**Назва роботи:** Дослідження алгоритмів перетворення матриць та отримання

агрегатних значень.

**Мета:** дослідити підходи до пошуку та перетворення на матрицях та

набути практичних навичок використання укладених керувальних дій

повторення і їх з’єднання під час складання програмних специфікацій.

**Варіант:** 27.

**Умова задачі:**

Розробити алгоритм та написати програму, яка складається з наступних

дій:

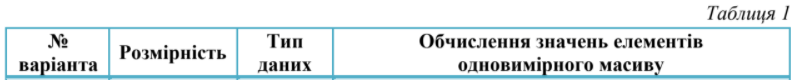
1. Опису змінної індексованого типу (двовимірний масив) згідно з варіантом

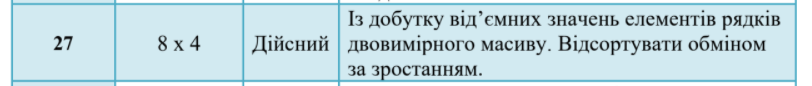
(табл. 1).

2. Ініціювання змінної, що описана в п.1 даного завдання.

3. Створення нової змінної індексованого типу (одновимірний масив) та її

ініціювання значеннями, що обчислюються згідно з варіантом (табл. 1).

****

****

**Постановка задачі:**

Результатом задачі є виведений на екран відсортований за зростанням одновимірний масив(максимально може бути 8 елементів, бо 8 рядків у матриці), ініційований добутками від’ємних значень елементів рядків двовимірного масиву 8\*4(8 рядків, 4 стовпця). Вхідні дані: розмірність двовимірної матриці. Спочатку процедура fillMatrix(matrix[][4]) заповнює матрицю дійсними значеннями, з допомогою вкладених циклів. Виклик функції arrayFromMatrix(matrix, arr) ініціює одновимірний масив добутками від’ємних значень елементів рядків двовимірного масиву та повертає його довжину, ця функція також використовує вкладені цикли. Процедура sortArray(arr, sizeOfArr) сортує масив обміном за зростанням(є вкладені цикли).

**Побудова математичної моделі:**

Складемо таблицю імен змінних:

**Змінні головної програми:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Змінна** | **Тип** | **Ім’я** | **Призначення** |
| Двовимірний масив 8\*4(змінна індексованого типу) | Дійсний | matrix[8][4] | Початкове дане |
| Одновимірний масив(нова змінна індексованого типу) | Дійсний | arr[8] | Результат |
| Кількість елементів, записаних у arr[8] | Цілий | sizeOfArr | Проміжне значення |

**Змінні підпрограми fillMatrix(matrix[][4]):**

|  |  |  |  |
| --- | --- | --- | --- |
| **Змінна** | **Тип** | **Ім’я** | **Призначення** |
| Лічильник, який використовується як індекс рядків двовимірного масива | Цілий | i | Параметр зовнішнього арифметичного циклу(значення від 0 до 7 включно) |
| Лічильник, який використовується як індекс стовпців двовимірного масива | Цілий | j | Параметр внутрішнього арифметичного циклу(значення від 0 до 7 включно) |
| Використовується для доступу до двовимірної матриці та роботи з нею | Покажчик на двовимірний масив дійсних чисел | matrix[][4] | Формальний параметр процедури |

**Змінні підпрограми printMatrix(matrix[][4]):**

|  |  |  |  |
| --- | --- | --- | --- |
| **Змінна** | **Тип** | **Ім’я** | **Призначення** |
| Лічильник, який використовується як індекс рядків двовимірного масива | Цілий | i | Параметр зовнішнього арифметичного циклу(значення від 0 до 7 включно) |
| Лічильник, який використовується як індекс стовпців двовимірного масива | Цілий | j | Параметр внутрішнього арифметичного циклу(значення від 0 до 7 включно) |
| Використовується для доступу до двовимірної матриці та роботи з нею | Покажчик на двовимірний масив дійсних чисел | matrix[][4] | Формальний параметр процедури |

**Змінні підпрограми arrayFromMatrix(matrix[][4] ,\*arr):**

|  |  |  |  |
| --- | --- | --- | --- |
| **Змінна** | **Тип** | **Ім’я** | **Призначення** |
| Кількість елементів, записаних у arr[8] | Цілий | num | Результат виконання функції |
| Лічильник, який використовується як індекс рядків двовимірного масива | Цілий | i | Параметр зовнішнього арифметичного циклу(значення від 0 до 7 включно) |
| Лічильник, який використовується як індекс стовпців двовимірного масива | Цілий | j | Параметр внутрішнього арифметичного циклу(значення від 0 до 7 включно) |
| Використовується для доступу до двовимірної матриці та роботи з нею, показує на перший її елемент | Покажчик на двовимірний масив дійсних чисел | matrix[][4] | Формальний параметр функції |
| Покажчик на перший елемент переданого одновимірного масиву | Покажчик дійсного типу | \*arr | Формальний параметр функції |
| Початкове значення хиба. Набуває істинного значення, якщо в рядку матриці є хоча б одне від’ємне число | Логічний | fl | Проміжне значення |
| Добуток від’ємних значень елементів певного рядка матриці(початкове знач. 1) | Дійсний | dob | Проміжне значення |

**Змінні підпрограми sortArray(\*arr, size):**

|  |  |  |  |
| --- | --- | --- | --- |
| **Змінна** | **Тип** | **Ім’я** | **Призначення** |
| Покажчик на перший елемент переданого одновимірного масиву | Покажчик дійсного типу | \*arr | Формальний параметр функції |
| Розмір цього одновимірного масиву | Цілий | size | Формальний параметр функції |
| Лічильник, який використовується як номер проходу по масиву | Цілий | i | Параметр зовнішнього арифметичного циклу(значення від 0 до size-2 включно) |
| Лічильник, який використовується як індекс елементів одновимірного масиву | Цілий | j | Параметр внутрішнього арифметичного циклу(значення від 0 до 7 включно) |
| Збереження одного зі значень при обміні елементів масиву значеннями | Дійсний | temp | Проміжне значення |

**Змінні підпрограми printArray**(\*arr, size):

|  |  |  |  |
| --- | --- | --- | --- |
| **Змінна** | **Тип** | **Ім’я** | **Призначення** |
| Покажчик на перший елемент переданого одновимірного масиву | Покажчик дійсного типу | \*arr | Формальний параметр функції |
| Розмір цього одновимірного масиву | Цілий | size | Формальний параметр функції |
| Лічильник використовується як індекс елемента масиву | Цілий | i | Параметр циклу |

Спочатку описуємо двовимірний масив 8\*4(8 рядків, 4 стовпця)( змінну індексованого типу), одновимірний масив на 8 елементів та ініціалізуємо їх нулями: matrix[8][4]={}; arr[8]={}.

Перша процедура fillMatrix(matrix) проходить за допомогою зовнішнього та внутрішнього циклів проходить по всіх елементах масиву та присвоює їм випадкові значення у діапазоні від -50 до 50, з можливою деякою дробовою частиною за формулою: matrix[i][j]=(rand()%101-50)\*10^(-rand()%3). Перша частина формули відповідає за діапазон, а друга за дробову частину числа.

Процедурою printMatrix(matrix[][4]), яка аналогічно за допомогою зовнішнього та внутрішнього циклів проходить по всіх елементах та виводить їх, виводимо усю матрицю, щоб перевірити в подальшому правильність виконання програми, побачити згенеровані значення.

Розглянемо наступну функцію arrayFromMatrix(matrix[][4] ,\*arr): На кожній ітерації зовнішнього циклу логічна змінна fl=false, а добутку від’ємних чисел надається значення 1: dob=1. Далі внутрішній цикл проходить весь рядок та знаходить від’ємні елементи. Якщо такі присутні, то fl=true, і добуток домножується на це число. Після завершення внутрішнього циклу отримаємо добуток від’ємних чисел. Далі ми аналізуємо змінну fl: якщо вона залишилася false, то це означає, що в рядку не було від’ємних чисел, і тому ми не записуємо dob в одновимірний масив. Інакше записуємо і збільшуємо кількість елементів у цьому масиві: num=num+1. Повертаємо num як результат виконання функції, це довжина одновимірного масиву, який ми будемо потім сортувати.

Процедура printArray(\*arr, size) викликається двічі, щоб побачити спочатку сформований масив з матриці, а потім відсортований – результат виконання програми.

Процедура sortArray(\*arr, size)сортує масив таким чином, що зовнішній цикл проходить size-1 ітерацій, за які весь масив точно буде впорядкованим. Значення лічильника зовнішнього циклу – номер проходу по масиву. А внутрішній цикл здійснює обхід всього масиву та обмін сусідніх елементів на кожній такій ітерації.

**Розв’язання**

Програмні специфікації запишемо у вигляді псевдокоду та блок-схем.

Крок 1. Визначимо основні дії.

Крок 2. Деталізуємо процедуру fillMatrix(matrix[][4]).

Крок 3. Деталізуємо процедуру printMatrix(matrix[][4]).

Крок 4. Деталізуємо функцію arrayFromMatrix(matrix[][4] ,\*arr).

Крок 5. Деталізуємо процедуру sortArray(\*arr, size).

Крок 6. Деталізуємо процедуру printArray(\*arr, size).

**Псевдокод**

Головна програма:

**початок**

matrix[8][4]={}

arr[8]={}

**виклик процедури** fillMatrix(matrix)

**виклик процедури** printMatrix(matrix)

sizeOfArr=arrayFromMatrix(matrix, arr)

**виклик процедури** printArray(arr, sizeOfArr)

**виклик процедури** sortArray(arr, sizeOfArr)

**виклик процедури** printArray(arr, sizeOfArr)

**кінець**

**Процедура fillMatrix(matrix[][4])**

**повторити**

**для** i **від** 0 **до** 8 **з кроком** 1

**повторити**

**для** j **від** 0 **до** 4 **з кроком** 1

matrix[i][j]=(rand()%101-50)\*10^(-rand()%3)

**все повторити**

**все повторити**

**кінець**

**Процедура printMatrix(matrix[][4])**

**повторити**

**для** i **від** 0 **до** 8 **з кроком** 1

**повторити**

**для** j **від** 0 **до** 4 **з кроком** 1

**виведення:** max[i][j]

**все повторити**

**все повторити**

**кінець**

**Функція** **arrayFromMatrix(matrix[][4] ,\*arr)**

**num=0**

**повторити**

**для** i **від** 0 **до** 8 **з кроком** 1

fl=false

dob=1

**повторити**

**для** j **від** 0 **до** 4 **з кроком** 1

dob=dob\*matrix[i][j]

fl=true

**все повторити**

**якщо** fl==true

**то**

arr[num]=dob

num=num+1

**все якшо**

**все повторити**

**return num**

**Процедура sortArray(\*arr, size)**

**повторити**

**для** i **від** 0 **до** size-1 **з кроком** 1

**повторити**

**для** j **від** 0 **до** size-1 **з кроком** 1

**якщо** arr[j]>arr[j+1]

**то**

temp=arr[j]

arr[j]=arr[j+1]

arr[j+1]=temp

**все якшо**

**все повторити**

**все повторити**

**кінець**

**Процедура printArray**(\*arr, size)

**повторити**

**для** i **від** 0 **до** size **з кроком** 1

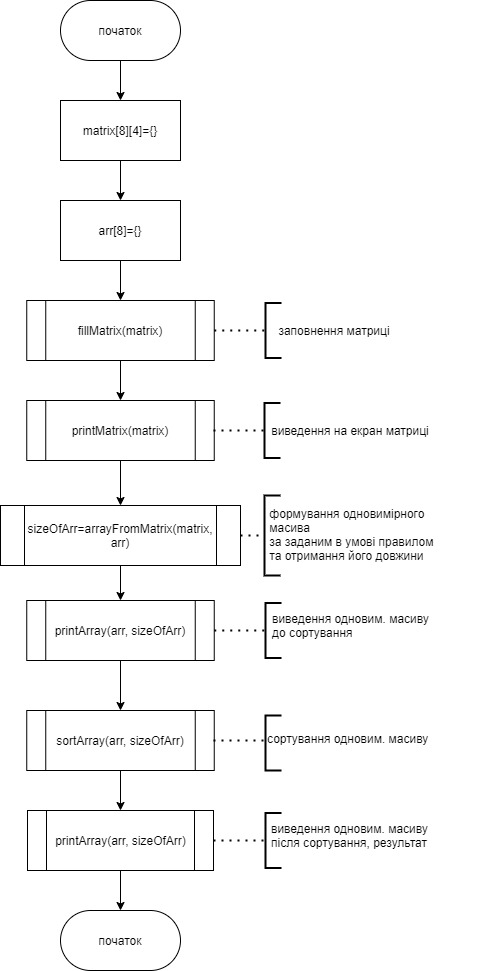
**виведення:** arr[i]

**все повторити**

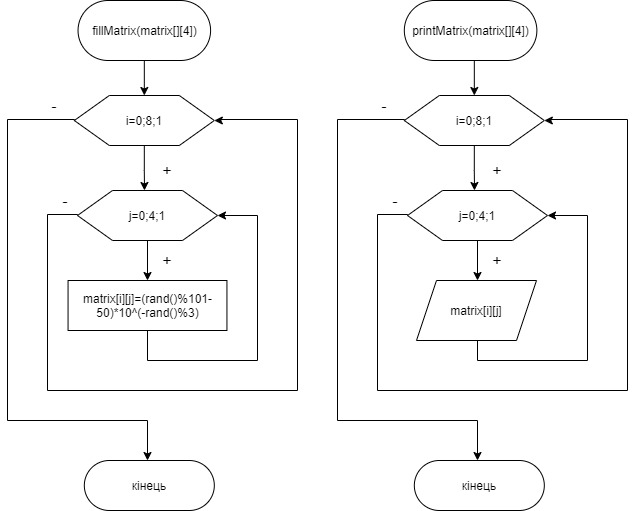
**кінець**

**Блок-схема алгоритму**

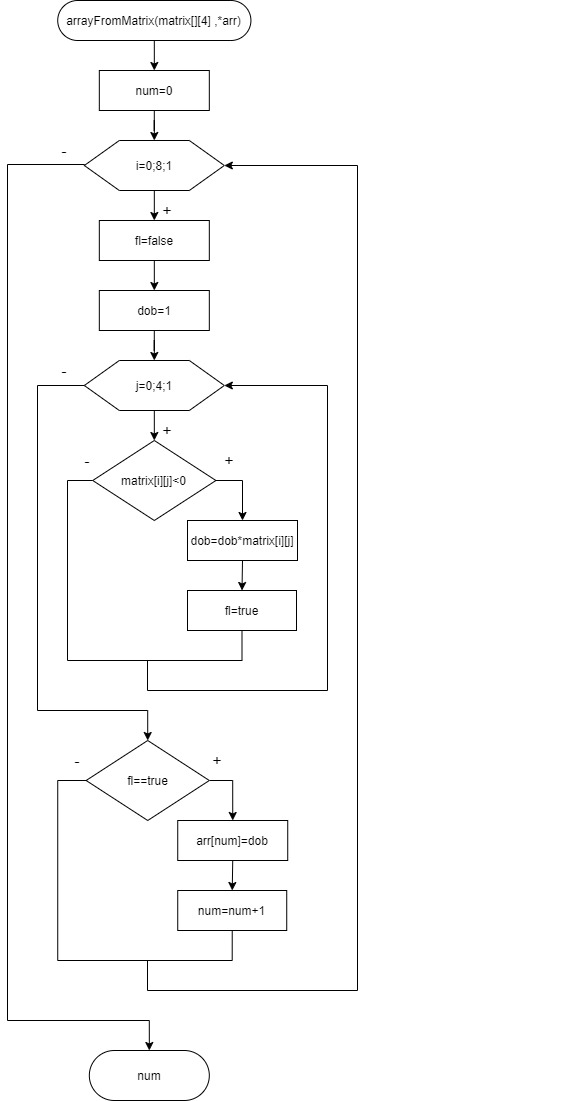
Головна програма



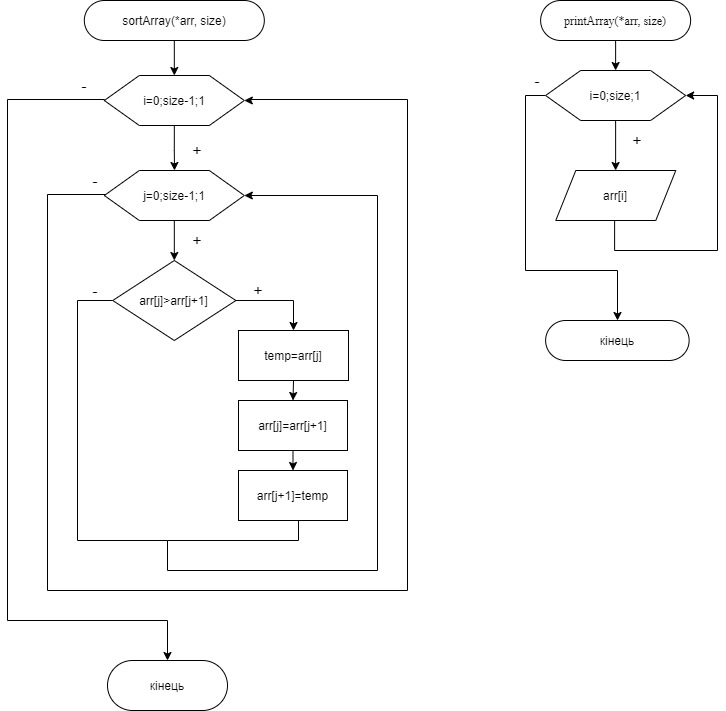
Процедура fillMatrix(matrix[][4]), процедура printMatrix(matrix[][4])



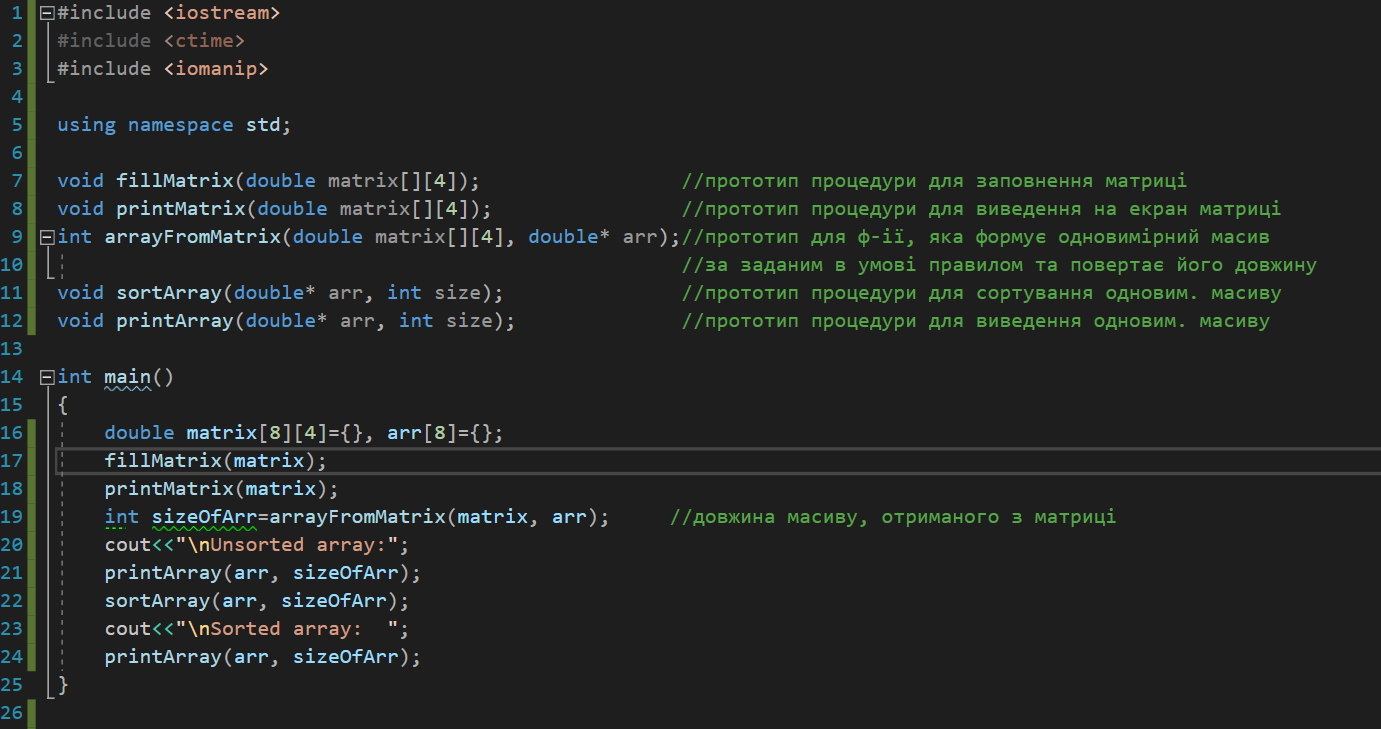
Функція arrayFromMatrix(matrix[][4] ,\*arr)

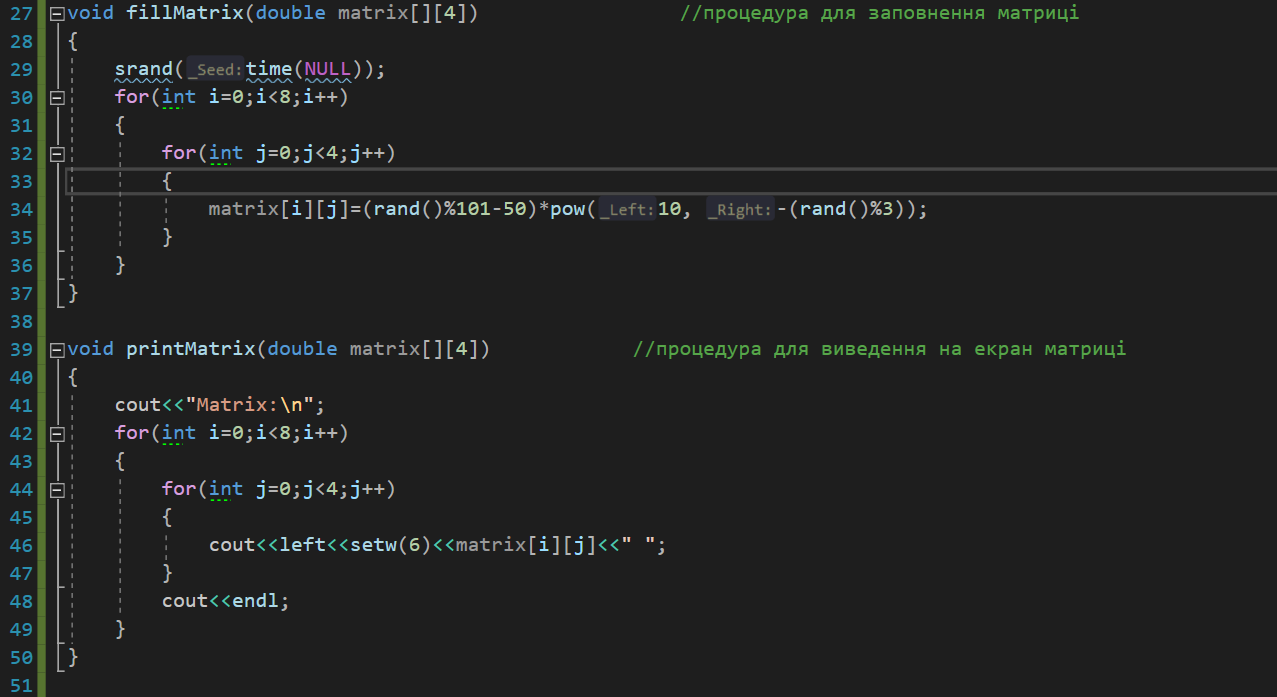


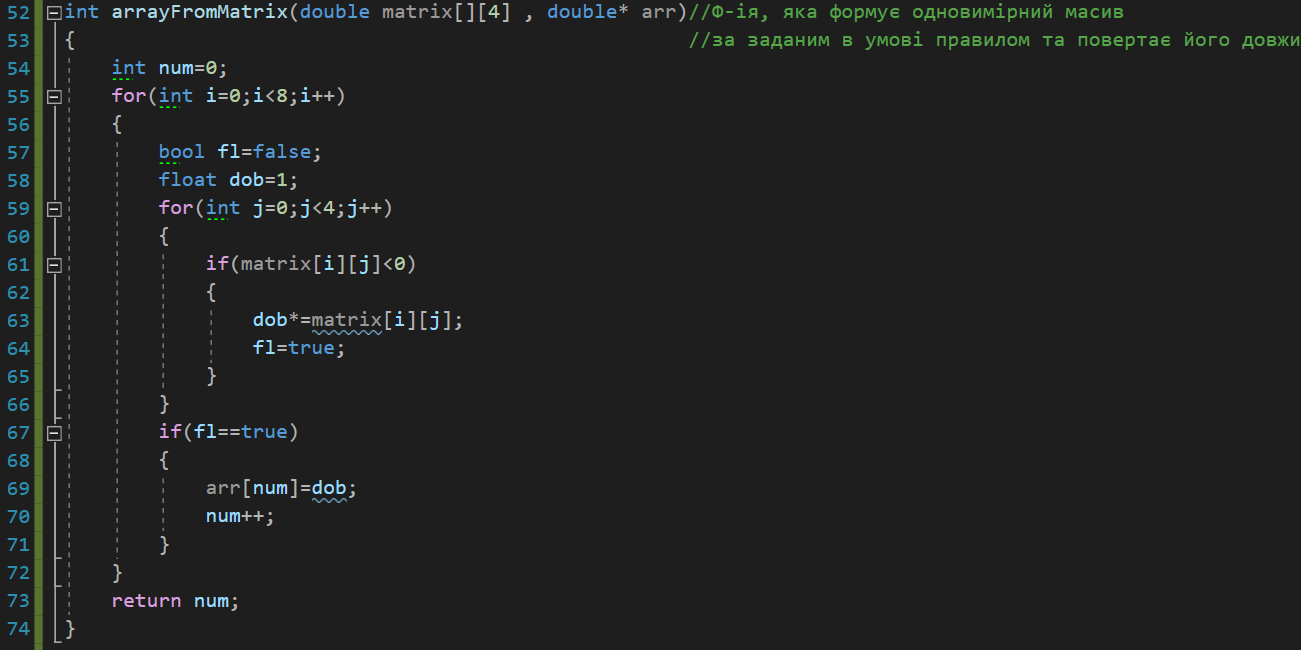
Процедура sortArray(\*arr, size), Процедура printArray(\*arr, size)

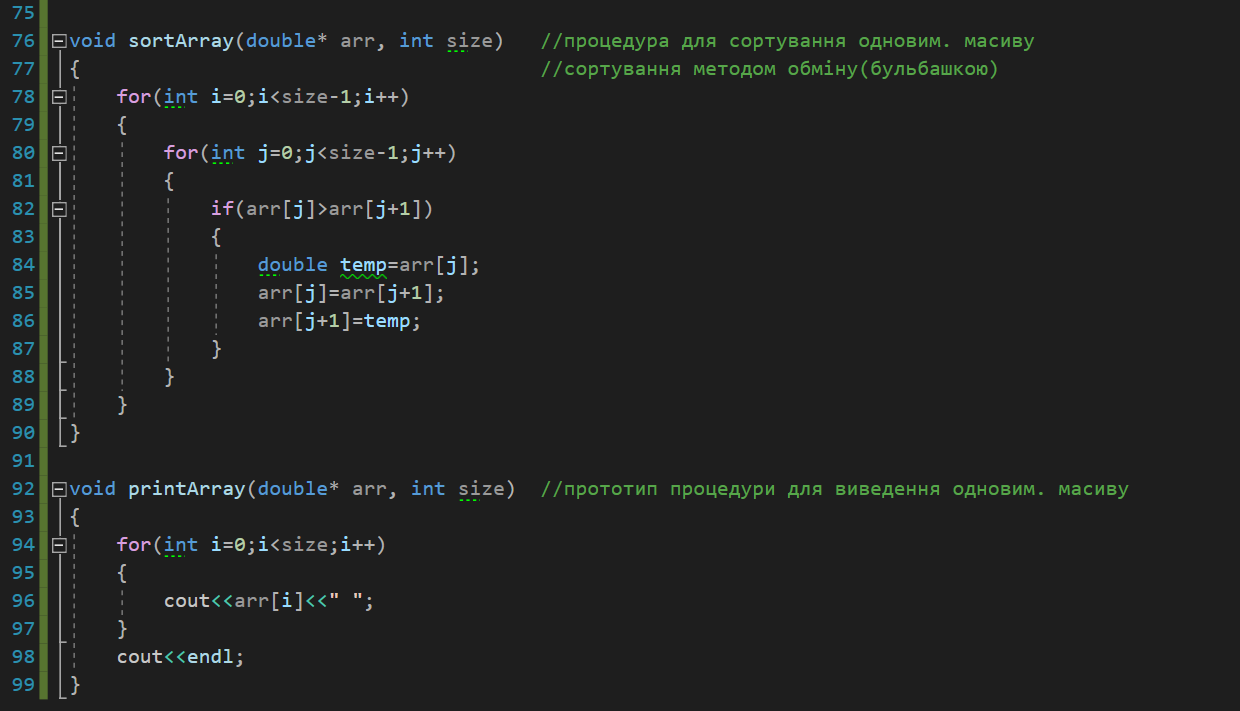


**Код програми(C++)**









**Випробування алгоритму**

|  |  |
| --- | --- |
| Блок | Дія |
|  | **початок** |
| 1 | matrix[8][4]={} |
| 2 | arr[8]={} |
| 3 | **виклик процедури** fillMatrix(matrix) |
| 4 | **виклик процедури** printMatrix(matrix) |

Matrix[8][4]=

|  |  |  |  |
| --- | --- | --- | --- |
| 0.21 | 10 | -0.48 | 3.4 |
| 3.8 | -1.1 | -0.01 | 2.9 |
| 0.35 | -1 | 5 | -1.9 |
| -3 | 3.9 | 0.4 | -3 |
| 0.8 | 7 | -0.11 | -28 |
| 0.31 | -2.4 | -0.3 | -3.2 |
| -0.28 | -0.19 | 0.2 | -14 |
| -4.5 | -0.2 | 29 | 0.32 |

|  |  |
| --- | --- |
| 5 | виклик функції arrayFromMatrix(matrix, arr) |
| 6 | num=0 |
| 7(1 ітерація зовн. циклу) | i=0 |
| 8 | fl=false |
| 9 | dob=1 |
| 10(внутрішній цикл виконав роботу) | fl=true; dob=-0.48, num=1 |
| 11(2 ітерація зовн. циклу) | i=1 |
| 12 | fl=false |
| 13 | dob=1 |
| 14(внутрішній цикл виконав роботу) | fl=true; dob=0.011, num=2 |
| 15(3 ітерація зовн. циклу) | i=2 |
| 16 | fl=false |
| 17 | dob=1 |
| 18(внутрішній цикл виконав роботу) | fl=true; dob=1.9, num=3 |
| 19(4 ітерація зовн. циклу) | i=3 |
| 20 | fl=false |
| 21 | dob=1 |
| 22(внутрішній цикл виконав роботу) | fl=true; dob=9, num=4 |
| 23(5 ітерація зовн. циклу) | i=4 |
| 24 | fl=false |
| 25 | dob=1 |
| 26(внутрішній цикл виконав роботу) | fl=true; dob=3.08, num=5 |
| 27(6 ітерація зовн. циклу) | i=5 |
| 28 | fl=false |
| 29 | dob=1 |
| 30(внутрішній цикл виконав роботу) | fl=true; dob=-2.304, num=6 |
| 31(7 ітерація зовн. циклу) | i=6 |
| 32 | fl=false |
| 33 | dob=1 |
| 34(внутрішній цикл виконав роботу) | fl=true; dob=-0.7448, num=7 |
| 35(8 ітерація зовн. циклу) | i=7 |
| 36 | fl=false |
| 37 | dob=1 |
| 38(внутрішній цикл виконав роботу) | fl=true; dob=0.9, num=8 |
| 39 | return 8 |
| 40 | sizeOfArr=8 |
|  | **виклик процедури** printArray(arr, sizeOfArr) |

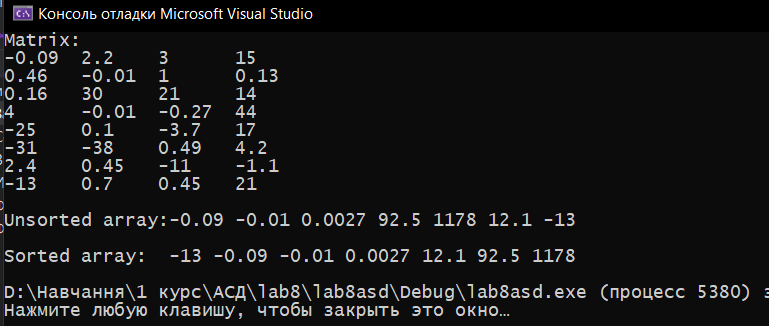
|  |  |
| --- | --- |
|  | виклик процедури sortArray(arr, sizeOfArr) |
|  | **виклик процедури** printArray(arr, sizeOfArr) |

**-0.48 0.011 1.9 9 3.08 -2.304 -0.7448 0.9**

**-2.304 -0.7448 -0.48 0.011 0.9 1.9 3.08 9**

|  |  |
| --- | --- |
|  | кінець |

**Тестування програми**



**Висновок:** Отже, у результаті даної роботи я створив алгоритм та написав програму, яка знаходить добутки від’ємних елементів у рядках матриці, записує їх в одновимірний масив та сортує цей масив обміном за зростанням. Заповнював матрицю випадковими значеннями у діапазоні від -50 до 50, з можливою деякою дробовою частиною, отриманими за формулою matrix[i][j]=(rand()%101-50)\*10^(-rand()%3). Процедура fillMatrix(matrix[][4]) заповнює матрицю дійсними значеннями, з допомогою вкладених циклів. Виклик функції arrayFromMatrix(matrix, arr) ініціює одновимірний масив добутками від’ємних значень елементів рядків двовимірного масиву та повертає його довжину. Процедура sortArray(arr, sizeOfArr) сортує масив обміном за зростанням(є вкладені цикли). Описав алгоритм за допомогою псевдокоду та блок-схеми, реалізував алгоритм на мові програмування C++. Виконав випробування алгоритму та тестування програми.