**Мета роботи** – одержати практичні навички роботи з класами та методами бібліотеки valarray різних динамічних структур даних.

**Постановка завдання:**

15. Робота з valarray

Докладно опишіть класи та методи бібліотеки valarray.

а) Реалізуйте задачу на Си за допомогою виділення пам’яті вказівником

б) Реалізуйте задачу за допомогою визначення типів, використовуючи valarray

в) Реалізуйте задачу за допомогою контейнеру vector.

За допомогою бібліотеки псевдоконтейнера valarray реалізувати роботу зі структурою МАТРИЦЯ (див. проект Smatrix).

Порівняти швидкодії програми з реалізаціями, що виконані на базі вектору та звичайного масиву.

За допомогою бібліотеки псевдоконтейнера valarray реалізувати роботу конволюції

**Текстове обґрунтування вибору структури пам’яті і його інтерфейсу.**

Було створено клас Matrixs в якому у privat частині реалізовано наступне

double \*\*M1;

valarray<valarray<double>> M2;

vector<vector<double>> M3;

int row;

int col;

Де M1 – матриця за допомогою виділення пам’яті вказівником

М2 – матриця реалізована за допомогою valarray

M3 – матриця реалізована за допомогою стандартного класу vector

та двох параметрів row та col, які відповідають за кількість строк в матриці та кількості стовпців в матриці.

А також у public частині було реалізовано наступні методи:

//конструктор за замовчуванням

Matrixs();

//матриця із вказівником

void UkazAdd();//заповнення вручну

void UkazFile();//заповнення з файлу

void UkazPrint();//виведення матриці на консоль

//матриця із valarray

void ValarrayAdd();//заповнення вручну

void ValarrayFile();//заповнення з файлу

void ValarrayPrint();//виведення матриці на консоль

//матриця із vector

void VectorAdd();//заповнення вручну

void VectorFile();//заповнення з файлу

void VectorPrint();//виведення матриці на консоль

//Конволюція матриць

void ConvolutionM1();

void ConvolutionM2();

void ConvolutionM3();

//Деструктор

~Matrixs();

**Опис розроблених методів**

Matrixs() – звичайний конструктор без параметрів. Присвоює початкові параметри row=0 та col=0

UkazAdd() – заповнення масиву в ручну за допомогою виділення пам’яті вказівником

UkazFile() – заповнення масиву з файлу за допомогою виділення пам’яті вказівником

UkazPrint() – вивід на консоль отриманого масиву

ValarrayAdd() – заповнення масиву в ручну за допомогою масиву valarray

UkazFile() – заповнення масиву з файлу за допомогою масиву valarray

UkazPrint() – вивід на консоль отриманого масиву

VectorAdd() – заповнення масиву в ручну за допомогою масиву vector

VectorFile() – заповнення масиву з файлу за допомогою масиву vector

VectorPrint() – вивід на консоль отриманого масиву

ConvolutionM1() – конволюція матриці створеною за допомогою виділення пам’яті вказівником

ConvolutionM2() – конволюція матриці створеною за допомогою масиву valarray

ConvolutionM3() – конволюція матриці створеною за допомогою масиву vector

~Matrixs() – деструктор класу

**Блок схема**

Початок програми

Меню програми

Введення вибору меню

1. Заповнення масиву вручну за допомогою виділення пам’яті вказівником

2. Заповнення масиву з файлу за допомогою виділення пам’яті вказівником

3. Заповнення масиву вручну за допомогою визначення типів, використовуючи valarray

4. Заповнення масиву з файлу за допомогою визначення типів, використовуючи valarray

5. Заповнення масиву вручну за допомогою контейнеру vector

6. Заповнення масиву з файлу за допомогою контейнеру vector

7. Виведення масиву на консоль за допомогою виділення пам’яті вказівником

8. Виведення масиву на консоль за допомогою визначення типів, використовуючи valarray

9. Виведення масиву на консоль за допомогою контейнеру vector

10. Конволюція масиву за допомогою виділення пам’яті вказівником

11. Конволюція масиву за допомогою визначення типів, використовуючи valarray

12. Конволюція масиву за допомогою контейнеру vector

0. Вихід

Вивід відповідних результатів

Завершення програми

**Лістинг програми**

***valarray.cpp***

//Підключення бібліотек

#include <iostream>

#include "Matrixs.h"

#include <Windows.h>

//Головна функція

int main()

{

//підключення української мови до консолі

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

//Екземпляр класу Matrixs

Matrixs M;

//змінна num

int num=0;

do {

//Меню програми

cout << "Виберіть пункт меню" << endl;

cout << "1. Заповнення масиву вручну за допомогою виділення памяті вказівником" << endl;

cout << "2. Заповнення масиву з файлу за допомогою виділення памяті вказівником" << endl;

cout << "3. Заповнення масиву вручну за допомогою визначення типів, використовуючи valarray" << endl;

cout << "4. Заповнення масиву з файлу за допомогою визначення типів, використовуючи valarray" << endl;

cout << "5. Заповнення масиву вручну за допомогою контейнеру vector" << endl;

cout << "6. Заповнення масиву з файлу за допомогою контейнеру vector" << endl;

cout << "7. Виведення масиву на консоль за допомогою виділення памяті вказівником" << endl;

cout << "8. Виведення масиву на консоль за допомогою визначення типів, використовуючи valarray" << endl;

cout << "9. Виведення масиву на консоль за допомогою контейнеру vector" << endl;

cout << "10. Конволюція масиву за допомогою виділення памяті вказівником" << endl;

cout << "11. Конволюція масиву за допомогою визначення типів, використовуючи valarray" << endl;

cout << "12. Конволюція масиву за допомогою контейнеру vector" << endl;

cout << "0. Вихід" << endl;

cin >> num; //вибір пункта меню

switch (num)

{

case 1: {

clock\_t start = clock();

M.UkazAdd();

clock\_t end = clock();

double seconds = (double)(end - start) / CLOCKS\_PER\_SEC;

cout << "Затрачений час: " << seconds << " секунд" << endl;

break;

}

case 2: {

clock\_t start = clock();

M.UkazFile();

clock\_t end = clock();

double seconds = (double)(end - start) / CLOCKS\_PER\_SEC;

cout << "Затрачений час: " << seconds << " секунд" << endl;

break;

}

case 3: {

clock\_t start = clock();

M.ValarrayAdd();

clock\_t end = clock();

double seconds = (double)(end - start) / CLOCKS\_PER\_SEC;

cout << "Затрачений час: " << seconds << " секунд" << endl;

break;

}

case 4: {

clock\_t start = clock();

M.ValarrayFile();

clock\_t end = clock();

double seconds = (double)(end - start) / CLOCKS\_PER\_SEC;

cout << "Затрачений час: " << seconds << " секунд" << endl;

break;

}

case 5: {

clock\_t start = clock();

M.VectorAdd();

clock\_t end = clock();

double seconds = (double)(end - start) / CLOCKS\_PER\_SEC;

cout << "Затрачений час: " << seconds << " секунд" << endl;

break;

}

case 6: {

clock\_t start = clock();

M.VectorFile();

clock\_t end = clock();

double seconds = (double)(end - start) / CLOCKS\_PER\_SEC;

cout << "Затрачений час: " << seconds << " секунд" << endl;

break;

}

case 7: {

clock\_t start = clock();

M.UkazPrint();

clock\_t end = clock();

double seconds = (double)(end - start) / CLOCKS\_PER\_SEC;

cout << "Затрачений час: " << seconds << " секунд" << endl;

break;

}

case 8: {

clock\_t start = clock();

M.ValarrayPrint();

clock\_t end = clock();

double seconds = (double)(end - start) / CLOCKS\_PER\_SEC;

cout << "Затрачений час: " << seconds << " секунд" << endl;

break;

}

case 9: {

clock\_t start = clock();

M.VectorPrint();

clock\_t end = clock();

double seconds = (double)(end - start) / CLOCKS\_PER\_SEC;

cout << "Затрачений час: " << seconds << " секунд" << endl;

break;

}

case 10: {

clock\_t start = clock();

M.ConvolutionM1();

M.UkazPrint();

clock\_t end = clock();

double seconds = (double)(end - start) / CLOCKS\_PER\_SEC;

cout << "Затрачений час: " << seconds << " секунд" << endl;

break;

}

case 11: {

clock\_t start = clock();

M.ConvolutionM2();

M.ValarrayPrint();

clock\_t end = clock();

double seconds = (double)(end - start) / CLOCKS\_PER\_SEC;

cout << "Затрачений час: " << seconds << " секунд" << endl;

break;

}

case 12: {

clock\_t start = clock();

M.ConvolutionM3();

M.VectorPrint();

clock\_t end = clock();

double seconds = (double)(end - start) / CLOCKS\_PER\_SEC;

cout << "Затрачений час: " << seconds << " секунд" << endl;

break;

}

default:

break;

}

} while (num != 0);

return 0;

}

***Matrixs.h***

#pragma once

#include <vector>

#include <valarray>

//використання простору імен std

using namespace std;

//клас Matrixs

class Matrixs

{

public:

//конструктор за замовчуванням

Matrixs();

//матриця із вказівником

void UkazAdd();//заповнення вручну

void UkazFile();//заповнення з файлу

void UkazPrint();//виведення матриці на консоль

//матриця із valarray

void ValarrayAdd();//заповнення вручну

void ValarrayFile();//заповнення з файлу

void ValarrayPrint();//виведення матриці на консоль

//матриця із vector

void VectorAdd();//заповнення вручну

void VectorFile();//заповнення з файлу

void VectorPrint();//виведення матриці на консоль

//Конволюція матриць

void ConvolutionM1();

void ConvolutionM2();

void ConvolutionM3();

//Деструктор

~Matrixs();

private:

double \*\*M1;

valarray<valarray<double>> M2;

vector<vector<double>> M3;

int row;

int col;

};

***Matrixs.cpp***

//Підключення бібліотек

#include "Matrixs.h"

#include <iostream>

#include <string>

#include <fstream>

//використання простору імен std

using namespace std;

//Конструктор

Matrixs::Matrixs()

{

col = 0;

row = 0;

}

//Заповнення масиву вручну за допомогою вказівника

void Matrixs::UkazAdd()

{

//введення розміру масиву

cout << "Введіть розміри матриці (row, col)" << endl;

cin >> row >> col;

//виділення пам'яті під масив

M1= new double\*[row];

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

M1[i] = new double[col];

//заповнення масиву

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

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

{

//введення значення

double num;

cout << "Введіть значення матріці [" << i << "," << j << "]" << endl;

cin >> num;

M1[i][j] = num;

}

}

//заповнення масиву з файлу

void Matrixs::UkazFile()

{

string line;

string file\_name;

//введення назви файлу

cout << "Введіть ім'я файлу" << endl;

cin >> file\_name;

std::ifstream in(file\_name);

int countcol = 0;

int countrow = 0;

if (in.is\_open())

{

while (getline(in, line))//построкове зчитування файлу

{

vector<double> tmp;//тимчасовий вектор

string word = "";

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

{

if (line[i] != ' ') //розбиття строки на слова

word += line[i];

else {

countcol++;

word = "";

}

}

if (word != "")

countcol++;

countrow++;

}

}

in.close();

row = countrow;

col = countcol / countrow;

countcol = 0;

countrow = 0;

//виділення пам'яті під масив

M1 = new double\* [row];

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

M1[i] = new double[col];

std::ifstream in2(file\_name);

if (in2.is\_open())

{

while (getline(in2, line))//построкове зчитування файлу

{

string word = "";

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

{

if (line[i] != ' ') //розбиття строки на слова

word += line[i];

else {

M1[countrow][countcol % col] = stod(word);

countcol++;

word = "";

}

}

if (word != "")

{

M1[countrow][countcol % col] = stod(word);

countcol++;

}

countrow++;

}

}

in2.close();

}

//виведення масиву на консоль

void Matrixs::UkazPrint()

{

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

{

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

cout << M1[i][j] << "\t";

cout << endl;

}

}

//заповнення масиву вручну за допомогою valarray;

void Matrixs::ValarrayAdd()

{

cout << "Введіть розміри матриці (row, col)" << endl;

cin >> row >> col;

M2.resize(row);

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

valarray<double> tmp;

tmp.resize(col);

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

{

double num;

cout << "Введіть значення матріці [" << i << "," << j << "]" << endl;

cin >> num;

tmp[j]=num;

}

M2[i]=tmp;

}

}

//заповнення масиву з файлу

void Matrixs::ValarrayFile()

{

string line;

string file\_name;

//введення назви файлу

cout << "Введіть ім'я файлу" << endl;

cin >> file\_name;

std::ifstream in(file\_name);

int countcol = 0;

int countrow = 0;

if (in.is\_open())

{

while (getline(in, line))//построкове зчитування файлу

{

vector<double> tmp;//тимчасовий вектор

string word = "";

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

{

if (line[i] != ' ') //розбиття строки на слова

word += line[i];

else {

countcol++;

word = "";

}

}

if (word != "")

countcol++;

countrow++;

}

}

in.close();

row = countrow;

col = countcol / countrow;

countcol = 0;

countrow = 0;

M2.resize(row);

std::ifstream in2(file\_name);

if (in2.is\_open())

{

while (getline(in2, line))//построкове зчитування файлу

{

valarray<double> tmp;

tmp.resize(col);

string word = "";

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

{

if (line[i] != ' ') //розбиття строки на слова

word += line[i];

else {

tmp[countcol % col] = stod(word);

countcol++;

word = "";

}

}

if (word != "")

{

tmp[countcol % col] = stod(word);

countcol++;

}

M2[countrow] = tmp;

countrow++;

}

}

in2.close();

}

//Вивід масиву на екран

void Matrixs::ValarrayPrint()

{

for (int i = 0; i < M2.size(); ++i)

{

for (int j = 0; j < M2[i].size(); ++j)

cout << M2[i][j] << "\t";

cout << endl;

}

}

//заповнення масиву вручну

void Matrixs::VectorAdd()

{

//Введення розмірів масиву

cout << "Введіть розміри матриці (row, col)" << endl;

cin >> row >> col;

//заповнення масиву

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

vector<double> tmp; //тимчасовий вектор

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

{

double num;

cout << "Введіть значення матріці [" << i << "," << j << "]" << endl;

cin >> num;

tmp.push\_back(num);

}

M3.push\_back(tmp);

}

}

//заповнення масиву з файлу

void Matrixs::VectorFile()

{

string line;

string file\_name;

//введення назви файлу

cout << "Введіть ім'я файлу" << endl;

cin >> file\_name;

std::ifstream in(file\_name);

if (in.is\_open())

{

while (getline(in, line))//построкове зчитування файлу

{

vector<double> tmp;//тимчасовий вектор

string word = "";

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

{

if (line[i] != ' ') //розбиття строки на слова

word += line[i];

else {

tmp.push\_back(stod(word));

word = "";

}

}

if (word != "")

tmp.push\_back(stod(word));

M3.push\_back(tmp);

}

}

in.close();

row = M3.size();

col = M3[0].size();

}

//Вивід масиву на екран

void Matrixs::VectorPrint()

{

for (int i = 0; i < M3.size(); ++i)

{

for (int j = 0; j < M3[i].size(); ++j)

cout << M3[i][j] << "\t";

cout << endl;

}

}

//конволюція матриці М1

void Matrixs::ConvolutionM1()

{

double MC[3][3] = { {0, 1, 0},

{0,-1, 0},

{0, 0, 0} };

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

{

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

{

int i0 = (i - 1 + row) % row;

int j0 = (j - 1 + col) % col;

int i1 = (i + row) % row;

int j1 = (j + col) % col;

int i2 = (i + 1 + row) % row;

int j2 = (j + 1 + col) % col;

M1[i][j] = M1[i0][j0] \* MC[0][0] + M1[i0][j1] \* MC[0][1] + M1[i0][j2] \* MC[0][2] +

M1[i1][j0] \* MC[1][0] + M1[i1][j1] \* MC[1][1] + M1[i1][j2] \* MC[1][2] +

M1[i2][j0] \* MC[2][0] + M1[i2][j1] \* MC[2][1] + M1[i2][j2] \* MC[2][2];

}

}

}

//конволюція матриці М2

void Matrixs::ConvolutionM2()

{

valarray<valarray<double>> MC = {{0, 1, 0},

{0,-1, 0},

{0, 0, 0}};

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

{

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

{

int i0 = (i - 1 + row) % row;

int j0 = (j - 1 + col) % col;

int i1 = (i + row) % row;

int j1 = (j + col) % col;

int i2 = (i + 1 + row) % row;

int j2 = (j + 1 + col) % col;

M2[i][j] = M2[i0][j0] \* MC[0][0] + M2[i0][j1] \* MC[0][1] + M2[i0][j2] \* MC[0][2] +

M2[i1][j0] \* MC[1][0] + M2[i1][j1] \* MC[1][1] + M2[i1][j2] \* MC[1][2] +

M2[i2][j0] \* MC[2][0] + M2[i2][j1] \* MC[2][1] + M2[i2][j2] \* MC[2][2];

}

}

}

//конволюція матриці М3

void Matrixs::ConvolutionM3()

{

vector<vector<double>> MC = {{0, 1, 0},

{0,-1, 0},

{0, 0, 0}};

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

{

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

{

int i0 = (i - 1 + row) % row;

int j0 = (j - 1 + col) % col;

int i1 = (i + row) % row;

int j1 = (j + col) % col;

int i2 = (i + 1 + row) % row;

int j2 = (j + 1 + col) % col;

M3[i][j] = M3[i0][j0] \* MC[0][0] + M3[i0][j1] \* MC[0][1] + M3[i0][j2] \* MC[0][2] +

M3[i1][j0] \* MC[1][0] + M3[i1][j1] \* MC[1][1] + M3[i1][j2] \* MC[1][2] +

M3[i2][j0] \* MC[2][0] + M3[i2][j1] \* MC[2][1] + M3[i2][j2] \* MC[2][2];

}

}

}

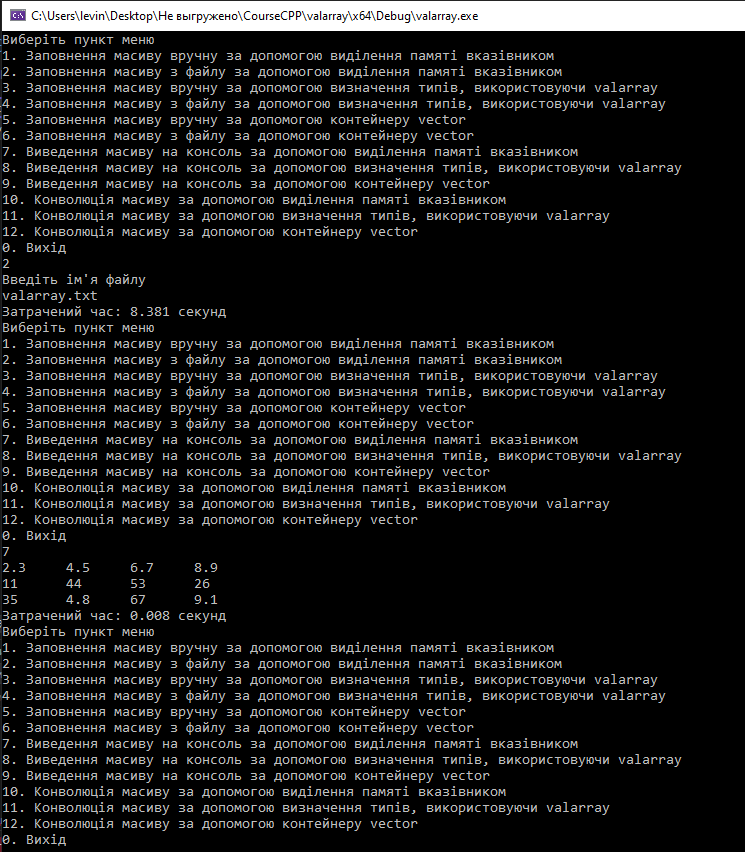
//Деструктор

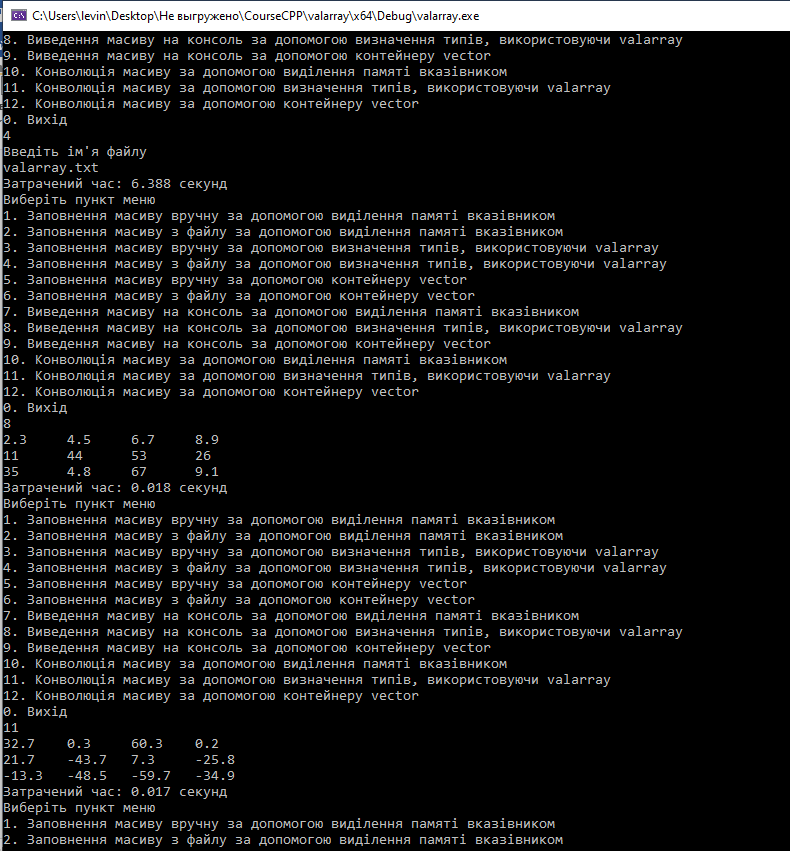
Matrixs::~Matrixs()

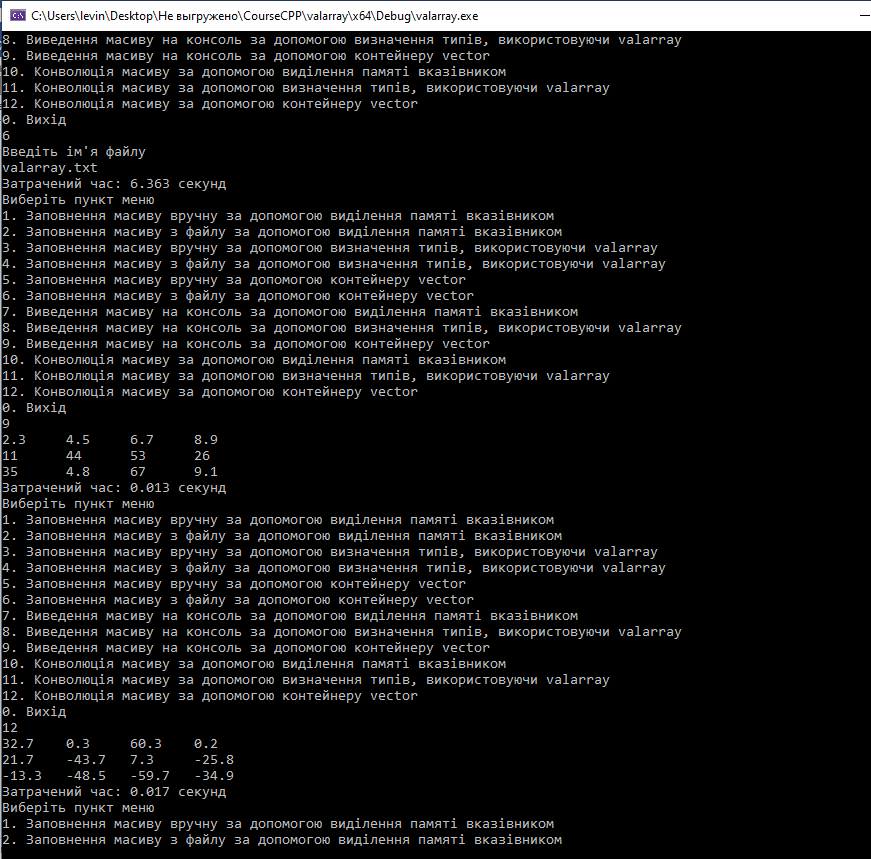
{

}

**Приклад роботи програми**







**Висновок**

Освоїв основи роботи з класами та векторами. Найшвидший метод обробки даних показав метод вказівок.