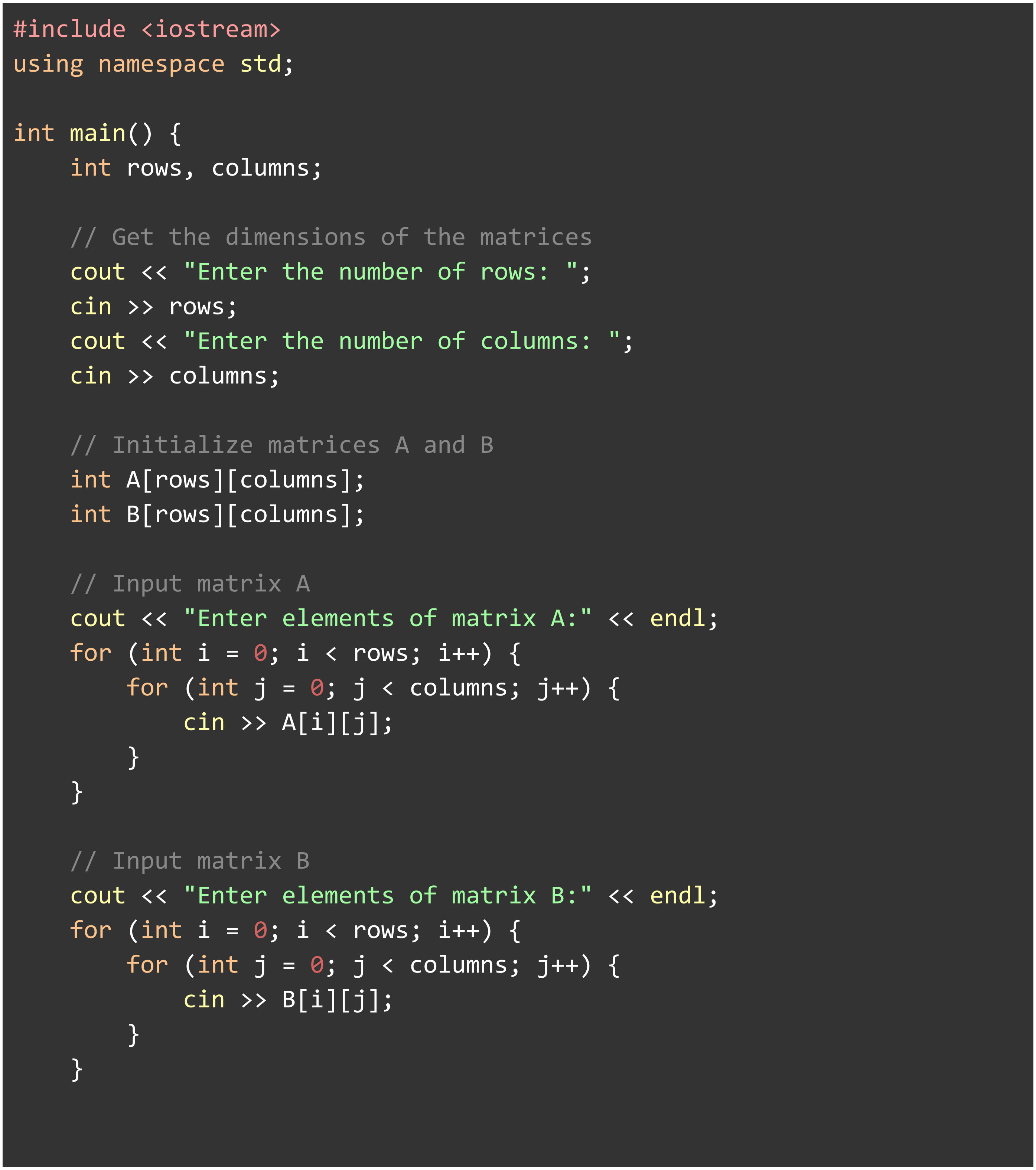
Day 43/180 Introduction to 2D array in c++

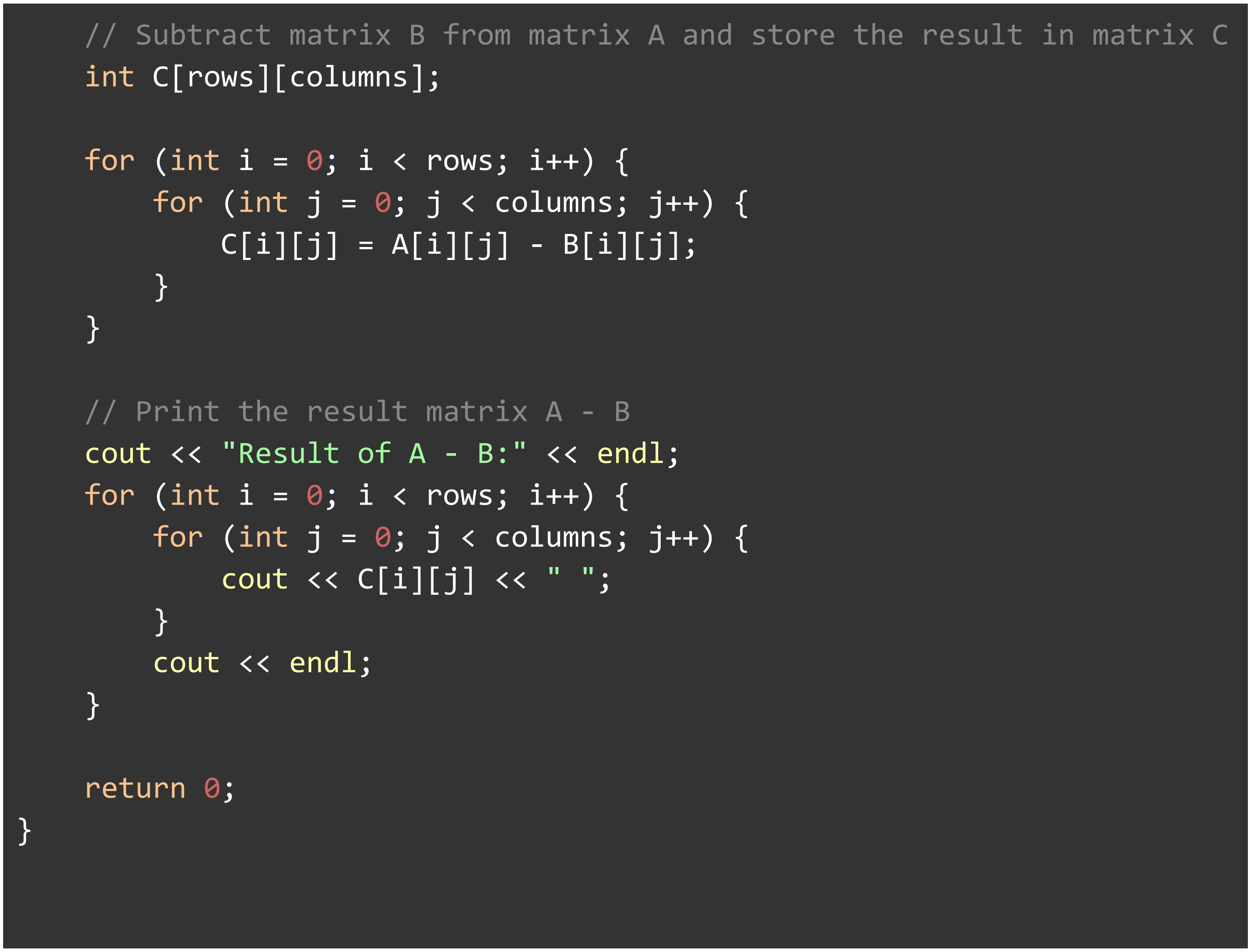
1: Print sum of each column in 2D array.
Solution:

```
#include <iostream>
using namespace std;

int main() {
    // Sample 2D array
    int array_2d[3][3] = {
        {1, 2, 3},
        {4, 5, 6},
        {7, 8, 9}
    };

    int num_rows = 3;    // Number of rows
    int num_columns = 3; // Number of columns

    // Initialize an array to store the sum of each column
    int column_sums[num_columns] = {0};

    // Calculate the sum of each column
    for (int i = 0; i < num_rows; i++) {
        for (int j = 0; j < num_columns; j++) {
            column_sums[j] += array_2d[i][j];
        }
    }

    // Print the sum of each column
    for (int j = 0; j < num_columns; j++) {
        cout << "Sum of column " << j + 1 << ": " << column_sums[j] << endl;
    }

    return 0;
}
```

2: Given 2 matrices A and B, Print A-B.
Solution:



```cpp
#include <iostream>
using namespace std;

int main() {
    int rows, columns;

    // Get the dimensions of the matrices
    cout << "Enter the number of rows: ";
    cin >> rows;
    cout << "Enter the number of columns: ";
    cin >> columns;

    // Initialize matrices A and B
    int A[rows][columns];
    int B[rows][columns];

    // Input matrix A
    cout << "Enter elements of matrix A:" << endl;
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < columns; j++) {
            cin >> A[i][j];
        }
    }

    // Input matrix B
    cout << "Enter elements of matrix B:" << endl;
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < columns; j++) {
            cin >> B[i][j];
        }
    }
```

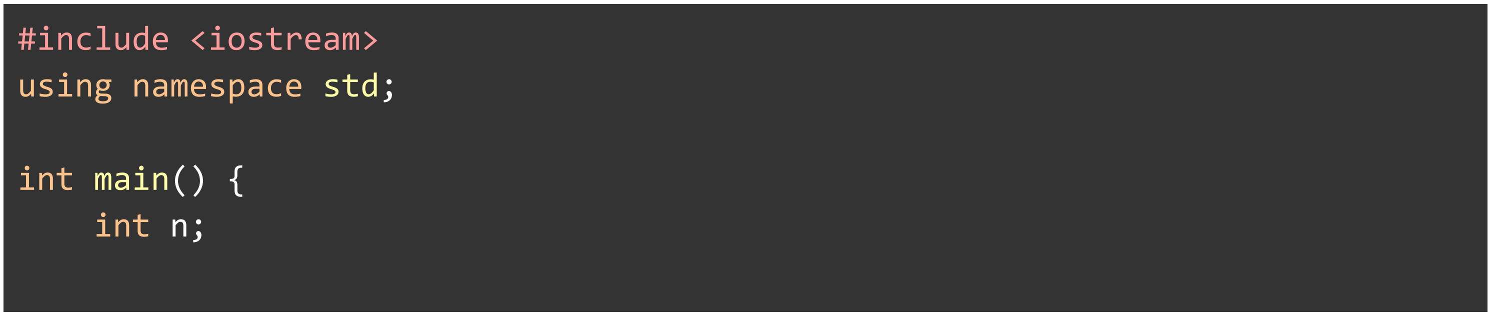


```
    // Subtract matrix B from matrix A and store the result in matrix C
    int C[rows][columns];

    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < columns; j++) {
            C[i][j] = A[i][j] - B[i][j];
        }
    }

    // Print the result matrix A - B
    cout << "Result of A - B:" << endl;
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < columns; j++) {
            cout << C[i][j] << " ";
        }
        cout << endl;
    }

    return 0;
}
```

3: Given a matrix of size n*n, Print sum of diagonal element.

Ex: 1 2 3
4 5 6
7 8 9

Solution for getting both diagonal sum:



```
#include <iostream>
using namespace std;

int main() {
    int n;
    // Get the size of the square matrix
    cout << "Enter the size of the square matrix (n x n): ";
    cin >> n;

    // Initialize the square matrix
    int matrix[n][n];

    // Input the elements of the matrix
    cout << "Enter the elements of the matrix:" << endl;
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            cin >> matrix[i][j];
        }
    }

    // Calculate the sum of the main diagonal elements
    int main_diagonal_sum = 0;
    for (int i = 0; i < n; i++) {
        main_diagonal_sum += matrix[i][i];
    }

    // Calculate the sum of the other diagonal elements
    int other_diagonal_sum = 0;
    for (int i = 0; i < n; i++) {
        other_diagonal_sum += matrix[i][n - 1 - i];
    }

    // Print the sums of both diagonal elements
    cout << "Sum of main diagonal elements: " << main_diagonal_sum << endl;
    cout << "Sum of other diagonal elements: " << other_diagonal_sum <<
endl;

    return 0;
}
```

Its answer: 1+5+9 , 3+5+7, So the total sum will be 1+5+9+3+5+7 = 30. Here we can see that 5 is included 2 times, so we should include it only 1 time so the final answer will be, 30-5 = 25. So the final answer will be 25.

Solution of 3rd Question:

```
#include <iostream>
using namespace std;

int main() {
    int n;

    // Get the size of the square matrix
    cout << "Enter the size of the square matrix (n x n): ";
    cin >> n;

    // Initialize the square matrix
    int matrix[n][n];

    // Input the elements of the matrix
    cout << "Enter the elements of the matrix:" << endl;
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            cin >> matrix[i][j];
        }
    }

    // Initialize the sum of diagonal elements
    int diagonal_sum = 0;

    // Calculate the sum of the main diagonal elements and exclude the center
element if n is odd
    for (int i = 0; i < n; i++) {
        diagonal_sum += matrix[i][i];
    }
    // Calculate the sum of the other diagonal elements
    for (int i = 0; i < n; i++) {
        diagonal_sum += matrix[i][n - 1 - i];
    }

    // If n is odd, subtract the center element once
    if (n % 2 == 1) {
        int center = n / 2;
        diagonal_sum -= matrix[center][center];
    }

    // Print the final sum of diagonal elements
    cout << "Sum of diagonal elements: " << diagonal_sum << endl;

    return 0;
}
```

4: What is the column major order?

Solution:

```
Column-major order is a way to store and access elements in a multi-dimensional
array or matrix by columns rather than by rows. In this order, elements of a
column are stored contiguously in memory. Let's consider a 3x3 matrix and
represent it in column-major order:

Suppose we have the following 3x3 matrix:

| 1  2  3 |
| 4  5  6 |
| 7  8  9 |
In column-major order, the elements are stored column by column. So, the elements
would be stored in memory as follows:

1, 4, 7, 2, 5, 8, 3, 6, 9
```

5: Largest Element: Find and print the largest element in the 2D array.

```
#include <iostream>
using namespace std;

int main() {
    int rows, columns;

    // Get the dimensions of the 2D array
    cout << "Enter the number of rows: ";
    cin >> rows;
    cout << "Enter the number of columns: ";
    cin >> columns;

    // Initialize the 2D array
    int array_2d[rows][columns];

    // Input the elements of the 2D array
    cout << "Enter the elements of the 2D array:" << endl;
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < columns; j++) {
            cin >> array_2d[i][j];
        }
    }

    // Initialize the maximum element to the first element of the array
    int max_element = array_2d[0][0];

    // Find the largest element in the 2D array
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < columns; j++) {
            if (array_2d[i][j] > max_element) {
                max_element = array_2d[i][j];
            }
        }
    }
    // Print the largest element
    cout << "The largest element in the 2D array is: " << max_element << endl;

    return 0;
}
```

6: Smallest Element: Find and print the smallest element in the 2D arrays.

```
#include <iostream>
using namespace std;

int main() {
    int rows, columns;

    // Get the dimensions of the 2D array
    cout << "Enter the number of rows: ";
    cin >> rows;
    cout << "Enter the number of columns: ";
    cin >> columns;

    // Initialize the 2D array
    int array_2d[rows][columns];

    // Input the elements of the 2D array
    cout << "Enter the elements of the 2D array:" << endl;
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < columns; j++) {
            cin >> array_2d[i][j];
        }
    }

    // Initialize the minimum element to the first element of the array
    int min_element = array_2d[0][0];
    // Find the smallest element in the 2D array
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < columns; j++) {
            if (array_2d[i][j] < min_element) {
                min_element = array_2d[i][j];
            }
        }
    }

    // Print the smallest element
    cout << "The smallest element in the 2D array is: " << min_element <<
endl;

    return 0;
}
```