МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ



**Дніпровський національний університет  
залізничного транспорту імені академіка В. Лазаряна**

Кафедра «Комп’ютерні інформаційні технології»

**Звіт №3**

**з навчальної практики**

**на тему: «Динамічні структури даних»**

Виконав: студент гр. ПЗ2011

Кулик Сергій Вадимович

Дніпро

2021

**Тема:** Динамічні структури даних

**Задача:** Розробити та реалізувати структуру даних, що відображає роботу з розрідженою матрицею (не менше 70% елементів дорівнюють нулю).

**Складність E**

**Вимоги до змісту:** Всі операції повинні бути оформлені у вигляді окремої бібліотеки, що підключаються до основної програми.

**Вимоги до оформлення програми:** Тип оформлення інтерфейсу користувача текстовий (консольна програма) або графічний за бажанням.

**Опис методів:** У якості структури для зберігання матриці використовував одновимірний динамічний масив.

Графічне представлення одновимірного масиву

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| i | 0 | 1 | 2 | 3 | 4 | … | N |
| Arr[i] | 20 | 4 | 9 | 5 | 7 | … | 10 |

**Функціональні вимоги до програми:**

Необхідно реалізувати наступні операції:

* додавання елемента матриці
* заміна значення на інше
* видалення елемента матриці за заданими індексами та по значенню
* пошук елемента матриці за заданими індексами
* друк матриці у звичному форматі та форматі розрідженої матриці – індекси та значення
* Сформувати одновимірний масив з сум додатних елементів рядків матриці (Варіант 14).

**Зовнішні специфікації:**

Формат вхідних даних

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № з/п | Найменування даних | Умовне позначення | Вимоги до даних | Приклад |
| 1 | Кількість рядків матриці | rows | Натуральне число | 10 |
| 2 | Кількість стовбців матриці | columns | Натуральне число | 10 |

Формат вихідних даних

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № з/п | Найменування даних | Умовне позначення | Вимоги до даних | Приклад |
| 1 | Матриця | arr | Таблиця цілих чисел, розріджена матриця | 0 0 1 0  0 2 0 0  0 3 0 5 |
| 2 | Одновимірний масив для індивідуального завдання | mass | Натуральні числа | 0, 2, 0, 1, 5, 2, 3 |

**Алгоритм:**

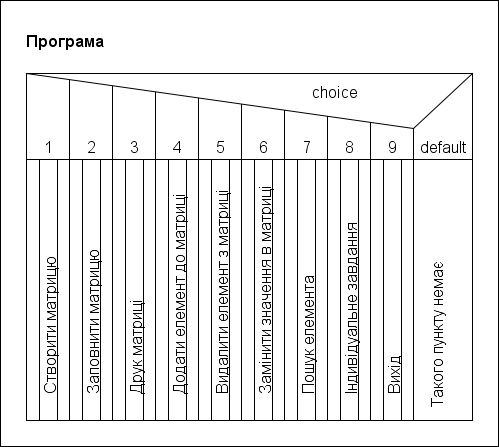


Рис. 1 – Алгоритм програми

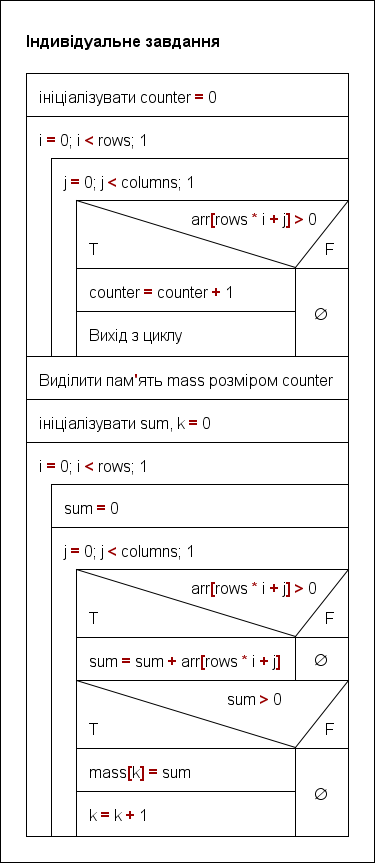


Рис. 2 – Алгоритм індивідуального завдання.

**Тести:**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Назва тесту | Вхідні дані | Результат |
| 1. | Матриця для тестування алгоритму індивідуального завдання | Матриця [3, 3]  1 1 1  0 0 0  -1 -1 -1 | Буде сформований масив на 1 елемент зі значенням 3. |
| 2. | Додавання елементу | Матриця [3, 3]  з тесту №1 | Нулбовий елемент замінится на число |
| 3. | Видалення елементу | Матриця [3, 3]  з тесту №1 | Ненульовий елемент стане нульовим |
| 4. | Заміна елементу | Матриця [3, 3]  з тесту №1 | Будь-який елемент може прийняти друге значення |
| 5. | Пошук елементу | Матриця [3, 3]  з тесту №1 | Вивести елемент за індексами |

**Контрольний приклад:**

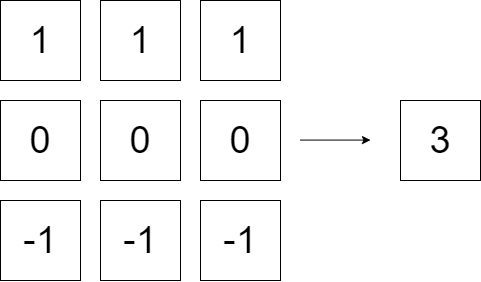
****

Рис. 3 – Контрольний приклад.

**Текст програми:**

Файл Source.cpp

#include <iostream>

#include<windows.h>

#include "Func.h"

using namespace std;

void menu()

{

cout << endl;

cout << "===================| МЕНЮ |=====================" << endl;

cout << "1. Створити матрицю." << endl;

cout << "2. Заповнити матрицю." << endl;

cout << "3. Друк матриці." << endl;

cout << "4. Додати елемент до матриці. " << endl;

cout << "5. Видалити елемент з матриці." << endl;

cout << "6. Замінити значення в матриці." << endl;

cout << "7. Пошук елемента." << endl;

cout << "8. Сформувати одновимірний масив з сум додатних елементів рядків матриці." << endl;

cout << "9. Вихід." << endl;

cout << "================================================" << endl;

cout << endl;

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

Matrix matrix;

bool exit = true;

while (exit)

{

menu();

int choise;

cin >> choise;

switch (choise)

{

case 1:

{

int a, b;

cout << "Введіть кількість рядків матриці: "; cin >> a;

cout << "Введіть кількість стовпців матриці: "; cin >> b;

matrix.rows = a;

matrix.columns = b;

matrix.create();

break;

}

case 2:

{

cout << "1. Заповнити вручну." << endl;

cout << "2. Заповнити генератором випадкових чисел." << endl;

cin >> choise;

switch (choise)

{

case 1:

{

matrix.fillManualMatrix();

break;

}

case 2:

{

matrix.fillRandomMatrix();

break;

}

default:

cout << "Такого пункту немає!" << endl;

break;

}

break;

}

case 3:

{

cout << "1. Друк у вигляді матриці" << endl;

cout << "2. Друк з індексами" << endl;

cin >> choise;

switch (choise)

{

case 1:

{

matrix.printArr();

break;

}

case 2:

{

matrix.printArrOfIndex();

break;

}

default:

cout << "Такого пункту немає!" << endl;

break;

}

break;

}

case 4:

{

matrix.addComponent();

break;

}

case 5:

{

cout << "1. Видалити компонент за індексами." << endl;

cout << "2. Видалити компонент за значенням." << endl;

cin >> choise;

switch (choise)

{

case 1:

{

matrix.removeComponentOfIndex();

break;

}

case 2:

{

matrix.removeComponentOfValue();

break;

}

default:

cout << "Такого пункту немає!" << endl;

break;

}

break;

}

case 6:

{

matrix.replaceComponent();

break;

}

case 7:

{

matrix.searchElement();

break;

}

case 8:

{

matrix.createArrayOfSum();

matrix.printArrayOfSum();

}

case 9:

{

exit = false;

break;

}

default:

{

cout << "Такого пункту немає!" << endl;

break;

}

}

}

}

Файл Func.h

#pragma once

#include <iostream>

#include <iomanip>

#include <ctime>

using namespace std;

struct Matrix

{

public:

int rows;

int columns;

private:

int\* arr = nullptr;

int\* mass = nullptr;

int counter = 0;

public:

void create(); //Створення матриці

void fillRandomMatrix(); //Заповнення матриці рандомно

void fillManualMatrix(); //Заповнення матриці вручну

void printArr(); //Друк матриці

void printArrOfIndex(); //Друк матриці з індексами

void addComponent(); //Додавання елемента в матрицю

void replaceComponent(); //Заміна компонента матриці

void removeComponentOfIndex(); //видалення компонента по індексу

int removeComponentOfValue(); //видалення компонента по значенню

void createArrayOfSum(); //Створення масиву із сум додатніх елеметів рядків матриці

void printArrayOfSum(); //Друк другого створеного масиву

void searchElement(); //пошук елемента

};

Файл Func.cpp

#include"Func.h"

void Matrix::create()

{

if (arr != nullptr)

{

delete[] arr;

arr = nullptr;

}

arr = new int[rows\*columns];

}

void Matrix::fillRandomMatrix()

{

srand(time(NULL));

int k = 0;

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

{

for (int j = 0; j < columns; j++, k++)

{

arr[k] = rand() % 90 + 1;

}

}

k = 0;

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

{

for (int j = 0; j < (columns \* 0.7) - 0.5; j++, k++)

{

arr[k] = 0;

}

}

k = 0;

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

{

for (int j = 0; j < columns; j++, k++)

{

swap(arr[rows\*i+j], arr[rand()% rows \* i + j]);

}

}

}

void Matrix::fillManualMatrix()

{

int k = 0;

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

{

for (int j = 0; j < columns; j++, k++)

{

cout << '[' << i << "][" << j << "] = "; cin >> arr[k];

}

}

}

void Matrix::printArr()

{

int k = 0;

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

{

for (int j = 0; j < columns; j++, k++)

{

cout << setw(4) << arr[k];

}

cout << endl << endl;

}

}

void Matrix::addComponent()

{

int i, j;

cout << endl;

cout << "Рядок елемента для додавання: "; cin >> i;

cout << "Стовбець елемента для додавання: "; cin >> j;

if (arr[rows \* i + j] == 0)

{

cout << "Введіть число для додаваня в матрицю: "; cin >> arr[rows \* i + j];

}

else

{

cout << "Не можна можна замінити значення!" << endl;

}

}

void Matrix::replaceComponent()

{

int i, j;

cout << endl;

cout << "Рядок елемента для заміни: "; cin >> i;

cout << "Стовбець елемента для заміни: "; cin >> j;

cout << "Введіть число для заміни в матриці: "; cin >> arr[rows \* i + j];

}

void Matrix::removeComponentOfIndex()

{

int i, j;

cout << endl;

cout << "Рядок елемента для видалення: "; cin >> i;

cout << "Стовбець елемента для видалення: "; cin >> j;

arr[rows \* i + j] = 0;

if (arr[rows \* i + j] == 0)

{

cout << "Елемент успішно видалено!" << endl;

}

else

{

cout << "Помилка видалення елемента!" << endl;

}

}

int Matrix::removeComponentOfValue()

{

int value;

cout << "Введіть значення елементу, який бажаєте видалити: "; cin >> value;

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

{

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

{

if (arr[rows \* i + j] == value)

{

arr[rows \* i + j] = 0;

return 0;

}

}

}

cout << "Такого значення немає!" << endl;

return 1;

}

void Matrix::createArrayOfSum()

{

if (mass != nullptr)

{

delete[] mass;

mass = nullptr;

}

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

{

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

{

if (arr[rows \* i + j] > 0)

{

counter++;

break;

}

}

}

mass = new int[counter];

int sum, k = 0;

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

{

sum = 0;

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

{

if (arr[rows \* i + j] > 0)

{

sum += arr[rows \* i + j];

}

}

if (sum > 0)

{

mass[k] = sum;

k++;

}

}

}

void Matrix::printArrayOfSum()

{

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

{

cout << "mass[" << i << "] = " << mass[i] << endl;

}

}

void Matrix::searchElement()

{

int i, j;

cout << endl;

cout << "Рядок елемента: "; cin >> i;

cout << "Стовбець елемента: "; cin >> j;

cout << "Знайдений елемент: " << arr[rows \* i + j] << endl;

}

void Matrix::printArrOfIndex()

{

int k = 0;

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

{

for (int j = 0; j < columns; j++ , k++)

{

cout << "arr[" << i << "][" << j << "] = " << arr[k] << endl;

}

}

}

**Результати тестувань:**

Тест 1

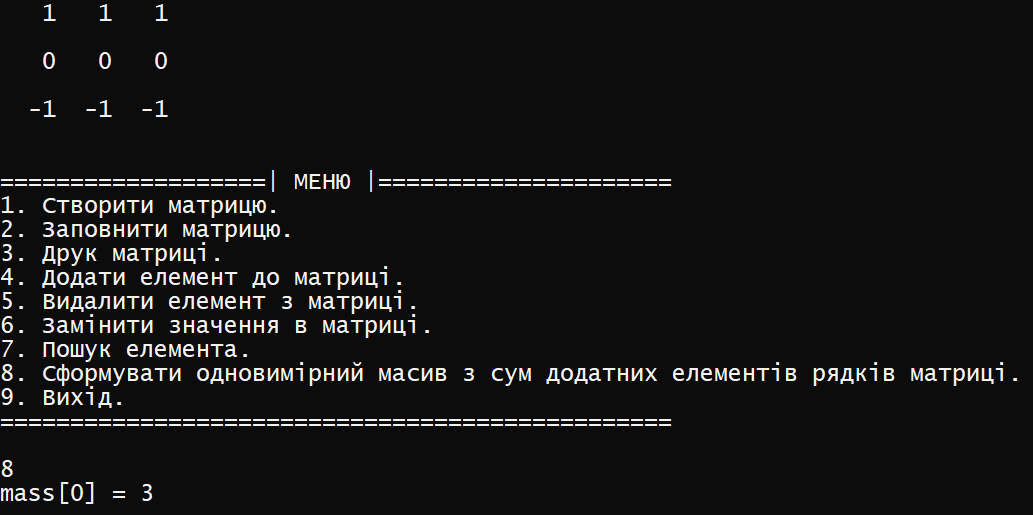


Рис. 4 – Тест 1

Тест 2

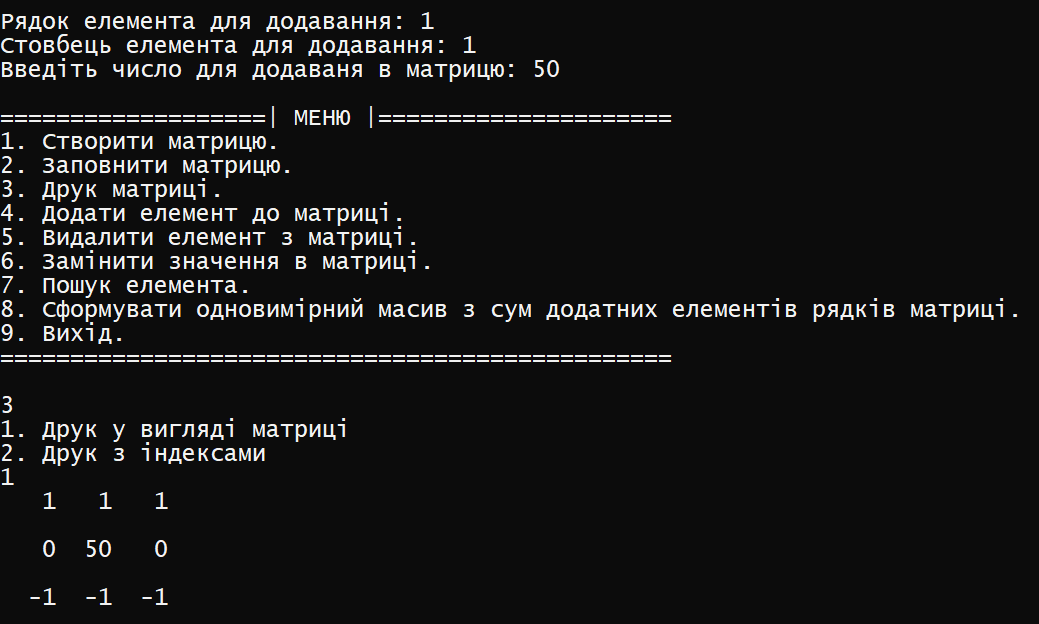


Рис. 5 – Тест 2

Тест 3

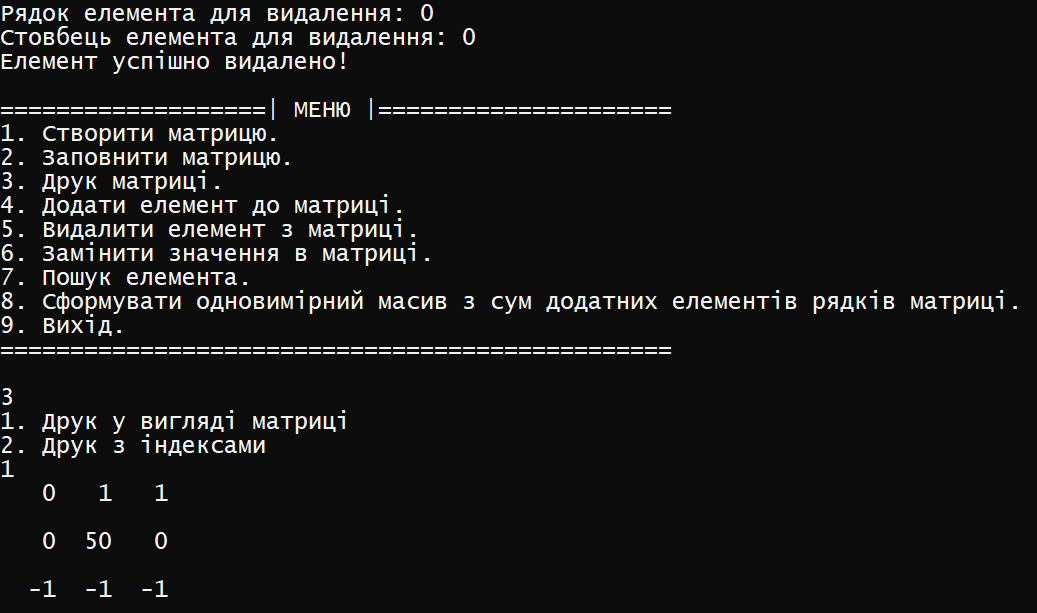


Рис. 6 – Тест 3

Тест 4

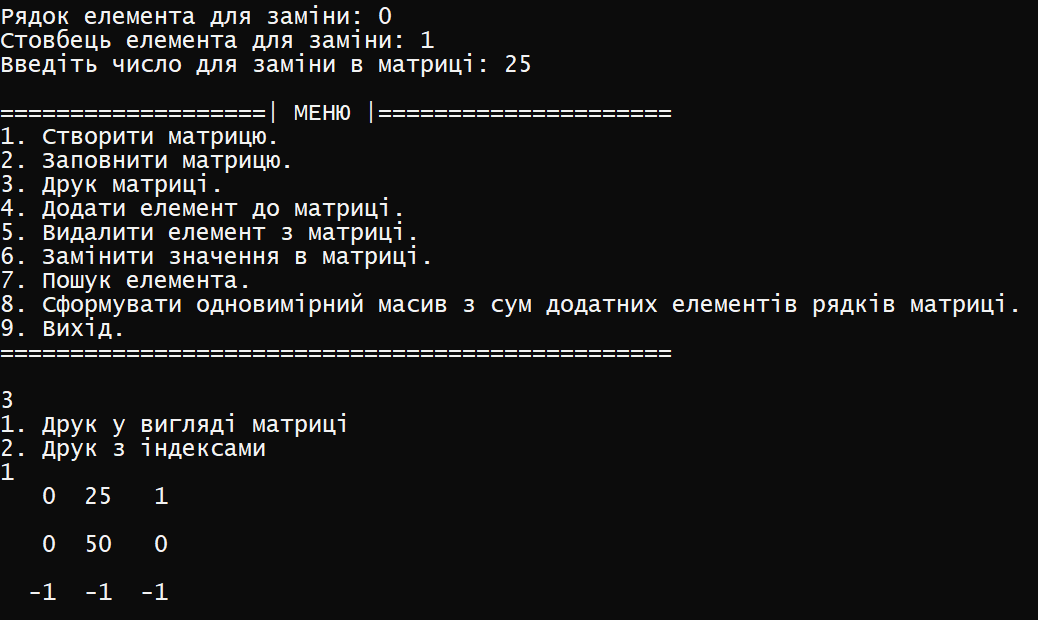


Рис. 7 – Тест 4

Тест 5

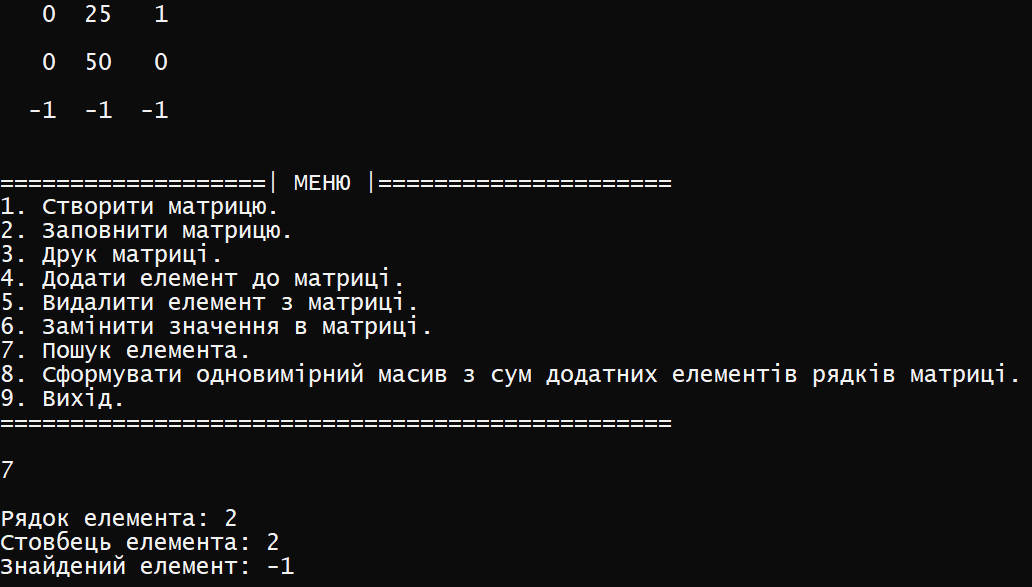


Рис. 8 – Тест 5

**Аналіз результатів:**

Як видно на скріншоті програма правильно виконує алгоритм індивідуального завдання. Для тестування була спеціально створена матриця з рядками в яких є додатні, нульові та від'ємні елементи. Програма корректно видаляє, додає, замінюює та відшукує елементи матриці.

**Висновок:**

Під час виконання практичної роботи, працював с одновмірним динамічним масивом. Так як це масив, швидкість доступу до елементів відбувається швидше ніж в двозв’язному, тому що доступ відбувається за індексом. Проте в список можна в любе місце додати не чіпаючи іншу частину списку, а для масива треба виділяти нову пам’ять та копіювати елементи з одного масиву в інший.