Міністерство освіти і науки України

Івано-Франківський національний технічний університет нафти і газу

Кафедра ІПЗ

**Лабораторна робота № 1**

**«*Базові типи і структури даних*»**

Виконав

ст. гр. ІП-22-1

Микитій Ю.М.

Перевірив

доцент Процюк В.Р.

Івано-Франківськ

2023

**Мета:** Опанування принципів роботи із статичними масивами, заповненими різноманітними значеннями елементів

**Умова: Варіант 15.**

1. Знайти суму і кількість елементів, розташованих до першого від’ємного елементу.  
2. Знайти в масиві і вивести на екран суму абсолютних значень від’ємних елементів масиву.

**Розв’язок:**

#include <iostream>

#include <locale>

#include <ctime>

#include <cstdlib>

using namespace std;

int main()

{

setlocale(LC\_CTYPE, "ukr"); // Встановлення локалізації для коректного виводу символів на українському

cout << "1. Завдання: " << endl;

int arr[14] = { 2,6,8,0,6,-9,-5,8,25,75,-9,12,-90, 25 }; // Створення масиву з 14 елементів

int count = 0, sum = 0;

for (int i = 0; i < 14; i++)

{

if (arr[i] < 0)

{

break; // Виходимо з циклу, якщо зустрічено від'ємний елемент

}

sum += arr[i]; // Додаємо елемент до суми

count++; // Збільшуємо лічильник

}

cout << "Сума: " << sum << endl; // Виводимо суму

cout << "Кiлькiсть елементiв: " << count << endl; // Виводимо кількість елементів

cout << "2. Завдання: " << endl;

int arr2[5][7];

for (int i = 0; i < 5; ++i)

{

for (int j = 0; j < 7; ++j)

{

arr2[i][j] = rand() % 101 - 50; // Генеруємо випадкові числа від -50 до 50 і заповнюємо масив

}

}

int SumAbsValue = 0;

for (int i = 0; i < 5; ++i)

{

for (int j = 0; j < 7; ++j)

{

if (arr2[i][j] < 0)

{

SumAbsValue += abs(arr2[i][j]); // Додаємо абсолютне значення від'ємного елемента до суми

}

}

}

for (int i = 0; i < 5; ++i)

{

for (int j = 0; j < 7; ++j)

{

cout << arr2[i][j] << "\t"; // Виводимо елементи масиву arr2

}

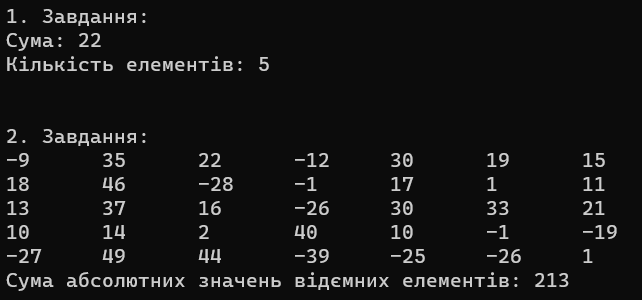
cout << endl;

}

cout << "Сума абсолютних значень вiдємних елементiв: " << SumAbsValue << endl; // Виводимо суму абсолютних значень від'ємних елементів

}

Результат:



**Висновок:** я опанував принципи роботи із статичними масивами, заповненими різноманітними значеннями елементів.

Додаткове:

Умова: Розробити спосіб економічного зберігання в памяті розріджених матриць(таблиць). Розробити процедури і функції для забезпечення доступу (читання-запис) до елементів матриці. В контрольному прикладі забезпечення читання і запис всіх елементів матриці. Оцінити час виконання операцій.

Саме завдання

всі нульові елементи розміщені на стрічках, індекси яких кратні 3

Розв’язок:  
#include <iostream>

#include <vector>

#include <map>

#include <locale>

#include <chrono> //бібліотеку для вимірювання часу

class SparseMatrix {

private:

std::map<int, std::map<int, int>> data;

public:

SparseMatrix() {}

void set(int row, int col, int value) {

if (row % 3 != 0)

data[row][col] = value;

}

int get(int row, int col) {

if (data.find(row) != data.end() && data[row].find(col) != data[row].end()) {

return data[row][col];

}

else {

return 0;

}

}

};

int main() {

setlocale(LC\_CTYPE, "ukr");

// Створення об'єкта класу SparseMatrix

SparseMatrix matrix;

// Початок вимірювання часу заповнення матриці

auto start\_fill = std::chrono::high\_resolution\_clock::now();

/// Заповнюємо матрицю ненульовими значеннями

matrix.set(0, 1, 5);

matrix.set(2, 2, 6);

matrix.set(3, 0, 5);

matrix.set(4, 1, 4);

// Зупинка вимірювання часу заповнення матриці

auto stop\_fill = std::chrono::high\_resolution\_clock::now();

// Обчислення часу заповнення

auto duration\_fill = std::chrono::duration\_cast<std::chrono::microseconds>(stop\_fill - start\_fill);

std::cout << "Час заповнення матрицi: " << duration\_fill.count() << " мiкросекунди" << std::endl;

// Початок вимірювання часу виведення матриці на екран

auto start\_print = std::chrono::high\_resolution\_clock::now();

// Виведення матриці на екран

std::cout << "Розрiджена матриця:" << std::endl;

for (int row = 0; row < 5; row++) {

for (int col = 0; col < 3; col++) {

int value = matrix.get(row, col);

std::cout << value << " ";

}

std::cout << std::endl;

}

// Зупинка вимірювання часу виведення матриці на екран

auto stop\_print = std::chrono::high\_resolution\_clock::now();

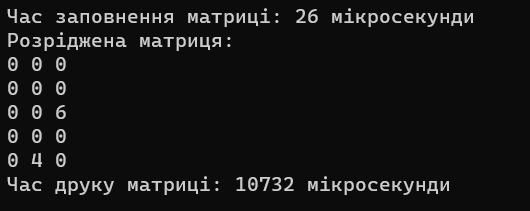
// Обчислення часу виведення

auto duration\_print = std::chrono::duration\_cast<std::chrono::microseconds>(stop\_print - start\_print);

std::cout << "Час друку матрицi: " << duration\_print.count() << " мiкросекунди" << std::endl;

return 0;

}  
  
  
Результат:

  
  
Висновок: Під час виконання цього додаткового завдання я зрозумів і довів собі що розріджена матриця є дуже ефективною в збережені пам’яті.