**16. Задачі, розв’язання яких потребує двовимірних масивів. Задачі, що провокують, але не вимагають використання двовимірних масивів. Приклади.**

*Задачі, які потребують використання двовимірних масивів – це задачі, виконання яких потребує проходу матриці по рядкам або стовбцям, тобто уявлення матриці саме як двовимірного масиву:*

- Транспонування матриці.

Транспонування буває по головній або побічній діагоналі. Для транспонування матриці треба уявляти її у вигляді двовимірного масиву, тобто пересуватись по елементам матриці за двома індексами. Є багато способів транспонування матриць(і квадратних і прямокутних), але всі вони реалізовані за допомогою двовимірного масиву(або імітуванням двовимірного масиву через одновимірний масив).

- Добуток двох матриць.

Для реалізації добутку матриць через формулу з алгебри  потребується три двовимірних масиви. Для множення матриць треба пересуватись по рядкам та стовбцям матриць, тобто реалізація матриці за допомогою двовимірного масиву необхідна.

- Сортування рядків або стовбців матриці.

Наприклад, сортування рядків матриці за зростанням: даний список призвіщ, відсортувати їх за алфавітом. Для цього цей список реалізується як двовимірний масив символів, а розташування його рядків сортується за зростанням.

Приклад транспонування:

#include <iostream>

using namespace std;

int main()

{

const int n = 4;

int arr[n][n] =

{

{1, 2, 3, 4},

{5, 6, 7 ,8},

{9, 10, 11, 12},

{13, 14, 15, 16}

};

int t;

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

{

for (int j = i; j < n; ++j)

{

t = arr[i][j];

arr[i][j] = arr[j][i];

arr[j][i] = t;

}

}

// вивід

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

{

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

{

cout << arr[i][j] << " ";

}

cout << endl;

}

return 0;

}

*Задачі, що провокують, але не вимагають використання двовимірних масивів – це задачі, які перейшли із задач для одновимірних масивів, наприклад:*

- Задана будь-яка матриця, треба вивести на екран всі елементи матриці в зворотному порядку.

Можна використати одновимірний масив: користувач вводить всі елементи матриці, а потім програма їх виводить на екран у зворотному порядку.

- Задана будь-яка матриця, треба відсортувати всі елементи за зростанням/спаданням тощо.

Можна також використати одновимірний масив для збереження всіх елементів матриці, а потім відсортувати його будь-яким алгоритмом сортування.

- Reverse. Задана матриця, треба переставити елементи цієї матриці у зворотному порядку.

Записати елементи матриці у одновимірний масив, а потім через функцію reverse переставити всі його елементи у зворотному порядку.

Приклад виводу матриці у зворотному порядку:

#include <iostream>

using namespace std;

int main()

{

const int n = 16;

int arr[n];

/\*ввести:

1 2 3 4

5 6 7 8

9 10 11 12

13 14 15 16

\*/

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

{

cin >> arr[i];

}

// вивід

for (int i = n - 1; i > -1; i--)

{

cout << arr[i] << " ";

if ((i) % 4 == 0)

{

cout << endl;

}

}

return 0;

}