[**https://www.w3resource.com/c-programming-exercises/array/index.php**](https://www.w3resource.com/c-programming-exercises/array/index.php)

[**https://codeforwin.org/c-programming/array-programming-exercises-and**](https://codeforwin.org/c-programming/array-programming-exercises-and)

**Write a program in C to print all unique elements in an array**

**#include <stdio.h>**

***// Main function***

**int main()**

**{**

**int arr1[100], n, ctr = 0; *// Declare an array to store integer values, n for array size, and ctr for counting duplicates***

**int i, j, k; *// Declare loop counters***

***// Prompt the user to input the number of elements to be stored in the array***

**printf("\n\nPrint all unique elements of an array:\n");**

**printf("------------------------------------------\n");**

**printf("Input the number of elements to be stored in the array: ");**

**scanf("%d", &n);**

***// Prompt the user to input n elements into the array***

**printf("Input %d elements in the array :\n", n);**

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

**{**

**printf("element - %d : ", i);**

**scanf("%d", &arr1[i]); *// Read the input and store it in the array***

**}**

***// Print unique elements in the array***

**printf("\nThe unique elements found in the array are: \n");**

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

**{**

**ctr = 0; *// Reset the counter for each element***

**for (j = 0, k = n; j < k + 1; j++)**

**{**

***/\* Increment the counter when the search value is duplicate. \*/***

**if (i != j)**

**{**

**if (arr1[i] == arr1[j])**

**{**

**ctr++;**

**}**

**}**

**}**

**if (ctr == 0)**

**{**

**printf("%d ", arr1[i]); *// Print the unique element***

**}**

**}**

**printf("\n\n");**

**return 0; *// Return 0 to indicate successful execution***

**}**

**counts the total number of duplicate elements in an array, and provides their indices and values**

**#include <stdio.h>**

**void findDuplicates(int arr[], int size) {**

**int i, j, count = 0;**

**printf("Duplicate elements and their indices:\n");**

***// To keep track of visited elements***

**int visited[size];**

**for (i = 0; i < size; i++) {**

**visited[i] = 0; *// Initialize visited array with 0***

**}**

**for (i = 0; i < size; i++) {**

**if (visited[i] == 1) {**

**continue; *// Skip already visited elements***

**}**

**int duplicate = 0; *// Flag to check for duplicates***

**for (j = i + 1; j < size; j++) {**

**if (arr[i] == arr[j]) {**

**if (duplicate == 0) {**

**printf("Value: %d, Index: %d\n", arr[i], i);**

**duplicate = 1;**

**count++;**

**}**

**printf("Value: %d, Index: %d\n", arr[i], j);**

**visited[j] = 1; *// Mark element as visited***

**}**

**}**

**}**

**printf("Total number of duplicate elements: %d\n", count);**

**}**

**int main() {**

**int arr[] = {1, 2, 3, 2, 4, 3, 5, 6, 3};**

**int size = sizeof(arr) / sizeof(arr[0]);**

**printf("Original array: ");**

**for (int i = 0; i < size; i++) {**

**printf("%d ", arr[i]);**

**}**

**printf("\n");**

**findDuplicates(arr, size);**

**return 0;**

**}**

**To remove an element from the middle of an array, shift all elements after the removed element one position to the left**

#include <iostream>

**void** removeElement**(int** arr**[],** **int&** size**,** **int** index**)** **{**

***// Check if the index is within the bounds of the array***

**if** **(**index **<** 0 **||** index **>=** size**)** **{**

std**::**cout **<<** "Index out of bounds" **<<** std**::**endl**;**

**return;**

**}**

***// Shift elements to the left***

**for** **(int** i **=** index**;** i **<** size **-** 1**;** i**++)** **{**

arr**[**i**]** **=** arr**[**i **+** 1**];**

**}**

***// Decrease the size of the array***

size**--;**

**}**

**int** main**()** **{**

***// Define an array***

**int** arr**[]** **=** **{**1**,** 2**,** 3**,** 4**,** 5**};**

**int** size **=** **sizeof(**arr**)** **/** **sizeof(**arr**[**0**]);**

***// Index of the element to be removed***

**int** indexToRemove **=** 2**;** ***// For example, removing the element at index 2 (which is '3')***

std**::**cout **<<** "Original array: "**;**

**for** **(int** i **=** 0**;** i **<** size**;** i**++)** **{**

std**::**cout **<<** arr**[**i**]** **<<** " "**;**

**}**

std**::**cout **<<** std**::**endl**;**

***// Remove the element at the specified index***

removeElement**(**arr**,** size**,** indexToRemove**);**

std**::**cout **<<** "Array after removal: "**;**

**for** **(int** i **=** 0**;** i **<** size**;** i**++)** **{**

std**::**cout **<<** arr**[**i**]** **<<** " "**;**

**}**

std**::**cout **<<** std**::**endl**;**

**return** 0**;**

**}**

**To print an array in reverse order**

#include <iostream>

**int** main**()** **{**

***// Define an array***

**int** arr**[]** **=** **{**1**,** 2**,** 3**,** 4**,** 5**};**

***// Calculate the size of the array***

**int** size **=** **sizeof(**arr**)** **/** **sizeof(**arr**[**0**]);**

***// Print the array in reverse order***

std**::**cout **<<** "Array in reverse order: "**;**

**for** **(int** i **=** size **-** 1**;** i **>=** 0**;** i**--)** **{**

std**::**cout **<<** arr**[**i**]** **<<** " "**;**

**}**

std**::**cout **<<** std**::**endl**;**

**return** 0**;**

**}**

**Separate odd and even from a 3D array**

#include <iostream>

**int** main**()** **{**

***// Define a 3D array of size 4x3x3***

**const** **int** x **=** 4**,** y **=** 3**,** z **=** 3**;**

**int** arr**[**x**][**y**][**z**]** **=** **{**

**{{**1**,** 2**,** 3**},** **{**4**,** 5**,** 6**},** **{**7**,** 8**,** 9**}},**

**{{**10**,** 11**,** 12**},** **{**13**,** 14**,** 15**},** **{**16**,** 17**,** 18**}},**

**{{**19**,** 20**,** 21**},** **{**22**,** 23**,** 24**},** **{**25**,** 26**,** 27**}},**

**{{**28**,** 29**,** 30**},** **{**31**,** 32**,** 33**},** **{**34**,** 35**,** 36**}}**

**};**

***// Calculate the total number of elements***

**const** **int** totalElements **=** x **\*** y **\*** z**;**

***// Define arrays to store odd and even numbers***

**int** oddNumbers**[**totalElements**];**

**int** evenNumbers**[**totalElements**];**

**int** oddIndex **=** 0**;**

**int** evenIndex **=** 0**;**

***// Iterate through the 3D array***

**for** **(int** i **=** 0**;** i **<** x**;** **++**i**)** **{**

**for** **(int** j **=** 0**;** j **<** y**;** **++**j**)** **{**

**for** **(int** k **=** 0**;** k **<** z**;** **++**k**)** **{**

**if** **(**arr**[**i**][**j**][**k**]** **%** 2 **==** 0**)** **{**

evenNumbers**[**evenIndex**++]** **=** arr**[**i**][**j**][**k**];**

**}** **else** **{**

oddNumbers**[**oddIndex**++]** **=** arr**[**i**][**j**][**k**];**

**}**

**}**

**}**

**}**

***// Print odd numbers***

std**::**cout **<<** "Odd Numbers: "**;**

**for** **(int** i **=** 0**;** i **<** oddIndex**;** **++**i**)** **{**

std**::**cout **<<** oddNumbers**[**i**]** **<<** " "**;**

**}**

std**::**cout **<<** std**::**endl**;**

diagonal

***// Print even numbers***

std**::**cout **<<** "Even Numbers: "**;**

**for** **(int** i **=** 0**;** i **<** evenIndex**;** **++**i**)** **{**

std**::**cout **<<** evenNumbers**[**i**]** **<<** " "**;**

**}**

std**::**cout **<<** std**::**endl**;**

**return** 0**;**

**}**

**compare two 3D arrays based on user input and check if they are identical, we need to take user input for the dimensions of the 3D arrays and then take the elements for both arrays. We will then compare the elements of the two arrays. If all elements are the same, the arrays are identical; otherwise, they are not.**

#include <iostream>

**using** **namespace** std**;**

**bool** areIdentical**(int\*\*\*** arr1**,** **int\*\*\*** arr2**,** **int** x**,** **int** y**,** **int** z**)** **{**

**for** **(int** i **=** 0**;** i **<** x**;** i**++)** **{**

**for** **(int** j **=** 0**;** j **<** y**;** j**++)** **{**

**for** **(int** k **=** 0**;** k **<** z**;** k**++)** **{**

**if** **(**arr1**[**i**][**j**][**k**]** **!=** arr2**[**i**][**j**][**k**])** **{**

**return** **false;**

**}**

**}**

**}**

**}**

**return** **true;**

**}**

**int\*\*\*** create3DArray**(int** x**,** **int** y**,** **int** z**)** **{**

**int\*\*\*** arr **=** **new** **int\*\*[**x**];**

**for** **(int** i **=** 0**;** i **<** x**;** i**++)** **{**

arr**[**i**]** **=** **new** **int\*[**y**];**

**for** **(int** j **=** 0**;** j **<** y**;** j**++)** **{**

arr**[**i**][**j**]** **=** **new** **int[**z**];**

**}**

**}**

**return** arr**;**

**}**

**void** input3DArray**(int\*\*\*** arr**,** **int** x**,** **int** y**,** **int** z**)** **{**

**for** **(int** i **=** 0**;** i **<** x**;** i**++)** **{**

**for** **(int** j **=** 0**;** j **<** y**;** j**++)** **{**

**for** **(int** k **=** 0**;** k **<** z**;** k**++)** **{**

cin **>>** arr**[**i**][**j**][**k**];**

**}**

**}**

**}**

**}**

**void** delete3DArray**(int\*\*\*** arr**,** **int** x**,** **int** y**)** **{**

**for** **(int** i **=** 0**;** i **<** x**;** i**++)** **{**

**for** **(int** j **=** 0**;** j **<** y**;** j**++)** **{**

**delete[]** arr**[**i**][**j**];**

**}**

**delete[]** arr**[**i**];**

**}**

**delete[]** arr**;**

**}**

**int** main**()** **{**

**int** x**,** y**,** z**;**

cout **<<** "Enter the dimensions of the 3D arrays (x y z): "**;**

cin **>>** x **>>** y **>>** z**;**

***// Create and input the first 3D array***

cout **<<** "Enter the elements of the first 3D array:" **<<** endl**;**

**int\*\*\*** arr1 **=** create3DArray**(**x**,** y**,** z**);**

input3DArray**(**arr1**,** x**,** y**,** z**);**

***// Create and input the second 3D array***

cout **<<** "Enter the elements of the second 3D array:" **<<** endl**;**

**int\*\*\*** arr2 **=** create3DArray**(**x**,** y**,** z**);**

input3DArray**(**arr2**,** x**,** y**,** z**);**

***// Check if the two 3D arrays are identical***

**if** **(**areIdentical**(**arr1**,** arr2**,** x**,** y**,** z**))** **{**

cout **<<** "The two 3D arrays are identical." **<<** endl**;**

**}** **else** **{**

cout **<<** "The two 3D arrays are not identical." **<<** endl**;**

**}**

***// Deallocate dynamic memory***

delete3DArray**(**arr1**,** x**,** y**);**

delete3DArray**(**arr2**,** x**,** y**);**

**return** 0**;**

**}**

**finding the average of even numbers among the non-diagonal elements of a 2D array in C++, we will perform the following steps:**

**Iterate through each element of the 2D array.**

**Check if the element is a non-diagonal element.**

**Check if the non-diagonal element is an even number.**

**Calculate the average of these even numbers.**

#include <iostream>

**using** **namespace** std**;**

**double** findEvenNonDiagonalAverage**(int\*\*** arr**,** **int** rows**,** **int** cols**)** **{**

**int** sum **=** 0**;**

**int** count **=** 0**;**

**for** **(int** i **=** 0**;** i **<** rows**;** i**++)** **{**

**for** **(int** j **=** 0**;** j **<** cols**;** j**++)** **{**

**if** **(**i **!=** j **&&** i **+** j **!=** cols **-** 1**)** **{** ***// Non-diagonal elements***

**if** **(**arr**[**i**][**j**]** **%** 2 **==** 0**)** **{** ***// Check if even***

sum **+=** arr**[**i**][**j**];**

count**++;**

**}**

**}**

**}**

**}**

**return** **(**count **==** 0**)** **?** 0 **:** **(double)**sum **/** count**;**

**}**

**int** main**()** **{**

**int** rows**,** cols**;**

cout **<<** "Enter the number of rows: "**;**

cin **>>** rows**;**

cout **<<** "Enter the number of columns: "**;**

cin **>>** cols**;**

***// Dynamic memory allocation for 2D array***

**int\*\*** arr **=** **new** **int\*[**rows**];**

**for** **(int** i **=** 0**;** i **<** rows**;** i**++)** **{**

arr**[**i**]** **=** **new** **int[**cols**];**

**}**

cout **<<** "Enter the elements of the array:" **<<** endl**;**

**for** **(int** i **=** 0**;** i **<** rows**;** i**++)** **{**

**for** **(int** j **=** 0**;** j **<** cols**;** j**++)** **{**

cin **>>** arr**[**i**][**j**];**

**}**

**}**

**double** average **=** findEvenNonDiagonalAverage**(**arr**,** rows**,** cols**);**

cout **<<** "Average of even non-diagonal elements: " **<<** average **<<** endl**;**

***// Deallocate dynamic memory***

**for** **(int** i **=** 0**;** i **<** rows**;** i**++)** **{**

**delete[]** arr**[**i**];**

**}**

**delete[]** arr**;**

**return** 0**;**

**}**

### **Program to reverse columns in given 2D Array (Matrix)**

Given a [2D array](https://www.geeksforgeeks.org/multidimensional-arrays-c-cpp/) arr[][]of integers of size M x N, where N is the number of columns and M is the number of rows in the array. The task is to reverse every column of the given 2D array

Input: arr[][] = {{3, 2, 1}  
 {4, 5, 6},  
 {9, 8, 7}}

Output: 9 8 7  
 4 5 6  
 3 2 1

// Java implementation of the

// above approach

**import** java**.**util**.\*;**

class GFG **{**

static int M **=** 3**;**

static int N **=** 3**;**

// A utility function

// for swapping two elements.

private static int**[][]** swap**(**int**[][]** arr**,** int start**,**

int i**,** int end**,** int j**)**

**{**

int temp **=** arr**[**start**][**i**];**

arr**[**start**][**i**]** **=** arr**[**end**][**j**];**

arr**[**end**][**j**]** **=** temp**;**

**return** arr**;**

**}**

// Print the arr[][]

static void printMatrix**(**int arr**[][])**

**{**

**for** **(**int i **=** 0**;** i **<** M**;** i**++)** **{**

**for** **(**int j **=** 0**;** j **<** N**;** j**++)** **{**

System**.**out**.**print**(**arr**[**i**][**j**]** **+** " "**);**

**}**

System**.**out**.**println**();**

**}**

**}**

// Function to reverse

// the given 2D arr[][]

static void reverseColumnArray**(**int arr**[][])**

**{**

// Print the arr[][] before

// reversing every column

printMatrix**(**arr**);**

System**.**out**.**println**();**

// Traverse each column of arr[][]

**for** **(**int i **=** 0**;** i **<** N**;** i**++)** **{**

// Initialise start and end index

int start **=** 0**;**

int end **=** M **-** 1**;**

// Till start < end, swap the

// element at start and end index

**while** **(**start **<** end**)** **{**

// Swap the element

arr **=** swap**(**arr**,** start**,** i**,** end**,** i**);**

// Increment start and decrement

// end for next pair of swapping

start**++;**

end**--;**

**}**

**}**

// Print the arr[][] after

// reversing every column

printMatrix**(**arr**);**

**}**

// Driver Code

public static void main**(**String**[]** args**)**

**{**

int arr**[][]**

**=** **{** **{** 3**,** 2**,** 1 **},** **{** 4**,** 5**,** 6 **},** **{** 9**,** 8**,** 7 **}** **};**

// Function call

reverseColumnArray**(**arr**);**

**}**

**}**

// This code is contributed by 29AjayKumar

## ### 1. Find the Maximum and Minimum Values of Each Row in a 2D Array

void findRowMaxMin**(**const vector**<**vector**<**int**>>&** matrix**)** **{**

**for** **(**int i **=** 0**;** i **<** matrix**.**size**();** **++**i**)** **{**

int maxVal **=** INT\_MIN**;**

int minVal **=** INT\_MAX**;**

**for** **(**int j **=** 0**;** j **<** matrix**[**i**].**size**();** **++**j**)** **{**

**if** **(**matrix**[**i**][**j**]** **>** maxVal**)** maxVal **=** matrix**[**i**][**j**];**

**if** **(**matrix**[**i**][**j**]** **<** minVal**)** minVal **=** matrix**[**i**][**j**];**

**}**

cout **<<** "Row " **<<** i **+** 1 **<<** " - Max: " **<<** maxVal **<<** ", Min: " **<<** minVal **<<** endl**;**

**}**

**}**

### 2. Count the Number of Prime Numbers in a 1D Array

bool isPrime**(**int n**)** **{**

**if** **(**n **<=** 1**)** **return** **false;**

**for** **(**int i **=** 2**;** i **<=** sqrt**(**n**);** **++**i**)** **{**

**if** **(**n **%** i **==** 0**)** **return** **false;**

**}**

**return** **true;**

**}**

int countPrimes**(**const vector**<**int**>&** arr**)** **{**

int count **=** 0**;**

**for** **(**int num **:** arr**)** **{**

**if** **(**isPrime**(**num**))** count**++;**

**}**

**return** count**;**

**}**

### 3. Transpose a Matrix (2D Array)

vector**<**vector**<**int**>>** transpose**(**const vector**<**vector**<**int**>>&** matrix**)** **{**

int rows **=** matrix**.**size**();**

int cols **=** matrix**[**0**].**size**();**

vector**<**vector**<**int**>>** transposed**(**cols**,** vector**<**int**>(**rows**));**

**for** **(**int i **=** 0**;** i **<** rows**;** **++**i**)** **{**

**for** **(**int j **=** 0**;** j **<** cols**;** **++**j**)** **{**

transposed**[**j**][**i**]** **=** matrix**[**i**][**j**];**

**}**

**}**

**return** transposed**;**

**}**

### 4. Sum of Elements Above and Below the Main Diagonal of a Square Matrix

void sumAboveBelowDiagonal**(**const vector**<**vector**<**int**>>&** matrix**)** **{**

int aboveSum **=** 0**,** belowSum **=** 0**;**

**for** **(**int i **=** 0**;** i **<** matrix**.**size**();** **++**i**)** **{**

**for** **(**int j **=** 0**;** j **<** matrix**[**i**].**size**();** **++**j**)** **{**

**if** **(**i **<** j**)** aboveSum **+=** matrix**[**i**][**j**];**

**if** **(**i **>** j**)** belowSum **+=** matrix**[**i**][**j**];**

**}**

**}**

cout **<<** "Sum above diagonal: " **<<** aboveSum **<<** endl**;**

cout **<<** "Sum below diagonal: " **<<** belowSum **<<** endl**;**

**}**

### 6. Rotate a Matrix 90 Degrees Clockwise

void rotateMatrix**(**vector**<**vector**<**int**>>&** matrix**)** **{**

int n **=** matrix**.**size**();**

**for** **(**int i **=** 0**;** i **<** n **/** 2**;** **++**i**)** **{**

**for** **(**int j **=** i**;** j **<** n **-** i **-** 1**;** **++**j**)** **{**

int temp **=** matrix**[**i**][**j**];**

matrix**[**i**][**j**]** **=** matrix**[**n **-** j **-** 1**][**i**];**

matrix**[**n **-** j **-** 1**][**i**]** **=** matrix**[**n **-** i **-** 1**][**n **-** j **-** 1**];**

matrix**[**n **-** i **-** 1**][**n **-** j **-** 1**]** **=** matrix**[**j**][**n **-** i **-** 1**];**

matrix**[**j**][**n **-** i **-** 1**]** **=** temp**;**

**}**

**}**

**}**

### 7. Frequency of Each Element in a 1D Array

### 8. Sum of Border Elements in a 2D Array

int sumBorderElements**(**const vector**<**vector**<**int**>>&** matrix**)** **{**

int sum **=** 0**;**

int rows **=** matrix**.**size**();**

int cols **=** matrix**[**0**].**size**();**

**for** **(**int i **=** 0**;** i **<** rows**;** **++**i**)** **{**

**for** **(**int j **=** 0**;** j **<** cols**;** **++**j**)** **{**

**if** **(**i **==** 0 **||** i **==** rows **-** 1 **||** j **==** 0 **||** j **==** cols **-** 1**)** **{**

sum **+=** matrix**[**i**][**j**];**

**}**

**}**

**}**

**return** sum**;**

**}**

### 9. Row or Column with the Highest Sum in a 2D Array

int rowWithHighestSum**(**const vector**<**vector**<**int**>>&** matrix**)** **{**

int maxSum **=** 0**,** maxRow **=** 0**;**

**for** **(**int i **=** 0**;** i **<** matrix**.**size**();** **++**i**)** **{**

int rowSum **=** 0**;**

**for** **(**int j **=** 0**;** j **<** matrix**[**i**].**size**();** **++**j**)** **{**

rowSum **+=** matrix**[**i**][**j**];**

**}**

**if** **(**rowSum **>** maxSum**)** **{**

maxSum **=** rowSum**;**

maxRow **=** i**;**

**}**

**}**

**return** maxRow**;**

**}**

### 10. Check if a 1D Array is a Palindrome