Đã bắt đầu vào lúc	Thứ sáu, 24 Tháng mười một 2023, 12:32 PM
Tình trạng	Đã hoàn thành
Hoàn thành vào lúc	Chủ nhật, 26 Tháng mười một 2023, 5:04 PM
Thời gian thực hiện	2 ngày 4 giờ
Điểm	4,00/4,00
Điểm	**10,00** của 10,00 (**100**%)

Câu hỏi 1

Chính xác

Điểm 1,00 của 1,00

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`

Test	Result
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

```
int findMaxColumn(int arr[][1000], int row, int col) {
    // Initialize variables to keep track of the maximum sum and its corresponding column index
    int maxSum = INT_MIN;
    int maxColumnIndex = -1;

    // Iterate through each column
    for (int j = 0; j < col; ++j) {
        // Calculate the sum of the current column
        int columnSum = 0;
        for (int i = 0; i < row; ++i) {
            columnSum += arr[i][j];
        }

        // Update maxSum and maxColumnIndex if the current column has a greater sum
        if (columnSum > maxSum) {
            maxSum = columnSum;
            maxColumnIndex = j;
        }
    }

    // Return the index of the column with the greatest sum
    return maxColumnIndex;
```

	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	✔

	Test	Expected	Got	
✔	`int arr[][1000] = {{-92,78,-2,-58,-37},{44,-4,30,-69,22}};` `cout << findMaxColumn(arr, 2,5);`	1	1	✔

Passed all tests! ✔

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi 2

Chính xác

Điểm 1,00 của 1,00

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

```
int diagonalProduct(int arr[][1000], int row, int col) {
    // Ensure the matrix is square
    if (row != col) {
        std::cout << "Matrix is not square. Cannot compute diagonal product." << std::endl;
        return 0;  // You might want to handle this case differently, depending on your requirements.
    }

    // Initialize product to 1
    int product = 1;

    // Calculate the product of elements on the main diagonal
    for (int i = 0; i < row; ++i) {
        product *= arr[i][i];
    }

    return product;
}
```

	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! ✔

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi 3

Chính xác

Điểm 1,00 của 1,00

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

```
bool isSymmetric(int arr[][1000], int row, int col) {
    // A matrix is symmetric if it is square (row == col)
    // and if arr[i][j] == arr[j][i] for all i, j
    if (row != col) {
        return false; // Not a square matrix, cannot be symmetric
    }

    for (int i = 0; i < row; ++i) {
        for (int j = 0; j < col; ++j) {
            if (arr[i][j] != arr[j][i]) {
                return false; // Element at (i, j) is not equal to element at (j, i)
            }
        }
    }

    // If the loop completes without returning false, the matrix is symmetric
    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! ✔

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi 4

Chính xác

Điểm 1,00 của 1,00

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

```
int diagonalDiff(int arr[][1000], int row, int col, int x, int y) {
    int diagonal1 = 0; // left
    int diagonal2 = 0; // right

    // Diagonal from top-left to bottom-right
    int i = x;
    int j = y;
    while (i >= 0 && j >= 0) {
        diagonal1 += arr[i][j];
        i--;
        j--;
    }

    i = x + 1;
    j = y + 1;
    while (i < row && j < col) {
        diagonal1 += arr[i][j];
        i++;
        j++;
    }

    // Diagonal from top-right to bottom-left
```

	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! ✔

Chính xác

Điểm cho bài nộp này: 1,00/1,00.