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

Білий О.В.

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

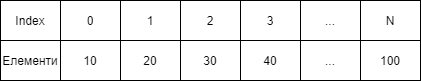


Рис. 1 – Графічне представлення одновимірного масиву

**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 елементів стовпців матриці (Варіант 2).

**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 | cols | Натуральне число | 10 |

Формат вихідних даних

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № з/п | Найменування даних | Умовне позначення | Вимоги до даних | Приклад |
| 1 | Матриця | array | Таблиця цілих чисел, розріджена матриця | 0 0 -6 0  0 0 1 0  0 0 -8 5 |
| 2 | Одновимірний масив для індивідуального завдання | mass | Цілі числа | 0, -2, 0, 1, -5, 2, -3 |

**Тести:**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Назва тесту | Вхідні дані | Результат |
| 1. | Індивідуальне завдання |  | Одновимірний масив зі значеннями |
| 2. | Додавання елементу | Матриця  з тесту №1 | Нульовий елемент замінится на число |
| 3. | Видалення елементу | Матриця  з тесту №1 | Ненульовий елемент стане нульовим |
| 4. | Заміна елементу | Матриця  з тесту №1 | Будь-який елемент може прийняти друге значення |
| 5. | Пошук елементу | Матриця з тесту №1 | Вивести елемент за індексами |

**Алгоритм:**

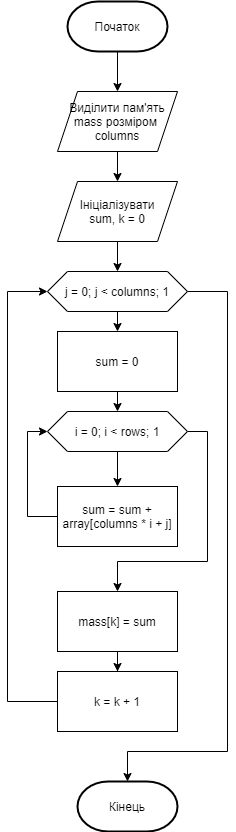
****

Рис. 2 – Алгоритм індивідуального завдання

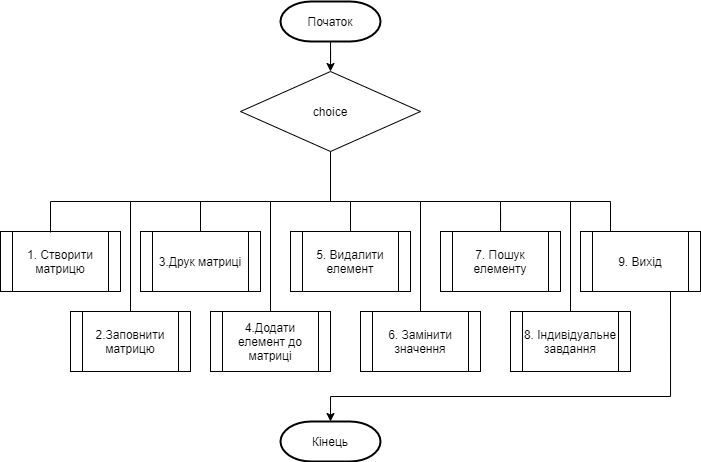
****

Рис. 3 – Алгоритм програми

**Контрольний приклад:**

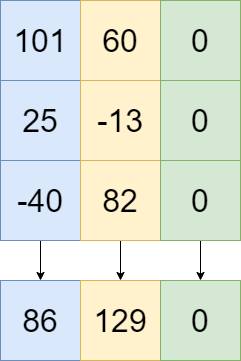


Рис. 4 – Контрольний приклад

**Текст програми:**

Файл Source.cpp

#include "Functions.h"

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

bool exit = true;

int\* array = nullptr, \*mass = nullptr;

int rows, cols;

while (exit)

{

printMenu();

int choise;

cin >> choise;

switch (choise)

{

case 1:

{

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

while (rows < 0)

{

cout << "Кількість рядків не може бути від'ємною,спробуйте ще раз! " << endl;

cin >> rows;

}

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

while (cols < 0)

{

cout << "Кількість стовпців не може бути від'ємною,спробуйте ще раз! " << endl;

cin >> cols;

}

array = createMatrix(rows, cols);

break;

}

case 2:

{

if (array == nullptr)

{

cout << "Необхідно спочатку створити матрицю!" << endl;

break;

}

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

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

cin >> choise;

switch (choise)

{

case 1:

{

fillManualMatrix(array, rows, cols);

break;

}

case 2:

{

fillRandomMatrix(array, rows, cols);

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:

{

printMatrix(array, rows, cols);

break;

}

case 2:

{

printMatrixOfIndex(array, rows, cols);

break;

}

default:

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

break;

}

break;

}

case 4:

{

int i, j, value;

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

while (i < 0 || i > rows)

{

cout << "Номер рядка не може бути від'ємним або більшим за розмірність матриці, спробуйте ще раз! " << endl;

cin >> i;

}

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

while (j < 0 || j > cols)

{

cout << "Номер стовбця не може бути від'ємним або більшим за розмірність матриці, спробуйте ще раз! " << endl;

cin >> j;

}

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

{

cout << "Введіть число для додаваня в матрицю: "; cin >> value;

addElementToMatrix(i, j, value, array, rows);

}

else

{

cout << "Не можна додати елемент в заповнену комірку матриці!" << endl;

}

break;

}

case 5:

{

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

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

cin >> choise;

switch (choise)

{

case 1:

{

int i, j;

cout << endl;

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

while (i < 0 || i > rows)

{

cout << "Номер рядка не може бути від'ємним або більшим за розмірність матриці, спробуйте ще раз! " << endl;

cin >> i;

}

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

while (j < 0 || j > cols)

{

cout << "Номер стовбця не може бути від'ємним або більшим за розмірність матриці, спробуйте ще раз! " << endl;

cin >> j;

}

deleteElementOfIndex(i, j, array, rows);

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

{

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

}

else

{

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

}

break;

}

case 2:

{

int value;

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

if (deleteElementOfValue(value, array, rows, cols) == 1)

{

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

}

break;

}

default:

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

break;

}

break;

}

case 6:

{

int i, j,value;

cout << endl;

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

while (i < 0 || i > rows)

{

cout << "Номер рядка не може бути від'ємним або більшим за розмірність матриці, спробуйте ще раз! " << endl;

cin >> i;

}

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

while (j < 0 || j > cols)

{

cout << "Номер стовбця не може бути від'ємним або більшим за розмірність матриці, спробуйте ще раз! " << endl;

cin >> j;

}

cout << "Введіть число для заміни в матриці: "; cin >> value;

replaceElementOfMatrix(i, j, value, array, rows);

break;

}

case 7:

{

int i, j;

cout << endl;

cout << "Рядок елемента: "; cin >> i;

while (i < 0 || i > rows)

{

cout << "Номер рядка не може бути від'ємним або більшим за розмірність матриці, спробуйте ще раз! " << endl;

cin >> i;

}

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

while (j < 0 || j > cols)

{

cout << "Номер стовбця не може бути від'ємним або більшим за розмірність матриці, спробуйте ще раз! " << endl;

cin >> j;

}

cout << "Знайдений елемент: " << searchElementOfMatrix(i, j, array, rows) << endl;

break;

}

case 8:

{

if (mass != nullptr)

{

delete[] mass;

mass = nullptr;

}

mass = createMatrixOfSum(array, rows, cols);

printMatrixOfSum(mass, cols);

break;

}

case 9:

{

exit = false;

break;

}

default:

{

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

break;

}

}

}

}

Файл Functions.h

#pragma once

#include <iostream>

#include<windows.h>

#include <iomanip>

#include <ctime>

using namespace std;

void printMenu(); //друк головного меню

int\* createMatrix(int rows, int columns); //Створення матриці

void fillRandomMatrix(int\* array, int rows, int columns); //Заповнення матриці рандомно

void fillManualMatrix(int\* array, int rows, int columns); //Заповнення матриці вручну

void printMatrix(int\* array, int rows, int columns); //Друк матриці

void printMatrixOfIndex(int\* array, int rows, int columns); //Друк матриці з індексами

void addElementToMatrix(int i, int j, int value, int\* array, int rows); //Додавання елемента в матрицю

void replaceElementOfMatrix(int i, int j, int value, int\* array, int rows); //Заміна компонента матриці

void deleteElementOfIndex(int i, int j, int\* array, int rows); //видалення компонента по індексу

int deleteElementOfValue(int value, int\* array, int rows, int columns); //видалення компонента по значенню

int\* createMatrixOfSum(int\* array, int rows, int columns); //Створення масиву із сум стовбців матриці

void printMatrixOfSum(int\* mass, int columns); //Друк другого створеного масиву

int searchElementOfMatrix(int i, int j, int\* array, int rows); //пошук елемента

Файл Functions.cpp

#include "Functions.h"

void printMenu()

{

cout << endl;

cout << "Головне 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 стовпців матриці." << endl;

cout << "9. Вихід." << endl;

cout << endl;

}

int\* createMatrix(int rows, int columns)

{

int\* array = nullptr;

array = new int[rows \* columns];

return array;

}

void fillRandomMatrix(int\* array, int rows, int columns)

{

srand(time(NULL));

int k = 0;

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

{

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

{

array[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++)

{

array[k] = 0;

}

}

k = 0;

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

{

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

{

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

}

}

}

void fillManualMatrix(int\* array, int rows, int columns)

{

int k = 0;

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

{

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

{

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

}

}

}

void printMatrix(int\* array, int rows, int columns)

{

int k = 0;

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

{

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

{

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

}

cout << endl << endl;

}

}

void printMatrixOfIndex(int\* array, int rows, int columns)

{

int k = 0;

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

{

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

{

if (array[k] != 0)

{

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

}

}

}

}

void addElementToMatrix(int i, int j, int value, int\* array, int rows)

{

array[rows \* i + j] = value;

}

void replaceElementOfMatrix(int i, int j, int value, int\* array, int rows)

{

array[rows \* i + j] = value;

}

void deleteElementOfIndex(int i, int j, int\* array, int rows)

{

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

}

int deleteElementOfValue(int value, int\* array, int rows, int columns)

{

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

{

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

{

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

{

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

return 0;

}

}

}

return 1;

}

int\* createMatrixOfSum(int\* array, int rows, int columns)

{

int\* mass = nullptr;

mass = new int[columns];

int sum, k = 0;

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

{

sum = 0;

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

{

sum += array[columns \* i + j];

}

mass[k] = sum;

k++;

}

return mass;

}

void printMatrixOfSum(int\* mass, int columns)

{

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

{

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

}

}

int searchElementOfMatrix(int i, int j, int\* array, int rows)

{

return array[rows \* i + j];

}

**Результати тестувань:**

Тест 1

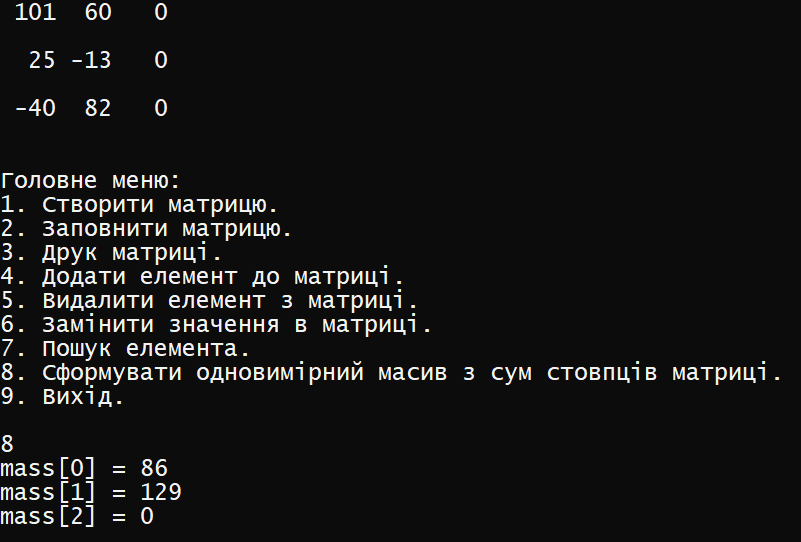


Рис. 5 – Тест 1

Тест 2

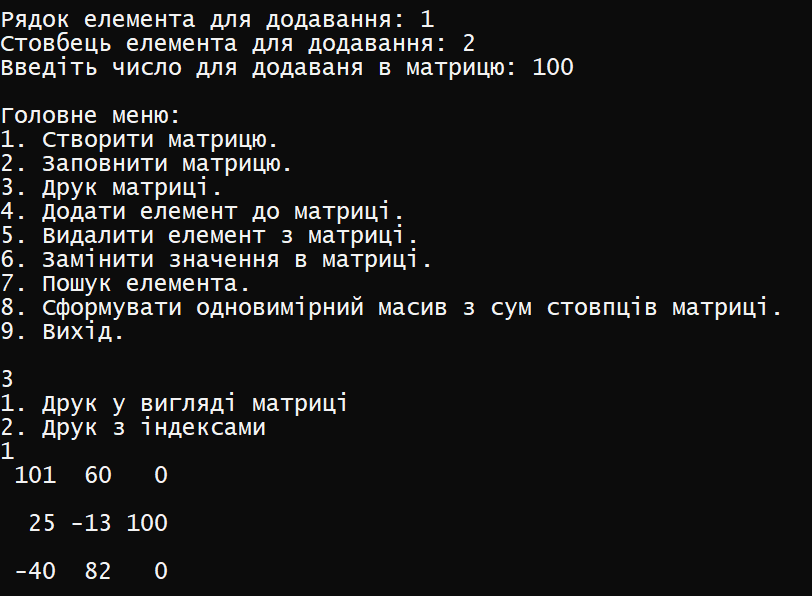


Рис. 6 – Тест 2

Тест 3

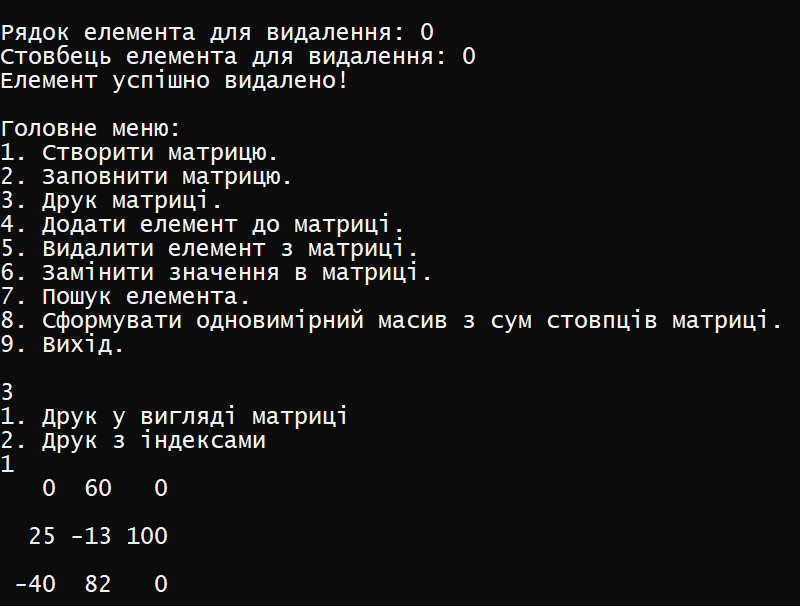


Рис. 7 – Тест 3

Тест 4

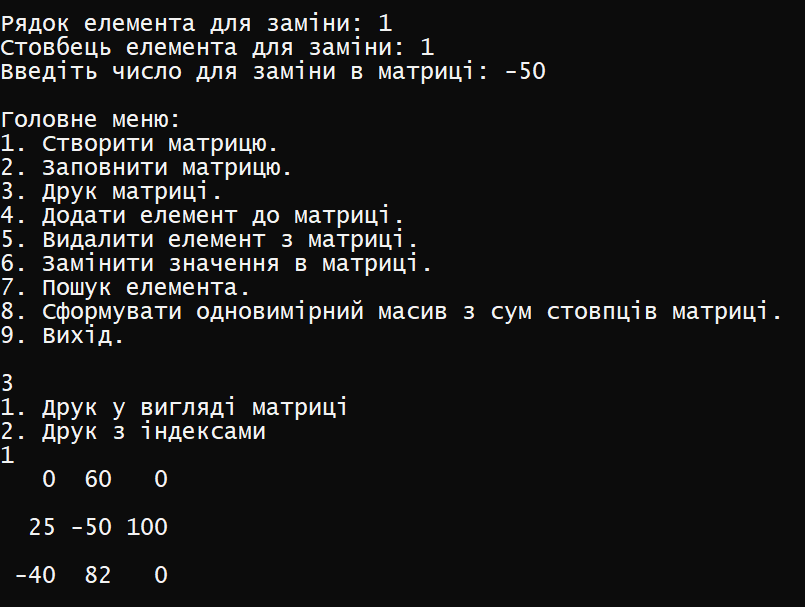


Рис. 8 – Тест 4

Тест 5

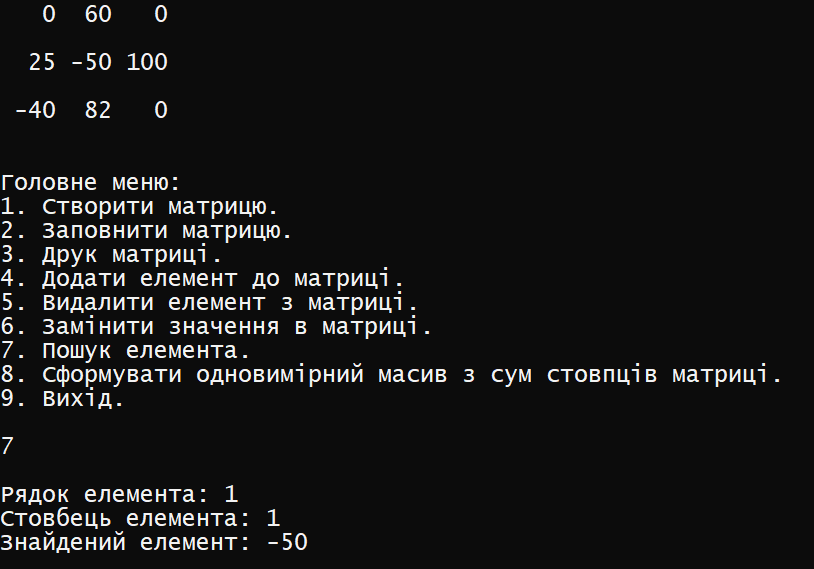


Рис. 9 – Тест 5

**Аналіз результатів:**

Програма працює коректно, вона дає змогу користувачу створювати та заповнювати матрицю, виводити матрицю на екран різними способами, додавати, видаляти, шукати та замінювати значення в матриці. Також програма може створювати новий одновимірний масив с сумами стовпців матриці.

**Висновок:**

При виконнані прктичної роботи, для зберігання матриці використовував одновимірний динамічний масив. У масиві доступ до елементів відбувається швидше, ніж у двозв’язному списку, адже масив це непреривна область пам’яті, а у списку доступ відбувається за вказівниками. Для того, щоб збільшити розміри масиву, необхідно виділятинову область пам’яті необхідних розмірів і копіювати значення з одного масиву в інший, та добавляти нові, тим часом у список можна у будь-яке місце добавити новий елемент.