

}

//Ввод элементов матрицы - заполнение

//списков

cout << "Введите элементы матрицы." << endl;

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

int flag = 0; //Показывает встречался ли ненулевой

//элемент в этой строке верхней части //матрицы, используется для заполнения //списка LI

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

cin >> element;

if (element != 0 && i < j) { flag++; }

if (i == j) { AD.push\_back(element);}

if (element != 0 && i < j) {

AU.push\_back(element);

LJ.push\_back(j+1);

}

if (element != 0 && i > j) {

AL.push\_back(element);

LJ\_2.push\_back(i + 1);

}

if (flag == 1 ) {

LI.push\_back(AU.get\_size());

flag = 2;

}

//Если мы находимся в верхней части

//матрицы, дошли до конца строки и не

//встретили ненулевого элемента, то //"пропускаем" клетку в списке LI

if (flag == 0 && i < j && j==col-1) {

LI.push\_back(0);

}

}

}

//Обработка "Пропущенных" клеток в

//списке LI. Если "Пропущена" последняя //клетка, записываем в неё количество //ненулевых элементов+1

if (LI[LI.get\_size() - 1] == 0) {

LI[LI.get\_size() - 1] = AU.get\_size() + 1;

}

for (int i = 0; i < LI.get\_size() -1; i++){

if (LI[i] == 0) {

int p = 0;

//Находим первую не нулевую клетку

//после серии нулей и переносим это //значение на нулевые клетки

for (int j = i; j < LI.get\_size() - 2; j++) {

if (LI[j + 1] != 0) {

p = LI[j + 1];

}

}

LI[i] = p;

}

}

//Проверка на то, что матрицы ялвяется структурно-симметричной

if (LJ\_2.get\_size() == LJ.get\_size()) {

int flag = 0;

for (int i = 0; i <LJ.get\_size(); i++){

if (LJ\_2[i] != LJ[i]) { flag = 1; }

}

if (flag == 0) {

cout << endl << "Матрица структурно-симметричная"<<endl;

cout << "Сформированный список AD: " << endl;

for (int i = 0; i < AD.get\_size(); i++) {

cout << AD[i] << " ";

}

cout << endl << "Сформированный список AU: " << endl;

for (int i = 0; i < AU.get\_size(); i++) {

cout << AU[i] << " ";

}

cout << endl << "Сформированный список AL: " << endl;

for (int i = 0; i < AL.get\_size(); i++) {

cout << AL[i] << " ";

}

cout << endl << "Сформированный список LJ: " << endl;

for (int i = 0; i < LJ.get\_size(); i++) {

cout << LJ[i] << " ";

}

cout << endl << "Сформированный список LI: " << endl;

for (int i = 0; i < LI.get\_size(); i++) {

cout << LI[i] << " ";

}

}

else{

cout << endl << "Матрица не является структурно-

симметричной" << endl;

exit(-1);

}

}

else {

cout << endl << "Матрица не является структурно-

симметричной"<<endl;

exit(-1);

}

//Поиск различных элементов в массиве

different.push\_back(AD[0]);

for (int i = 1; i < AD.get\_size() -1; i++) {

int flag = 0;

for (int j = 0; j < different.get\_size() - 1; j++) {

if (different[j] == AD[i]) {

flag = 1;

}

}

if (flag == 0) {

different.push\_back(AD[i]);

}

}

for (int i = 0; i < AU.get\_size() -1; i++) {

int flag = 0;

for (int j = 0; j < different.get\_size() -1; j++) {

if (different[j] == AU[i]) {

flag = 1;

}

}

if (flag == 0) {

different.push\_back(AU[i]);

}

}

for (int i = 0; i < AL.get\_size() -1; i++) {

int flag = 0;

for (int j = 0; j < different.get\_size() -1; j++) {

if (different[j] == AL[i]) {

flag = 1;

}

}

if (flag == 0) {

different.push\_back(AL[i]);

}

}

//Вывод различных элементов

cout <<"Количетство разлиных элементов: "<<different.get\_size()

<<endl<< "Различные элементы массива:" << endl;

for (int j = 0; j < different.get\_size() -1; j++) {

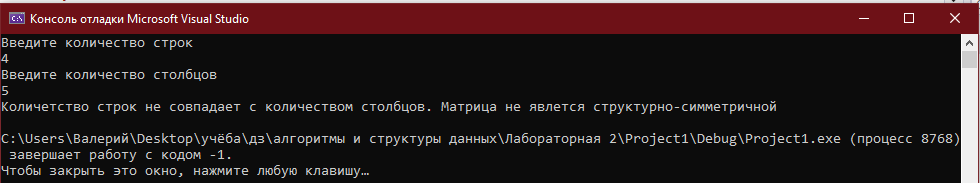
cout << different[j]<<" ";

}

cout << endl;

return 0;

}

**Результаты работы программы**

