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

*Алгоритм для* *перетворення звичайного запису в бездужковий*:

Зчитуємо символ поки не закінчаться і для кожного робимо наступне:

* Якщо **Ai** є числом то виводимо його;
* Якщо символ **Ai** є функцією завантажити його в стек;
* Якщо символ **Ai='('** то поміщаємо його в стек;
* Якщо символ **Ai=')'** то:
* До тих пір, поки верхнім елементом стека не стане відкриваюча дужка, виштовхуємо елементи з стека у вихідний рядок. При цьому відкриваюча дужка видаляється з стека, але у вихідну рядок не додається. Якщо після цього кроку на вершині стека виявляється символ функції, виштовхуємо його у вихідний рядок. Якщо стек закінчився раніше, ніж ми зустріли відкриває дужку, це означає, що у виразі або невірно поставлений розділовий знак, або не узгодженні дужки.
* Якщо символ **Ai** є оператором то:

1. поки …

… (Якщо оператор асоційований, або ліво-асоційований) пріоритет оператора менше або дорівнює пріоритету оператора, що знаходиться на вершині стека … … (Якщо оператор право-асоціювання) пріоритет оператора менше пріоритету оператора, що знаходиться на вершині стека … … Виштовхуємо верхні елементи стека у вихідний рядок; 2. Поміщаємо оператор в стек.

*Алгоритм для обчислення значення виразу:*

Для всіх символів робимо наступне:

* Якщо **Аі** число, то вкласти його у стек;
* Якщо **Аі** оператор, то:
  + - Витягуємо із стеку два числа;
    - Виконуємо дію із числами і результат вкладаємо в стек;
* Якщо **Аі** є функцією то:
  + - Витягуємо із стеку одне число;
    - Визначаємо значення функції із відповідним аргументом та поміщаємо результат у стек;
* В кінці роботи в стеку знаходитиметься результат виразу.

Текст програми на мові С++:

// Файл Lab01.cpp

#include <iostream>

#include "stdafx.h"

#include "Calculator.h"

#include "Matrix.h"

#include "UInterface.h"

using namespace std;

void goToMatrix(UInterface myInterface)

{

int sizeMatrix;

myInterface.getID(12);

myInterface.inputSize(sizeMatrix);

Matrix myMatrix(sizeMatrix);

myMatrix.matrixMain(myMatrix);

}

void main()

{

UInterface myInterface;

Calculator myCalculator;

setlocale(LC\_ALL, "ukr");

char character;

do

{

myInterface.getID(1);

myInterface.inputCharacter(character);

switch (character)

{

case '1':

myCalculator.calculatorMain();

break;

case '2':

goToMatrix(myInterface);

break;

default:

break;

}

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

system("pause");

}

// Файл Calculator.h

#pragma once

#include <iostream>

#include <string>

#include <stack>

#include "UInterface.h"

using namespace std;

class Calculator

{

private:

static bool isPrimeNumber(int number, UInterface myInterface)

{

if (number < 2)

{

myInterface.getID(7);

return false;

}

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

if (number % i == 0)

{

myInterface.getID(7);

return false;

}

return true;

}

static int priority(char operation)

{

switch (operation)

{

case '+':

case '-': return 1;

case '\*':

case '/': return 2;

case '^': return 3;

default: system("pause");

}

}

static 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();

}

static bool checkStr(string str, int sizestr, UInterface myInterface)

{

int openBrackets = 0, closedBrackets = 0;

int operands = -1, operators = 0;

bool p = true;

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

{

myInterface.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)

{

myInterface.getID(9);

return false;

}

break;

case '(':

if (i > 0)

{

if (isdigit(str.at(i - 1)) || !isdigit(str.at(i + 1)))

{

myInterface.getID(23);

return false;

}

}

openBrackets++;

break;

case ')':

if (i == 1)

{

myInterface.getID(25);

return false;

}

closedBrackets++;

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

{

myInterface.getID(11);

return false;

}

if (!isdigit(str.at(i - 1)))

{

myInterface.getID(24);

return false;

}

break;

case '=':

if (i != sizestr - 1)

{

myInterface.getID(10);

return false;

}

break;

default:

myInterface.getID(5);

return false;

break;

}

}

if (operands != operators)

{

myInterface.getID(4);

return false;

}

if (openBrackets != closedBrackets)

{

myInterface.getID(6);

return false;

}

return true;

}

static bool pushNumberToStack(stack <double>& stk1, string& tmp, UInterface myInterface)

{

try

{

stk1.push(stod(tmp));

if (!isPrimeNumber(stoi(tmp), myInterface))

return false;

tmp = "\0";

}

catch (invalid\_argument)

{

myInterface.getID(17);

return false;

}

catch (out\_of\_range)

{

myInterface.getID(18);

return false;

}

return true;

}

public:

static void calculatorMain()

{

string str;

string polandForm = "\0";

stack <double> stk1;

stack <char> stk2;

string tmp = "\0";

int dynamicBrackets = 0;

unsigned int sizeStr, sizeStk2;

UInterface myInterface;

myInterface.getID(2);

myInterface.inputString(str);

sizeStr = str.size();

if (!checkStr(str, sizeStr, myInterface))

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())

{

polandForm += tmp;

polandForm += " ";

if (!pushNumberToStack(stk1, tmp, myInterface))

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)))

{

polandForm += stk2.top();

polandForm += " ";

calculation(stk1, stk2);

}

}

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

}

break;

case '(':

dynamicBrackets++;

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

break;

case ')':

if (!tmp.empty())

{

polandForm += tmp;

polandForm += " ";

if (!pushNumberToStack(stk1, tmp, myInterface))

return;

}

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

{

polandForm += stk2.top();

polandForm += " ";

calculation(stk1, stk2);

}

stk2.pop();

dynamicBrackets--;

break;

case '=':

if (!tmp.empty())

{

polandForm += tmp;

polandForm += " ";

if (!pushNumberToStack(stk1, tmp, myInterface))

return;

}

break;

default:

break;

}

}

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

{

polandForm += stk2.top();

polandForm += " ";

calculation(stk1, stk2);

}

myInterface.getPolandForm(polandForm);

myInterface.getID(8);

myInterface.getResult(stk1.top());

}

};

// Файл Matrix.h

#pragma once

#include <iostream>

#include <fstream>

#include <vector>

#include <string>

#include <ctime>

#include "UInterface.h"

using namespace std;

class Matrix

{

private:

int size;

vector<vector<double> > vct;

static void inputMainMatrix(Matrix &myMatrix, UInterface myInterface)

{

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

for (int j = 0; j < myMatrix.size; j++)

if (j > i)

myInterface.inputElement(myMatrix.vct[i][j], i, j);

else

{

myMatrix.vct[i][j] = 0;

myInterface.outputElement(myMatrix.vct[i][j], i, j);

}

}

static void inputAdditionalMatrix(Matrix& myAdditionalMatrix, UInterface myInterface)

{

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

for (int j = 0; j < myAdditionalMatrix.size; j++)

myInterface.inputElement(myAdditionalMatrix.vct[i][j], i, j);

}

static void outputMatrix(Matrix myMatrix, UInterface myInterface)

{

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

for (int j = 0; j < myMatrix.size; j++)

myInterface.outputElement(myMatrix.vct[i][j], i, j);

}

static void additionMatrix(Matrix& myMatrix, UInterface myInterface)

{

Matrix tmp(myMatrix.size);

myInterface.getID(15);

inputAdditionalMatrix(tmp, myInterface);

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

for (int j = 0; j < myMatrix.size; j++)

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

}

static void subtractionMatrix(Matrix& myMatrix, UInterface myInterface)

{

Matrix tmp(myMatrix.size);

myInterface.getID(15);

inputAdditionalMatrix(tmp, myInterface);

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

for (int j = 0; j < myMatrix.size; j++)

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

}

static void multiplicationMatrix(Matrix& myMatrix, UInterface myInterface)

{

Matrix tmp1(myMatrix.size), tmp2(myMatrix.size);

myInterface.getID(15);

inputAdditionalMatrix(tmp2, myInterface);

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

for (int j = 0; j < myMatrix.size; j++)

{

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

myMatrix.vct[i][j] = 0;

}

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

for (int j = 0; j < myMatrix.size; j++)

for (int k = 0; k < myMatrix.size; k++)

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

}

static void divisionMatrix(Matrix& myMatrix, UInterface myInterface)

{

Matrix tmp1(myMatrix.size), tmp2(myMatrix.size), E(myMatrix.size);

double dtmp;

myInterface.getID(15);

inputAdditionalMatrix(tmp2, myInterface);

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

for (int j = 0; j < myMatrix.size; j++)

{

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

myMatrix.vct[i][j] = 0;

if (i == j)

E.vct[i][j] = 1;

else

E.vct[i][j] = 0;

}

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

{

dtmp = tmp2.vct[i][i];

for (int j = 0; j < myMatrix.size; j++)

{

tmp2.vct[i][j] /= dtmp;

E.vct[i][j] /= dtmp;

}

for (int k = i + 1; k < myMatrix.size; k++)

{

dtmp = tmp2.vct[k][i];

for (int j = 0; j < myMatrix.size; j++)

{

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

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

}

}

}

for (int i = myMatrix.size - 1; i > 0; i--)

{

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

{

dtmp = tmp2.vct[k][i];

for (int j = 0; j < myMatrix.size; j++)

{

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

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

}

}

}

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

for (int j = 0; j < myMatrix.size; j++)

for (int k = 0; k < myMatrix.size; k++)

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

}

static void saveMatrix(Matrix myMatrix, UInterface myInterface)

{

unsigned int start = clock(), finish;

ofstream myfile;

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

if (!myfile.is\_open())

{

myInterface.getID(16);

return;

}

myfile << myMatrix.size;

myfile << "\n";

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

for (int j = 0; j < myMatrix.size; j++)

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

{

myfile << myMatrix.vct[i][j];

myfile << ';';

myfile << i;

myfile << ';';

myfile << j;

myfile << "\n";

}

myInterface.getID(19);

myInterface.memory(sizeof(vector<double>) \* pow(myMatrix.size, 2), myfile.tellp());

myfile.close();

finish = clock();

myInterface.executionTime(finish - start);

}

static bool loadMatrix(Matrix& myMatrix, UInterface myInterface)

{

unsigned int start = clock(), finish;

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())

{

myInterface.getID(21);

return false;

}

k = stoi(element);

if (k != myMatrix.size)

{

myMatrix.vct.resize(k);

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

myMatrix.vct[i].resize(k);

if (k > myMatrix.size)

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

for (int j = 0; j < myMatrix.size; j++)

{

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

myMatrix.vct[i][j] = 0;

}

myMatrix.size = k;

}

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

{

getline(myfile, raw, ';');

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

iRaw = stoi(raw);

iCol = stoi(column);

dElement = stod(element);

myMatrix.vct[iRaw][iCol] = dElement;

}

myInterface.getID(20);

myfile.close();

}

else

{

myInterface.getID(22);

return false;

}

}

catch (invalid\_argument)

{

myInterface.getID(17);

return false;

}

catch (out\_of\_range)

{

myInterface.getID(18);

return false;

}

finish = clock();

myInterface.executionTime(finish - start);

return true;

}

public:

Matrix(int sizeMatrix)

{

size = sizeMatrix;

vct.resize(size);

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

vct[i].resize(size);

}

static void matrixMain(Matrix myMatrix)

{

UInterface myInterface;

char ch = '1';

myInterface.getID(13);

inputMainMatrix(myMatrix, myInterface);

do

{

myInterface.getID(14);

myInterface.inputCharacter(ch);

switch (ch)

{

case '1':

additionMatrix(myMatrix, myInterface);

break;

case '2':

subtractionMatrix(myMatrix, myInterface);

break;

case '3':

multiplicationMatrix(myMatrix, myInterface);

break;

case '4':

divisionMatrix(myMatrix, myInterface);

break;

case '5':

saveMatrix(myMatrix, myInterface);

break;

case '6':

if (!loadMatrix(myMatrix, myInterface))

return;

break;

case '7':

outputMatrix(myMatrix, myInterface);

break;

}

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

}

};

// Файл UInterface.h

#pragma once

#include <iostream>

#include <string>

#include <vector>

using namespace std;

class UInterface

{

public:

void getID(int id)

{

switch (id)

{

case 1:

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

cout << "\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;

case 23:

cout << "\nПеред дужкою немає оператора або пiсля вiдкритої дужки немає операнда!\n\n";

break;

case 24:

cout << "\nПеред закритою дужкою не може бути оператор!\n\n";

break;

case 25:

cout << "\nВираз не може починатись закритою дужкою!\n\n";

break;

default:

break;

}

}

static void inputString(string& str)

{

getline(cin, str);

}

static void getResult(double result)

{

cout << result << endl << endl;

}

static void inputSize(int& size)

{

cin >> size;

}

static void inputElement(double& element, int line, int column)

{

cout << "Element [" << line << ", " << column << "] = ";

cin >> element;

}

static void outputElement(double element, int line, int column)

{

cout << "Element [" << line << ", " << column << "] = ";

cout << element << endl;

}

static void inputCharacter(char& character)

{

cin >> character;

cin.ignore();

}

static 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;

}

static void executionTime(unsigned int time)

{

cout << "Час виконання функцiї: " << time << " мс" << endl;

}

static void getPolandForm(string polandForm)

{

cout << "Польська форма: " << polandForm << endl;

}

};

Результати комп'ютерної реалізації:

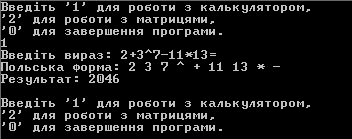


Рис.1. Приклад роботи з калькулятором.

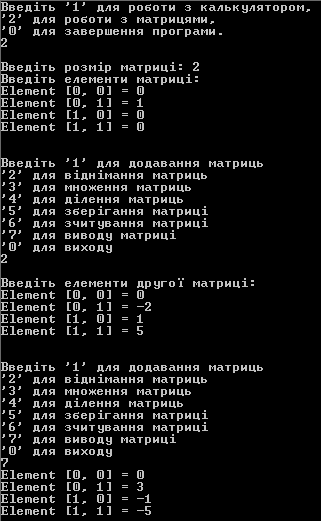


Рис.2. Приклад роботи з матрицями.

**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.