

Trạng thái	Đã xong
Bắt đầu vào lúc	Thứ Tư, 30 tháng 4 2025, 5:17 PM
Kết thúc lúc	Thứ Tư, 30 tháng 4 2025, 6:10 PM
Thời gian thực hiện	53 phút 26 giây

Câu hỏi 1

Đúng

Đạt điểm 1,00

Mô tả tiếng Việt:

Hãy hiện thực hàm readArray() được khai báo như sau:

```
int** readArray()
```

Hàm này sẽ đọc dữ liệu cho một ma trận 2 chiều, mỗi chiều có 10 phần tử. Các phần tử của ma trận sẽ được nhập vào từ bàn phím (từ phần tử a[0][0] cho đến a[9][9]). Tuy nhiên nếu phần tử a[i][j] được nhập là 0 thì tất cả các phần tử còn lại trên hàng (a[i][k], j<k<10) đều được tự động gán là 0, chương trình sẽ đọc tiếp phần tử a[i+1][0] từ bàn phím. Hàm readArray sẽ trả về một con trỏ tới mảng 2 chiều đã nhập này.

**Đầu vào:** Các phần tử có trong mảng 2 chiều, mỗi phần tử là một số nguyên dương có giá trị không vượt quá 1000.

**Đầu ra:** Con trỏ tới mảng 2 chiều vừa tạo

English version:

Implement the function readArray() that is declared as below syntax:

```
int** readArray()
```

The function reads a two-dimensional matrix each of which consists of 10 elements. These elements are entered from the keyboard (from a[0][0] to a[9][9]). If a[i][j] is assigned to 0, all remained element of the row (a[i][k], j<k<10) will automatically assigned to 0, and the function will continue to input the next-row element from the keyboard. Moreover, this function also returns a pointer which points to the two-dimensional matrix just entered.

**Input:** The positive integer matrix's elements which not surpass 1000.

**Output:** The pointer that points to the two-dimensional matrix just entered.

For example:

Test	Input	Result
1	1 2 3 4 5 6 7 8 9 10 0 1 0 2 0 3 0 4 5 0 6 7 0 8 0 9 0 10 11 12 13 14 0	1 2 3 4 5 6 7 8 9 10 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 3 0 0 0 0 0 0 0 0 0 4 5 0 0 0 0 0 0 0 0 6 7 0 0 0 0 0 0 0 0 8 0 0 0 0 0 0 0 0 0 9 0 0 0 0 0 0 0 0 0 10 11 12 13 14 0 0 0 0 0
2	0 0 0 0 0 0 0 0 0 0	



0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Answer: (penalty regime: 0 %)

Reset answer

```
1 int** readArray()
2 {
3     //TODO
4     int **arr = new int*[10];
5     for(int i = 0; i < 10; ++i){
6         arr[i] = new int[10];
7         for(int j = 0; j < 10; ++j){
8             cin >> arr[i][j];
9             if(arr[i][j] == 0){
10                for(int k = j + 1; k < 10; ++k){
11                    arr[i][k] = 0;
12                }
13                break;
14            }
15        }
16    }
17    return arr;
18 }
```

	Test	Input	Expected	Got	
✓	1	1 2 3 4 5 6 7 8 9 10 0 1 0 2 0 3 0 4 5 0 6 7 0 8 0 9 0 10 11 12 13 14 0	1 2 3 4 5 6 7 8 9 10 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 3 0 0 0 0 0 0 0 0 0 4 5 0 0 0 0 0 0 0 0 6 7 0 0 0 0 0 0 0 0 8 0 0 0 0 0 0 0 0 0 9 0 0 0 0 0 0 0 0 0 10 11 12 13 14 0 0 0 0 0	1 2 3 4 5 6 7 8 9 10 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 3 0 0 0 0 0 0 0 0 0 4 5 0 0 0 0 0 0 0 0 6 7 0 0 0 0 0 0 0 0 8 0 0 0 0 0 0 0 0 0 9 0 0 0 0 0 0 0 0 0 10 11 12 13 14 0 0 0 0 0	✓
✓	2	0 0 0 0 0 0 0 0 0 0 0 0 0	0 0	0 0	✓

Passed all tests! ✓

Câu hỏi 2

Đúng

Đạt điểm 1,00

Mô tả tiếng Việt:

Hiện thực hàm `void addElement(int*& arr, int n, int val, int index)`

nhận vào một mảng động `arr` có chính xác `n` phần tử và tiến hành  
chèn giá trị `val` vào vị trí thứ `index`.

Đầu vào: Mảng một chiều `arr` có kích thước `n`, giá trị cần chèn `val` và  
vị trí cần chèn `index`.

Đầu ra: Mảng `arr` sau khi chèn.

Lưu ý: Việc chèn phần tử vào mảng động phải được thực hiện bằng cách giải phóng mảng cũ có n phần tử và cấp phát mảng mới có n+1 phần tử.

**English version:**

Implement the function **void addElement(int\*& arr, int n, int val, int index)** that inputs a dynamic array, arr, consisting of exactly n elements and insert a value, val, into the a specific position, index.

Input: The n-size dynamic array needs to be inserted the value, val, into the specific position, index.

Output: The dynamic array after insert.

Note: Insertion of elements into a dynamic array must be executed by freeing the old array and allocating new memory for the new one.

**For example:**

Test	Input	Result
1	2 2 3 1 1	2 1 3
2	2 2 3 1 2	2 3 1

**Answer:** (penalty regime: 0 %)

Reset answer

```
1 void addElement(int*& arr, int n, int val, int index) {
2     // TODO
3     if(index >= 0 && index <= n){
4         int *narr = new int[n+1];
5
6         for(int i = 0; i < index; ++i){
7             narr[i] = arr[i];
8         }
9         narr[index] = val;
10
11        for(int i = index; i < n; ++i){
12            narr[i+1] = arr[i];
13        }
14        delete [] arr;
15        arr = narr;
16    }
17 }
18 }
```

	Test	Input	Expected	Got	
✓	1	2 2 3 1 1	2 1 3	2 1 3	✓

Passed all tests! ✓

Hiện thực hàm `int* flatten(int** matrix, int r, int c)` trả về một mảng một chiều được “làm phẳng” từ mảng hai chiều có kích thước  $r \times c$  (bằng cách nối các hàng của mảng hai chiều lại với nhau).

Đầu vào: Mảng hai chiều có kích thước  $r \times c$ .

Đầu ra: Mảng một chiều sau khi được “làm phẳng” từ mảng hai chiều đầu vào.

English version:

Implement the function `int* flatten(int** matrix, int r, int c)` that returns a one-dimensional array flattened from a two-dimensional matrix of size  $r \times c$  (by concatenating all the matrix rows).

Input: The two-dimensional matrix of size  $r \times c$

Output: The one-dimensional array flattened from the previous two-dimensional matrix.

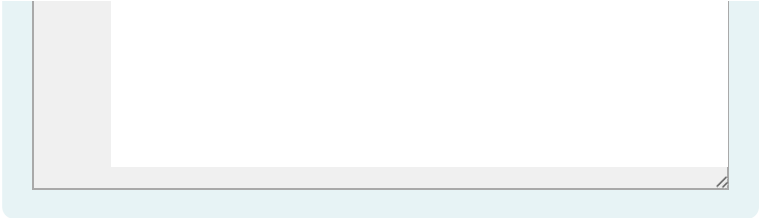
For example:

Test	Input	Result
1	2 3 1 2 3 4 5 6	1 2 3 4 5 6
2	2 3 1 2 3 4 0 0	1 2 3 4 0 0
3	3 3 1 2 3 4 5 6 2 9 -99	1 2 3 4 5 6 2 9 -99
4	3 4 1 2 3 4 4 5 6 0 -1 8 8 100	1 2 3 4 4 5 6 0 -1 8 8 100
5	4 4 1 2 4 4 4 5 3 0 2 5 1 6 7 7 8 4	1 2 4 4 4 5 3 0 2 5 1 6 7 7 8 4
1	4 1 1 4 2 3	1 4 2 3
7	1 4 1 2 4 4	1 2 4 4

Answer: (penalty regime: 0 %)

Reset answer

```
1 int* flatten(int** matrix, int r, int c) {
2     //TODO
3
4     int *arr = new int[r * c];
5     for(int i = 0; i < r; ++i){
6         for(int j = 0; j < c; ++j){
7             arr[i * c + j] = matrix[i][j];
8         }
9     }
10    return arr;
11 }
```



	Test	Input	Expected	Got	
✓	1	2 3 1 2 3 4 5 6	1 2 3 4 5 6	1 2 3 4 5 6	✓

Passed all tests! ✓

Câu hỏi 4

Đúng

Đạt điểm 1,00

Mô tả tiếng Việt:

Hiện thực hàm **char\* concatStr(char\* str1, char\* str2)** trả về một chuỗi là kết quả sau khi nối 2 chuỗi str1 và str2 thành một chuỗi duy nhất.

Đầu vào: Hai chuỗi str1 và str2.

Đầu ra: Chuỗi được nối từ 2 chuỗi con str1 và str2.

Lưu ý: Không được phép sử dụng các hàm hỗ trợ của thư viện string và string.h cho bài tập này.

English version:

Implement the function **char\* concatStr(char\* str1, char\* str2)** that return a string merged from two smaller string str1 and str2.

Input: Two string str1 and str2.

Output: The string merged from two smaller string str1 and str2.

Note: The string and string.h library are not allowed to use for this exercise.

For example:

Test	Result
<pre>char s1[] = "Hello, "; char s2[] = "how are you?"; char* s = concatStr(s1, s2); cout &lt;&lt; s; delete[] s;</pre>	Hello, how are you?
<pre>char s1[] = "Nice to "; char s2[] = "meet you."; char* s = concatStr(s1, s2); cout &lt;&lt; s; delete[] s;</pre>	Nice to meet you.
<pre>char s1[] = "Nice "; char s2[] = "to meet "; char s3[] = "you."; char* temp = concatStr(s1, s2); char* s = concatStr(temp, s3); cout &lt;&lt; s; delete[] s; delete[] temp;</pre>	Nice to meet you.
	Ho Chi Minh University of Technology.

<pre>char s1[] = "Ho Chi Minh"; char s2[] = "University"; char s3[] = "of Technology."; char* temp = concatStr(s1, s2); char* s = concatStr(temp, s3); cout &lt;&lt; s; delete[] s; delete[] temp;</pre>	
<pre>char s1[] = "This question "; char s2[] = "is as easy as "; char s3[] = "the other."; char* temp = concatStr(s1, s2); char* s = concatStr(temp, s3); cout &lt;&lt; s; delete[] s; delete[] temp;</pre>	This question is as easy as the other.
<pre>char s1[] = "That's "; char s2[] = "a good idea."; char* s = concatStr(s1, s2); cout &lt;&lt; s; delete[] s;</pre>	That's a good idea.
<pre>char s1[] = "123"; char s2[] = "456"; char* s = concatStr(s1, s2); cout &lt;&lt; s; delete[] s;</pre>	123456
<pre>char s1[] = ""; char s2[] = "CSE"; char* s = concatStr(s1, s2); cout &lt;&lt; s; delete[] s;</pre>	CSE
<pre>char s1[] = ""; char s2[] = ""; char* s = concatStr(s1, s2); cout &lt;&lt; s; delete[] s;</pre>	

**Answer:** (penalty regime: 0 %)

Reset answer

```
1 char* concatStr(char* str1, char* str2) {
2     // TODO
3     int len1 = 0;
4     while (*(str1 + len1) != '\0') {
5         len1++;
6     }
7     int len2 = 0;
8     while (*(str2 + len2) != '\0') {
9         len2++;
10    }
11    char* res= new char[len1 + len2 + 1];
12    char *temp = res;
13    while(*str1 != '\0'){
```

```

14         *res= *str1;
15         ++str1;
16         ++res;
17     }
18     while(*str2 != '\0'){
19         *res = *str2;
20         ++str2;
21         ++res;
22     }
23     *res = '\0';
24     return temp;
25 }

```

Passed all tests! ✓

#### Câu hỏi 5

Đúng

Đạt điểm 1,00

#### Mô tả tiếng Việt:

Chuyển vị của một ma trận 2 chiều là một phần quan trọng trong việc tính toán trên ma trận nói riêng và đại số tuyến tính nói chung.

Gọi B là ma trận sau khi chuyển vị của ma trận A thì ma trận B có tính chất là  $b[i][j] = a[j][i]$ .

Hãy viết hàm **int\*\* transposeMatrix(int\*\* matrix, int r, int c)** thực hiện phép chuyển vị trên ma trận đã được đề cập bên trên.

#### Đầu vào:

- Con trỏ tới mảng 2 chiều. Mỗi phần tử trong mảng 2 chiều có giá trị trong khoảng (-1000; 1000).
- Kích thước mảng 2 chiều là 1 cặp số dương r, c. Trong đó: r là số hàng của ma trận, c là số cột của ma trận. Giá trị n không vượt quá 1000.

**Đầu ra:** Con trỏ tới mảng hai chiều sau khi được chuyển vị. trong trường hợp ma trận đầu vào rỗng, trả về con trỏ null.

#### English version:

Transposition of a two-dimensional matrix is an important term for matrix calculations in particular and linear algebra in general.

A matrix B transposed from a matrix A that satisfied the following formula  $b[i][j] = a[j][i]$ .

Implement the function **int\*\* transposeMatrix(int\*\* matrix, int r, int c)** that perform the transposition of the matrix mentioned above.

Input:

- The pointer that points to a two-dimensional matrix each of whose elements is in the range (-1000; 1000).
- The size of the matrix consists of the number of row r and the number of column n.

Output: The pointer that points to transposed two-dimensional matrix. If the input matrix is empty, return the null pointer.

#### For example:

Test	Input	Result
1	2 2 1 2 3 4	1 3 2 4
2	1 1 1	1
3		1 4 7 2 5 8 3 6 9



	3 3 1 2 3 4 5 6 7 8 9	
4	4 4 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16	1 5 9 13 2 6 10 14 3 7 11 15 4 8 12 16
5	2 2 10 12 14 16	10 14 12 16
6	2 3 1 2 3 4 5 6	1 4 2 5 3 6
7	1 3 1 2 3	1 2 3
8	3 1 1 1 2	1 1 2
9	0 0	NULL
10	1 2 1 2	1 2

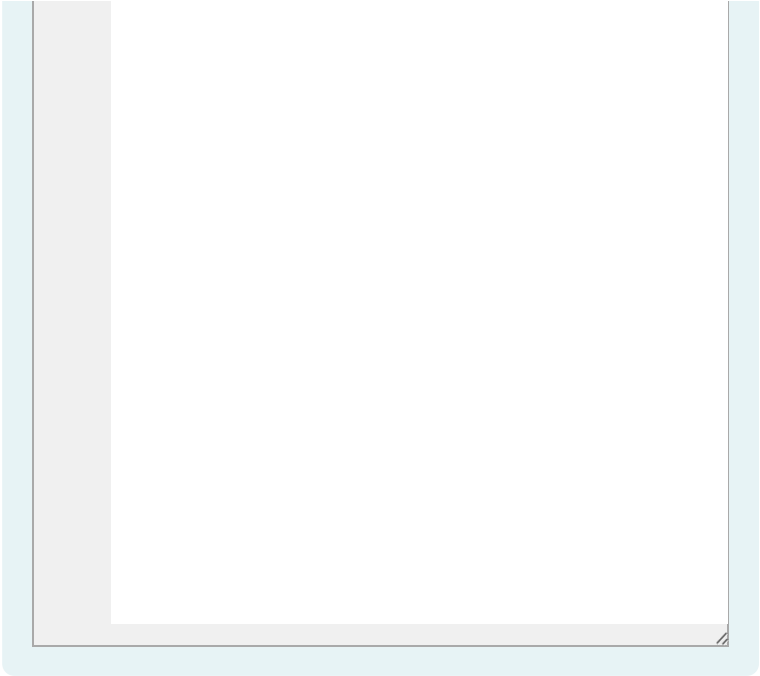
**Answer:** (penalty regime: 0 %)

Reset answer

```

1 int** transposeMatrix(int** matrix, int r, int c)
2     // TODO
3     if(matrix == nullptr || r <= 0 || c <= 0){
4         return nullptr;
5     }
6     int **tran = new int*[c];
7     for(int i = 0; i < c; ++i){
8         tran[i] = new int[r];
9     }
10
11     for(int i = 0; i < c; ++i){
12         for(int j = 0; j < r; ++j){
13             tran[i][j] = matrix[j][i];
14         }
15     }
16     return tran;
17 }

```



	Test	Input	Expected	Got	
✓	1	2 2 1 2 3 4	1 3 2 4	1 3 2 4	✓

Passed all tests! ✓