Trạng thái	Đã xong
Bắt đầu vào lúc	Chủ Nhật, 30 tháng 3 2025, 10:26 PM
Kết thúc lúc	Chủ Nhật, 30 tháng 3 2025, 10:32 PM
Thời gian thực hiện	5 phút 54 giây

Câu hỏi 1

Đúng

Mô tả tiếng Việt:

Cho mảng 2 chiều chứa các số nguyên, kích thước M x N.

Hiện thực hàm:

int findMaxColumn(int arr[][1000], int row, int col);

Trong đó; `arr`, `row` và `col` lần lượt là mảng 2 chiều, số hàng và số cột của mảng. Tìm chỉ số của cột có tổng tất cả các phần tử lớn nhất.

Lưu ý: Cột đầu tiên được đánh chỉ số 0. Nếu có nhiều hơn một cột có tổng lớn nhất, ta chọn cột có chỉ số lớn nhất.

Ghi chú: (Các) thư viện `iostream` và `climits` đã được khai báo, và `namespace std` đã được sử dụng.

English version:

Given a two-dimensional array whose each element is integer, its size is M x N.

Implement the following function:

int findMaxColumn(int arr[][1000], int row, int col);

Where `arr`, `row` and `col` are the given two-dimensional array, its number of rows and its number of columns. Find the index of the column which has the greatest sum of all elements on it.

Note: The first column of the given array is numbered by 0. If there are more than one column whose sum is the greatest, choose the column with the greatest index.

Note: Libraries `iostream` and `climits` have been imported, and `namespace std` has been used.

For example:

Test	Result
`int arr[][1000] = {{-44,64,-6},{87,92,-19},{-92,53,-38},{-39,-92,21}};` `cout << findMaxColumn(arr, 4, 3);`	1
`int arr[][1000] = {{-92,78,-2,-58,-37},{44,-4,30,-69,22}};` `cout << findMaxColumn(arr, 2,5);`	1

Answer: (penalty regime: 0 %)

Reset answer

```
// Hàm tìm chỉ số của cột có tổng lớn nhất
int findMaxColumn(int arr[][1000], int row, int col) {
    int maxSum = INT_MIN;  // Khởi tạo tổng lớn nhất bằng giá trị nhỏ nhất có thể
    int maxColIndex = -1;  // Khởi tạo chỉ số cột có tổng lớn nhất

    // Duyệt qua từng cột
    for (int j = 0; j < col; j++) {
        int currentSum = 0;  // Khởi tạo tổng của cột hiện tại

        // Tính tổng các phần tử trong cột j
        for (int i = 0; i < row; i++) {
            currentSum += arr[i][j];
        }

        // Cập nhật cột có tổng lớn nhất
        // Lưu ý: Nếu có cùng tổng, chọn cột có chỉ số lớn hơn (>=)
        if (currentSum >= maxSum) {
            maxSum = currentSum;
            maxColIndex = j;
        }
    }

    return maxColIndex;
}
```

	Test	Expected	Got	
✓	`int arr[][1000] = {{-44,64,-6},{87,92,-19},{-92,53,-38},{-39,-92,21}};` `cout << findMaxColumn(arr, 4, 3);`	1	1	✓
✓	`int arr[][1000] = {{-92,78,-2,-58,-37},{44,-4,30,-69,22}};` `cout << findMaxColumn(arr, 2,5);`	1	1	✓

Passed all tests! ✓

Câu hỏi **2**

Đúng

Mô tả tiếng Việt:

Cho mảng 2 chiều chứa các số nguyên, kích thước N x N.

Hiện thực hàm:

int diagonalProduct(int arr[][1000], int row, int col);

Trong đó; `arr`, `row` và `col` lần lượt là mảng 2 chiều, số hàng và số cột của mảng. Tìm tích của tất cả các phần tử trong đường chéo chính của mảng.

Ghi chú: (Các) thư viện `iostream`, và `string` đã được khai báo, và `namespace std` đã được sử dụng.

English version:

Given a two-dimensional array whose each element is integer, its size is N x N.

Implement the following function:

int diagonalProduct(int arr[][1000], int row, int col);

Where `arr`, `row` and `col` are the given two-dimensional array, its number of rows and its number of columns. Find the product of all elements on the main diagonal of this array.

Note: Libraries `iostream`, and `string` have been imported, and `namespace std` has been used.

For example:

Test	Result
`int arr[][1000] = {{-45,18,-37},{-2,-31,24},{-48,-33,-48}};` `cout << diagonalProduct(arr,3,3);`	-66960
`int arr[][1000] = {{-11,44,18,33},{-34,-9,-42,-42},{47,-26,4,-8},{-35,11,-34,-19}};` `cout << diagonalProduct(arr,4,4);`	-7524

Answer: (penalty regime: 0 %)

Reset answer

```
1
2
3  // Hàm tính tích các phần tử trên đường chéo chính
4  int diagonalProduct(int arr[][1000], int row, int col) {
5      // Khởi tạo biến tích bằng 1
6      int product = 1;
7
8      // Duyệt qua các phần tử trên đường chéo chính
9      // Đường chéo chính có các phần tử nằm ở vị trí (i,i)
10     int size = min(row, col); // Đảm bảo không vượt quá kích thước mảng
11
12     for (int i = 0; i < size; i++) {
13         // Nhân tích với phần tử hiện tại
14         product *= arr[i][i];
15     }
16
17     return product;
18 }
```

	Test	Expected	Got	
✓	`int arr[][1000] = {{-45,18,-37},{-2,-31,24},{-48,-33,-48}};` `cout << diagonalProduct(arr,3,3);`	-66960	-66960	✓
✓	`int arr[][1000] = {{-11,44,18,33},{-34,-9,-42,-42},{47,-26,4,-8},{-35,11,-34,-19}};` `cout << diagonalProduct(arr,4,4);`	-7524	-7524	✓

Passed all tests! ✓

Câu hỏi **3**

Đúng

Mô tả tiếng Việt:

Cho mảng 2 chiều chứa các số nguyên, kích thước N x N.

Hiện thực hàm:

bool isSymmetric(int arr[][1000], int row, int col);

Trong đó; `arr`, `row` và `col` lần lượt là mảng 2 chiều, số hàng và số cột của mảng. Một ma trận được gọi là đối xứng nếu với mọi i, j; giá trị của phần tử ở hàng i, cột j luôn bằng giá trị của phần tử ở hàng j, cột i. Kiểm tra xem mảng này có phải là một ma trận đối xứng hay không; trả về **true** nếu mảng này là ma trận đối xứng; ngược lại, trả về **false**.

Ghi chú: (Các) thư viện `iostream` và `string` đã được khai báo, và `namespace std` đã được sử dụng.

English version:

Given a two-dimensional array whose each element is integer, its size is N x N.

Implement the following function:

bool isSymmetric(int arr[][1000], int row, int col);

Where `arr`, `row` and `col` are the given two-dimensional array, its number of rows and its number of columns. A matrix is called as **symmetric matrix** if for all i, j; the value of the element on row i, column j is equal to the value of the element on row j, column i. Check whether the given array is **symmetric matrix** or not; return **true** if it is, otherwise return **false**.

Note: Libraries `iostream` and `string` have been imported, and `namespace std` has been used.

For example:

Test	Result
`int arr[][1000] = {{1,4,6}, {4,5,3}, {6,3,9}};` `cout << isSymmetric(arr,3,3);`	1
`int arr[][1000] = {{1,9,6}, {4,5,3}, {6,3,9}};` `cout << isSymmetric(arr,3,3);`	0

Answer: (penalty regime: 0 %)

Reset answer

```
// Hàm kiểm tra ma trận đối xứng
bool isSymmetric(int arr[][1000], int row, int col) {
    // Điều kiện cần để ma trận đối xứng là kích thước ma trận phải là vuông
    if (row != col) {
        return false;
    }

    // Kiểm tra tính đối xứng: arr[i][j] = arr[j][i] với mọi i, j
    for (int i = 0; i < row; i++) {
        for (int j = 0; j < col; j++) {
            if (arr[i][j] != arr[j][i]) {
                return false; // Nếu có một cặp không đối xứng, trả về false
            }
        }
    }

    // Nếu tất cả các cặp đều đối xứng, trả về true
    return true;
}
```

	Test	Expected	Got	
✓	`int arr[][1000] = {{1,4,6}, {4,5,3}, {6,3,9}};` `cout << isSymmetric(arr,3,3);`	1	1	✓
✓	`int arr[][1000] = {{1,9,6}, {4,5,3}, {6,3,9}};` `cout << isSymmetric(arr,3,3);`	0	0	✓

Passed all tests! ✓

Câu hỏi **4**

Đúng

Mô tả tiếng Việt:

Cho mảng 2 chiều chứa các số nguyên, kích thước M x N.

Hiện thực hàm:

int diagonalDiff(int arr[][1000], int row, int col, int x, int y);

Trong đó; `arr`, `row` và `col` lần lượt là mảng 2 chiều, số hàng, số cột của mảng; `x` và `y` biểu thị ô có số hàng là x và số cột là y trong mảng đã cho (`0≤x<row` và `0≤y<col`). Tổng của một đường chéo là tổng tất cả các phần tử nằm trên đường chéo đó. Tìm giá trị tuyệt đối của hiệu giữa hai đường chéo đi qua ô có số hàng x và số cột y.

Ghi chú: (Các) thư viện `iostream`, `vector` và `string` đã được khai báo, và `namespace std` đã được sử dụng.

English version:

Given a two-dimensional array whose each element is an integer, its size is M x N.

Implement the following function:

int diagonalDiff(int arr[][1000], int row, int col, int x, int y);

Where `arr`, `row` and `col` are the given two-dimensional array, its number of rows and its number of columns. ; `x` and `y` represent the cell whose index of the row is x and index of the column is y (`0≤x<row` và `0≤y<col`). The sum of a diagonal is the sum of all elements on it. Find the absolute value of the difference between the sums of two diagonal containing the cell which is represented by x and y of the given array.

Note: Libraries `iostream`, `vector`, and `string` have been imported, and `namespace std` has been used.

For example:

Test	Result
`int arr[][1000] = {{55,85,47,58},{31,4,60,67},{94,69,71,73},{51,62,64,90}};` `cout << diagonalDiff(arr,4,4,1,2);`	20
`int arr[][1000] = {{88,72,65,37},{82,84,34,12},{74,46,88,44}};` `cout << diagonalDiff(arr,3,4,1,0);`	26

Answer: (penalty regime: 0 %)

Reset answer

```
/**
 * Hàm tính hiệu giá trị tuyệt đối giữa tổng hai đường chéo đi qua điểm (x,y)
 * @param arr: mảng 2 chiều chứa các số nguyên
 * @param row: số hàng của mảng
 * @param col: số cột của mảng
 * @param x: chỉ số hàng của ô cần xét
 * @param y: chỉ số cột của ô cần xét
 * @return: giá trị tuyệt đối của hiệu giữa tổng hai đường chéo
 */
int diagonalDiff(int arr[][1000], int row, int col, int x, int y) {
    // Tính tổng của đường chéo chính (từ trái trên xuống phải dưới)
    int mainDiagonalSum = 0;
    int r = x, c = y;

    // Di chuyển lên trái để tìm điểm bắt đầu của đường chéo chính
    // Tiếp tục di chuyển cho đến khi gặp biên trên hoặc biên trái của ma trận
    while (r > 0 && c > 0) {
        r--;
        c--;
    }

    // Từ điểm bắt đầu, duyệt và tính tổng toàn bộ đường chéo
    // Dừng lại khi gặp biên dưới hoặc biên phải của ma trận
    while (r < row && c < col) {
        mainDiagonalSum += arr[r][c];
        r++;
        c++;
    }

    // Tính tổng của đường chéo phụ (từ phải trên xuống trái dưới)
    int antiDiagonalSum = 0;
    r = x;  // Reset về vị trí ban đầu
    c = y;
```

```
34
35      // Di chuyển lên phải để tìm điểm bắt đầu của đường chéo phụ
36      // Tiếp tục di chuyển cho đến khi gặp biên trên hoặc biên phải của ma trận
37      while (r > 0 && c < col - 1) {
38          r--;
39          c++;
40      }
41
42      // Từ điểm bắt đầu, duyệt và tính tổng toàn bộ đường chéo phụ
43      // Dừng lại khi gặp biên dưới hoặc biên trái của ma trận
44      while (r < row && c >= 0) {
45          antiDiagonalSum += arr[r][c];
46          r++;
47          c--;
48      }
49
50      // Trả về giá trị tuyệt đối của hiệu giữa hai tổng
51      return abs(mainDiagonalSum - antiDiagonalSum);
52  }
```

	Test	Expected	Got	
✓	`int arr[][1000] = {{55,85,47,58},{31,4,60,67},{94,69,71,73},{51,62,64,90}};` `cout << diagonalDiff(arr,4,4,1,2);`	20	20	✓
✓	`int arr[][1000] = {{88,72,65,37},{82,84,34,12},{74,46,88,44}};` `cout << diagonalDiff(arr,3,4,1,0);`	26	26	✓

Passed all tests! ✓