**СТРУКТУРА**

**звіту з лабораторної роботи**

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**

**“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**

**імені ІГОРЯ СІКОРСЬКОГО”**

**ЗВІТ**

**з лабораторної роботи №6**

**з навчальної дисципліни “Програмування складних алгоритмів”**

**Тема: Розріджені матриці**

**Варіант №1**

**Виконав студент групи ТР–15**

Руденко Владислав Ігорович

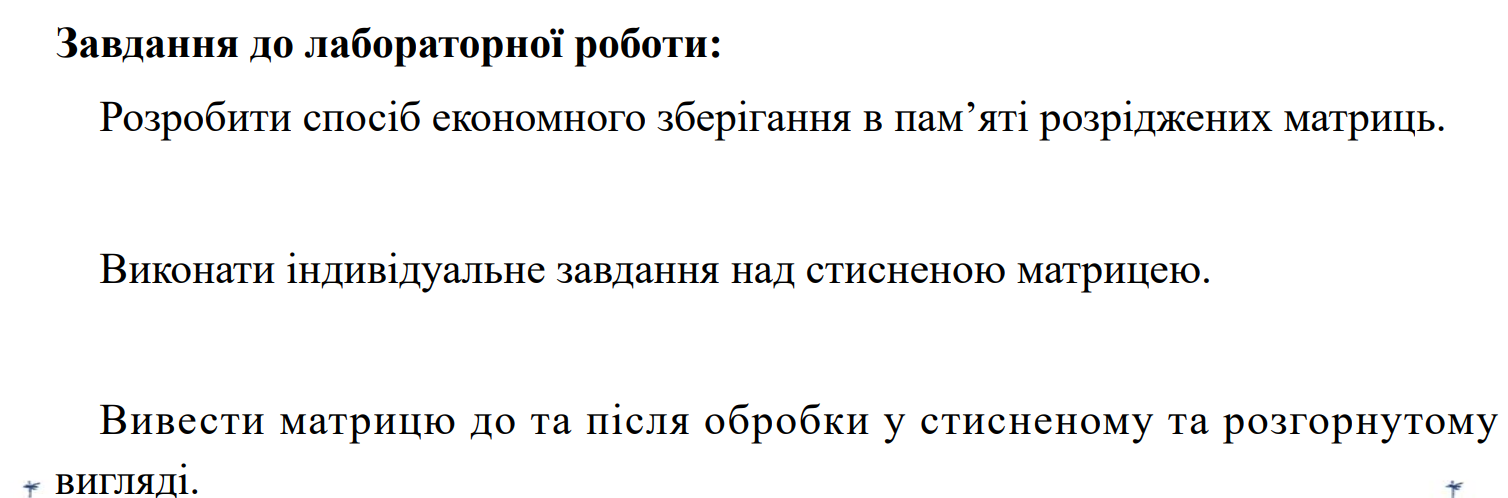
з оцінкою \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

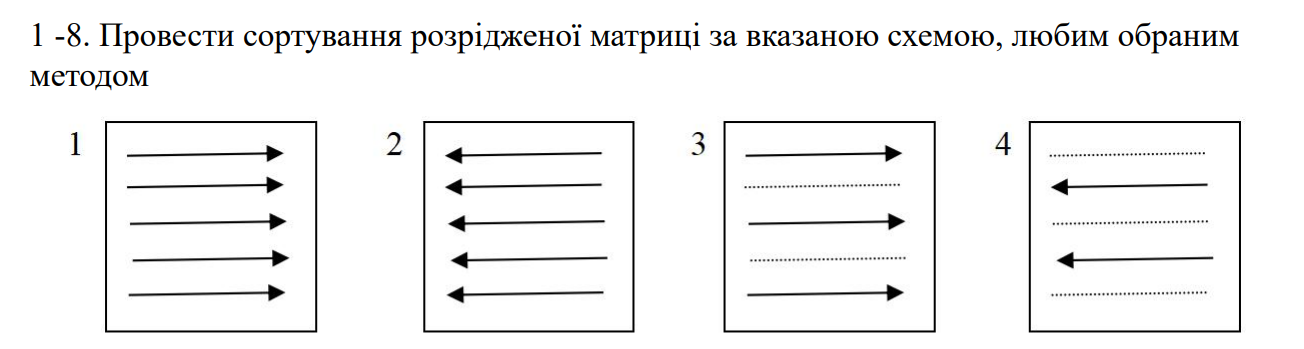
**Перевірив доцент кафедри**

Андрій ОНИСЬКО\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_ року

**Київ 2022**

**І. Завдання + Мета:**



1. **Теоретична частина.**

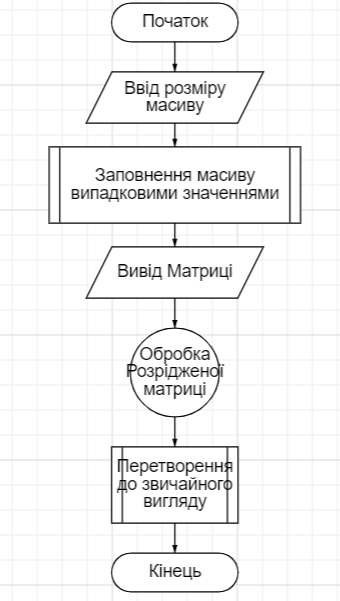
В чисельний аналіз і наукові обчислення, a розріджена матриця або розріджений масив є матриця в якому більшість елементів дорівнює нулю. Не існує чіткого визначення, скільки елементів повинно дорівнювати нулю, щоб матриця розглядалася розріджений але загальним критерієм є те, що кількість ненульових елементів приблизно відповідає кількості рядків або стовпців. На відміну від цього, якщо більшість елементів ненульові, тоді розглядається матриця щільний. Кількість нульових значень елементів, поділене на загальну кількість елементів (наприклад, m × n для матриці m × n), іноді називають розрідженість матриці.

Концептуально розрідженість відповідає системам з кількома попарними взаємодіями. Наприклад, розглянемо лінію куль, з'єднаних пружинами від однієї до іншої: це розріджена система, оскільки зв'язані лише сусідні кульки. На відміну від цього, якби одна і та ж лінія куль мала пружини, що з’єднують кожну кульку з усіма іншими кульками, система відповідала б щільній матриці. Поняття розрідженості корисно в комбінаторика та сфери застосування, такі як теорія мережі і чисельний аналіз, які зазвичай мають низьку щільність значущих даних або з'єднань. Великі розріджені матриці часто з'являються в науковий або машинобудування додатки при вирішенні диференціальні рівняння з частинними похідними.

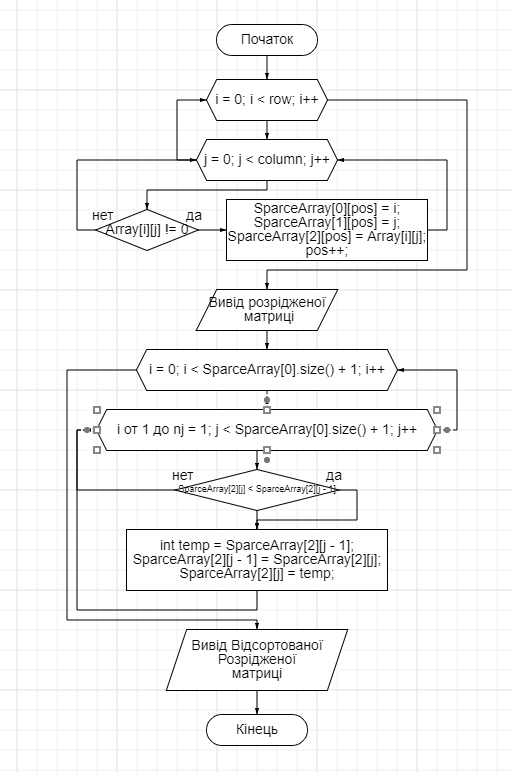
При зберіганні та маніпулюванні розрідженими матрицями на a комп'ютер, корисно і часто потрібно використовувати спеціалізовані алгоритми і структури даних які використовують переваги розрідженої структури матриці. Для розріджених матриць були створені спеціалізовані комп'ютери, оскільки вони поширені в галузі машинного навчання. Операції із використанням стандартних щільно-матричних структур та алгоритмів є повільними та неефективними при застосуванні до великих розріджених матриць як обробка та пам'яті витрачаються на нулі. Розріджені дані за своєю природою легше стиснута і, отже, вимагає значно менше зберігання. Деякі дуже великі розріджені матриці неможливо маніпулювати за допомогою стандартних алгоритмів щільної матриці.

**ІІ. Результати виконання лабораторної роботи.**

**Блок Схеми:**



*Блок-Схема 1 (основна частина)*



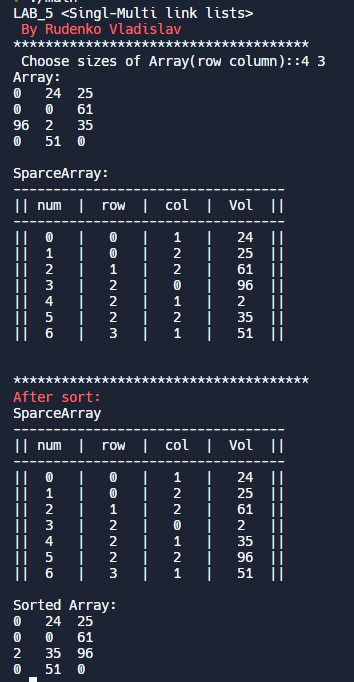
*Блок-Схема 2 (Обробка Розрідженої матриці)*

**Опис програми (написана на мові Сі++):**

Метою програми було створення розрідженої матриці, її особливість в тім, що вона записує тільки значення, які не дорівнюють 0. Для доказу її ефективності було створено звичайний двовимірний масив, заповнений значеннями які при 50% шансі дорівнювали б 0. Цю матрицю згодом перезаписано в розріджену, шляхом порівняння значень в матриці та умовами зберігання, якщо все гаразд, значення записується до розрідженої матриці. Розріджена матриця це масив які складається з 3х стовпців, які відповідають за положення елементу та його значення. Та рядків, які є кількістю елементів масива, які не дорівнюють 0. Після чого Роз.матрицю було представлено користувачеві та почато сортування її по рядкам (зрівнюючи з звичайною матрицею) для цього брався діапазон, в якому усі значення мали мати однакові значення рядка, за цими діапазонами відбувалось сортування бульбашкою. При завершені виконання сортування матриця виводилась користувачеві в новому вигляді та конвертується у вигляд звичайної матриці, оскільки параметри ширини та довжини первинної матриці ми зберігали, ми маємо можливість виконати цей процес. Завершена та відсортована звичайна матриця виводиться користувачеві, після чого виконання завершується.

**Результати роботи:**

**Результати роботи у вигляді скріншотів:**



**Посилання на repl.it:  
<https://replit.com/join/vkhyvldhgc-hetik>**

**Ш. Висновки.**

В ході виконання лабораторної роботи №6 було дослідженно спосіб економічного зберігання даних, а саме використання розріджених матриць. Було створено таку матрицю, використовуючи значення звичайного двовимірного масиву, який має в собі нулі. Над розрідженою матрицею проведено сортування вздовж рядків. В ході виконання було виявленно та опрацьовано деяку кількусть помилок.

**Програмний код:  
main.cpp:**

#include <iostream>

#include <vector>

#include <chrono>

using namespace std;

using milliseconds = chrono::duration<long long, micro>;

#define RED "\x1b[31m"

#define RESET "\x1b[0m"

vector<vector<int>> Array;

vector<vector<int>> SparceArray(3);

int row, column, i, j, elements = 0;

int main()

{

srand(time(0));

int pos = 0, temp, min, max;

cout << "LAB\_5 <Singl-Multi link lists>\n" RED " By Rudenko Vladislav " RESET " \n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n Choose sizes of Array(row column)::";

cin >> row >> column;

Array.resize(row);

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

{

Array[i].resize(column);

for (j = 0; j < column; j++)

{

if (rand() % 2 == 0)

Array[i][j] = 0;

else

{

Array[i][j] = rand() % 100;

elements++;

}

}

}

cout << "Array:\n";

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

{

for (j = 0; j < column; j++)

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

cout << endl;

}

for (i = 0; i < 3; i++)

SparceArray[i].resize(elements);

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

for (j = 0; j < column; j++)

{

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

{

SparceArray[0][pos] = i;

SparceArray[1][pos] = j;

SparceArray[2][pos] = Array[i][j];

pos++;

}

}

cout << "\nSparceArray:" << endl;

cout << "----------------------------------" << endl;

cout << "|| num | row | col | Vol ||\n";

cout << "----------------------------------" << endl;

for (i = 0; i < elements; i++)

cout << "||\t" << i << "\t|\t" << SparceArray[0][i] << "\t|\t" << SparceArray[1][i] << "\t|\t" << SparceArray[2][i] << "\t||" << endl;

for (int i = 0; i < SparceArray[0].size() + 1; i++) {

for (int j = 1; j < SparceArray[0].size() + 1; j++) {

if(SparceArray[0][j] != SparceArray[0][j - 1]) continue;

if(SparceArray[2][j] < SparceArray[2][j - 1])

{

int temp = SparceArray[2][j - 1];

SparceArray[2][j - 1] = SparceArray[2][j];

SparceArray[2][j] = temp;

}

}

}

cout << endl << "\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n" RED "After sort:" RESET " \nSparceArray" << endl;

cout << "----------------------------------" << endl;

cout << "|| num | row | col | Vol ||\n";

cout << "----------------------------------" << endl;

for (i = 0; i < elements; i++)

cout << "||\t" << i << "\t|\t" << SparceArray[0][i] << "\t|\t" << SparceArray[1][i] << "\t|\t" << SparceArray[2][i] << "\t||" << endl;

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

{

for (j = 0; j < column; j++)

{

for (int f = 0; f < SparceArray[0].size(); f++)

{

Array[i][j] = 0;

if(SparceArray[0][f] == i && j == SparceArray[1][f])

{

Array[i][j] = SparceArray[2][f];

break;

}

}

}

}

cout << "\nSorted Array:\n";

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

{

for (j = 0; j < column; j++)

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

cout << endl;

}

}