Міністерство освіти і науки України

Національний університет “Львівська політехніка”

Кафедра інформаційних систем та мереж

ЗВІТ

про виконання лабораторної роботи № 1

з дисципліни Алгоритми та структури даних

Варіант 18

Виконав студент групи СА-32

Маленчак Валентин Богданович

Прийняв викладач Щербак О.Д.

*Львів 2017*

**Тема:** моделювання представлення в пам'яті векторів і таблиць.

**Мета роботи:** набуття навичок розміщення в пам'яті векторів і таблиць.

**Завдання на роботу:** ознайомитися з основними типами даних вибраної мови програмування та реалізувати калькулятор простих чисел з підтримкою зворотного польського запису. Додати можливість поелементого +,-,/,\* векторів та таблиць. Розробити спосіб економного зберігання в пам'яті розріджених матриць (таблиць). Розробити процедури і функції для забезпечення доступу (читання-запис) до елементів матриці. В контрольному прикладі забезпечити читання і запис всіх елементів матриці. Оцінити час виконання операцій.\

**Індивідуальне завдання:** всі нульові елементи розміщені на головній діагоналі і в нижній половині області нижче діагоналі.

# Хід роботи

Текст програми на мові С++:

// Файл Lab01.cpp

#include "stdafx.h"

#include <iostream>

#include "Calculator.h"

#include "Matrix.h"

#include "Interface.h"

using namespace std;

void CalculatorMain();

void MatrixMain();

void GetID(int id);

void InputCharacter(char & ch);

int main()

{

setlocale(LC\_ALL, "ukr");

char ch;

do

{

GetID(1);

InputCharacter(ch);

switch (ch)

{

case '1':

CalculatorMain();

break;

case '2':

MatrixMain();

break;

default:

break;

}

} while (ch == 49 || ch == 50);

system("pause");

}

// Файл Calculator.h

#pragma once

#include <iostream>

#include <string>

#include <stack>

#include "Interface.h"

using namespace std;

void GetID(int id);

void InputString(string & str);

void GetResult(double result);

bool Isprimenumber(int num)

{

if (num < 2)

{

GetID(7);

return false;

}

for (int i = 2; pow(i, 2) <= num; i++)

if (num % i == 0)

{

GetID(7);

return false;

}

return true;

}

int Priority(char a)

{

switch (a)

{

case '+':

case '-': return 1;

case '\*':

case '/': return 2;

case '^': return 3;

default: system("pause");

}

}

void Calculation(stack <double> & stk1, stack <char> & stk2)

{

double num2 = stk1.top();

stk1.pop();

double num1 = stk1.top();

stk1.pop();

switch (stk2.top())

{

case '+':

stk1.push(num1 + num2);

break;

case '-':

stk1.push(num1 - num2);

break;

case '\*':

stk1.push(num1 \* num2);

break;

case '/':

stk1.push(num1 / num2);

break;

case '^':

stk1.push(pow(num1, num2));

break;

}

stk2.pop();

}

bool CheckStr(string str, int sizestr)

{

int openbrackets = 0, closedbrackets = 0;

int operands = -1, operators = 0;

bool p = true;

if (str.at(sizestr - 1) != '=')

{

GetID(3);

return false;

}

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

{

switch (str.at(i))

{

case '1':

case '2':

case '3':

case '4':

case '5':

case '6':

case '7':

case '8':

case '9':

case '0':

if (p)

if (isdigit(str.at(i)))

{

operands++;

p = false;

}

break;

case '+':

case '-':

case '\*':

case '/':

case '^':

operators++;

p = true;

if (i == 0 || i == sizestr - 2)

{

GetID(9);

return false;

}

break;

case '(':

openbrackets++;

break;

case ')':

closedbrackets++;

if (str.at(i - 1) == '(')

{

GetID(11);

return false;

}

break;

case '=':

if (i != sizestr - 1)

{

GetID(10);

return false;

}

break;

default:

GetID(5);

return false;

break;

}

}

if (operands != operators)

{

GetID(4);

return false;

}

if (openbrackets != closedbrackets)

{

GetID(6);

return false;

}

return true;

}

bool PushNumberToStack(stack <double> & stk1, string & tmp)

{

try

{

stk1.push(stod(tmp));

if (!Isprimenumber(stoi(tmp)))

return false;

tmp = "\0";

}

catch (invalid\_argument)

{

GetID(17);

return false;

}

catch (out\_of\_range)

{

GetID(18);

return false;

}

return true;

}

void CalculatorMain()

{

string str;

stack <double> stk1;

stack <char> stk2;

string tmp = "\0";

int dynamicbrackets = 0;

unsigned int sizestr, sizestk2;

GetID(2);

InputString(str);

sizestr = str.size();

if (!CheckStr(str, sizestr))

return;

for (unsigned int i = 0; i < sizestr; i++)

{

switch (str.at(i))

{

case '1':

case '2':

case '3':

case '4':

case '5':

case '6':

case '7':

case '8':

case '9':

case '0':

tmp += str.at(i);

break;

case '+':

case '-':

case '\*':

case '/':

case '^':

sizestk2 = stk2.size();

if (!tmp.empty())

if (!PushNumberToStack(stk1, tmp))

return;

if (stk2.size() - dynamicbrackets == 0)

stk2.push(str.at(i));

else

{

for (unsigned int j = sizestk2; j > 0; j--)

{

if (stk2.top() == '(') break;

if (Priority(stk2.top()) >= Priority(str.at(i)))

Calculation(stk1, stk2);

}

stk2.push(str.at(i));

}

break;

case '(':

dynamicbrackets++;

stk2.push(str.at(i));

break;

case ')':

if (!tmp.empty())

if (!PushNumberToStack(stk1, tmp))

return;

while (stk2.top() != '(')

Calculation(stk1, stk2);

stk2.pop();

dynamicbrackets--;

break;

case '=':

if (!tmp.empty())

if (!PushNumberToStack(stk1, tmp))

return;

break;

default:

break;

}

}

while (stk1.size() != 1 && !stk2.empty())

Calculation(stk1, stk2);

GetID(8);

GetResult(stk1.top());

}

// Файл Matrix.h

#pragma once

#include <iostream>

#include <fstream>

#include <vector>

#include <string>

#include "Interface.h"

using namespace std;

void GetID(int id);

void InputSize(int & n);

void InputMatrixElement(double & element, int i, int j);

void OutputMatrixElement(double element, int i, int j);

void InputCharacter(char & ch);

void Memory(int memory1, int memory2);

void InputMainMatrix(vector < vector <double> > & vct, int n)

{

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

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

if (j > i)

InputMatrixElement(vct[i][j], i, j);

else

{

vct[i][j] = 0;

OutputMatrixElement(vct[i][j], i, j);

}

}

void InputAdditionalMatrix(vector < vector <double> > & tmp, int n)

{

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

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

InputMatrixElement(tmp[i][j], i, j);

}

void OutputMatrix(vector < vector <double> > vct, int n)

{

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

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

OutputMatrixElement(vct[i][j], i, j);

}

void AdditionMatrix(vector < vector <double> > & vct, int n)

{

vector < vector <double> > tmp(n, vector<double>(n));

GetID(15);

InputAdditionalMatrix(tmp, n);

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

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

vct[i][j] += tmp[i][j];

}

void SubtractionMatrix(vector < vector <double> > & vct, int n)

{

vector < vector <double> > tmp(n, vector<double>(n));

GetID(15);

InputAdditionalMatrix(tmp, n);

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

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

vct[i][j] -= tmp[i][j];

}

void MultiplicationMatrix(vector < vector <double> > & vct, int n)

{

vector < vector <double> > tmp1(n, vector<double>(n)), tmp2(n, vector<double>(n));

GetID(15);

InputAdditionalMatrix(tmp2, n);

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

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

{

tmp1[i][j] = vct[i][j];

vct[i][j] = 0;

}

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

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

for (int k = 0; k < n; k++)

vct[i][j] += tmp1[i][k] \* tmp2[k][j];

}

void DivisionMatrix(vector < vector <double> > & vct, int n)

{

vector < vector <double> > tmp1(n, vector<double>(n)), tmp2(n, vector<double>(n)), E(n, vector<double>(n));

double dtmp;

GetID(15);

InputAdditionalMatrix(tmp2, n);

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

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

{

tmp1[i][j] = vct[i][j];

vct[i][j] = 0;

if (i == j)

E[i][j] = 1;

else

E[i][j] = 0;

}

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

{

dtmp = tmp2[i][i];

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

{

tmp2[i][j] /= dtmp;

E[i][j] /= dtmp;

}

for (int k = i + 1; k < n; k++)

{

dtmp = tmp2[k][i];

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

{

tmp2[k][j] -= tmp2[i][j] \* dtmp;

E[k][j] -= E[i][j] \* dtmp;

}

}

}

for (int i = n - 1; i > 0; i--)

{

for (int k = i - 1; k >= 0; k--)

{

dtmp = tmp2[k][i];

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

{

tmp2[k][j] -= tmp2[i][j] \* dtmp;

E[k][j] -= E[i][j] \* dtmp;

}

}

}

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

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

for (int k = 0; k < n; k++)

vct[i][j] += tmp1[i][k] \* tmp2[k][j];

}

void SaveMatrix(vector < vector <double> > vct, int n)

{

ofstream myfile;

myfile.open("MyFile.csv", ios\_base::out | ios\_base::trunc);

if (!myfile.is\_open())

{

GetID(16);

return;

}

myfile << n;

myfile << "\n";

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

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

if (vct[i][j] != 0)

{

myfile << vct[i][j];

myfile << ';';

myfile << i;

myfile << ';';

myfile << j;

myfile << "\n";

}

GetID(19);

Memory(sizeof(vct) \* pow(n, 2), myfile.tellp());

myfile.close();

}

bool LoadMatrix(vector < vector <double> > & vct, int & n)

{

ifstream myfile;

myfile.open("MyFile.csv");

string element, raw, column;

int iRaw, iCol;

double dElement;

int k;

try

{

if (myfile.is\_open())

{

getline(myfile, element, '\n');

if (element.empty())

{

GetID(21);

return false;

}

k = stoi(element);

if (k != n)

{

vct.resize(k);

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

vct[i].resize(k);

if (k > n)

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

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

{

if (vct[i][j] == NULL)

vct[i][j] = 0;

}

n = k;

}

while (getline(myfile, element, ';'))

{

getline(myfile, raw, ';');

getline(myfile, column, '\n');

iRaw = stoi(raw);

iCol = stoi(column);

dElement = stod(element);

vct[iRaw][iCol] = dElement;

}

GetID(20);

myfile.close();

}

else

{

GetID(22);

return false;

}

}

catch (invalid\_argument)

{

GetID(17);

return false;

}

catch (out\_of\_range)

{

GetID(18);

return false;

}

return true;

}

void MatrixMain()

{

int n;

GetID(12);

InputSize(n);

vector < vector <double> > vct(n, vector<double>(n));

char ch = '1';

GetID(13);

InputMainMatrix(vct, n);

do

{

GetID(14);

InputCharacter(ch);

switch (ch)

{

case '1':

AdditionMatrix(vct, n);

break;

case '2':

SubtractionMatrix(vct, n);

break;

case '3':

MultiplicationMatrix(vct, n);

break;

case '4':

DivisionMatrix(vct, n);

break;

case '5':

SaveMatrix(vct, n);

break;

case '6':

if (!LoadMatrix(vct, n))

return;

break;

case '7':

OutputMatrix(vct, n);

break;

}

} while (ch >= 49 && ch <= 55);

}

// Файл Interface.h

#pragma once

#include <iostream>

#include <string>

#include <vector>

using namespace std;

void GetID(int id)

{

switch (id)

{

case 1:

cout << "Введiть '1' для роботи з калькулятором,\n'2' для роботи з матрицями,\n'0' для завершення програми.\n";

break;

case 2:

cout << "Введiть вираз: ";

break;

case 3:

cout << "Вираз повинен закiнчуватися символом '='!\n\n";

break;

case 4:

cout << "Кiлькiсть операторiв та операндiв не спiвпадають!\n\n";;

break;

case 5:

cout << "Недопустимий символ!\n\n";

break;

case 6:

cout << "Кiлькiсть вiдкритих дужок не вiдповiдає кiлькостi закритих!\n\n";

break;

case 7:

cout << "Не всi числа є простими!\n\n";

break;

case 8:

cout << "Результат: ";

break;

case 9:

cout << "Вираз не повинен починатися/закiнчуватися оператором!\n\n";

break;

case 10:

cout << "Знак '=' в недопустимому мiсцi\n\n";

break;

case 11:

cout << "Дужки пустi!\n\n";

break;

case 12:

cout << "\nВведiть розмiр матрицi: ";

break;

case 13:

cout << "Введiть елементи матрицi:\n";

break;

case 14:

cout << "\n\nВведiть '1' для додавання матриць";

cout << "\n'2' для вiднiмання матриць\n'3' для множення матриць";

cout << "\n'4' для дiлення матриць\n'5' для зберiгання матрицi\n'6' для зчитування матрицi";

cout << "\n'7' для виводу матрицi\n'0' для виходу\n";

break;

case 15:

cout << "\nВведiть елементи другої матрицi:\n";

break;

case 16:

cout << "Файл не може бути вiдкритим або створеним\n";

break;

case 17:

cout << "Неможливо виконати приведення типiв!\n\n";

break;

case 18:

cout << "Вихiд за границю дiапазона типу даних!\n\n";

break;

case 19:

cout << "\nФайл збережено!\n";

break;

case 20:

cout << "\nФайл зчитано!\n";

break;

case 21:

cout << "\nЗчитування неможливе, оскiльки файл пустий!\n\n";

break;

case 22:

cout << "\nПомилка вiдкривання файла!\n\n";

break;

default:

break;

}

}

void InputString(string & str)

{

getline(cin, str);

}

void GetResult(double result)

{

cout << result << endl << endl;

}

void InputSize(int & n)

{

cin >> n;

}

void InputMatrixElement (double & element,int i, int j)

{

cout << "Element [" << i << ", " << j << "] = ";

cin >> element;

}

void OutputMatrixElement(double element, int i, int j)

{

cout << "Element [" << i << ", " << j << "] = ";

cout << element << endl;

}

void InputCharacter(char & ch)

{

cin >> ch;

cin.ignore();

}

void Memory(int memory1, int memory2)

{

cout << "Пам'ять, яку займає вектор - " << memory1 << " байт(а)." << endl;

cout << "Пам'ять, яку займає файл - " << memory2 << " байт(а)." << endl;

}

Результати комп'ютерної реалізації:

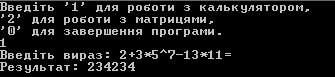


Рис.1. Приклад роботи з калькулятором.

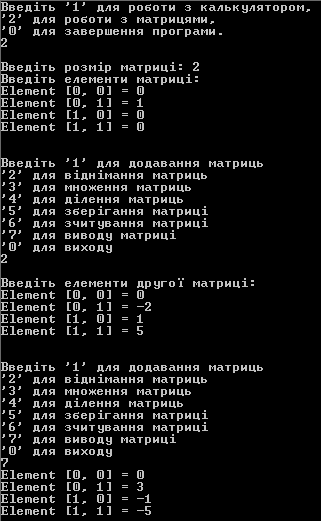


Рис.2. Приклад роботи з матрицями.

**Висновок:** розроблена програма призначена для обрахунків арифметичних виразів з простими числами та дій над матрицями і векторами. Обмеження програми: розмір числа не повинен перевищувати розмір відповідного типу даних, відсутність математичних функцій (синус, експонента). Можливим варіантом вдосконалення програми є створення графічного інтерфейсу користувача. Під час виконання роботи, я навчився працювати з такими структурами даних як стек та вектор.