

## Simple array declaration

```
string cars[4] = {"Volvo", "BMW", "Ford", "Mazda"};
cout << cars[0];
```

## Change an Array Element

```
string cars[4] = {"Volvo", "BMW", "Ford", "Mazda"};
cars[0] = "Opel";
cout << cars[0];
```

## Loop Through an Array

You can loop through the array elements with the `for` loop.

The following example outputs all elements in the **cars** array:

### Example

```
string cars[5] = {"Volvo", "BMW", "Ford", "Mazda", "Tesla"};
for (int i = 0; i < 5; i++) {
    cout << cars[i] << "\n";
}
```

## Omit Array Size

In C++, you don't have to specify the size of the array. The compiler is smart enough to determine the size of the array based on the number of inserted values:

```
string cars[] = {"Volvo", "BMW", "Ford"}; // Three array elements
```

The example above is equal to:

```
string cars[3] = {"Volvo", "BMW", "Ford"};
```

# Get the Size of an Array

To get the size of an array, you can use the `sizeof()` operator:

## Example

```
int myNumbers[5] = {10, 20, 30, 40, 50};  
cout << sizeof(myNumbers);
```

## Loop Through an Array with sizeof()

In the [Arrays and Loops Chapter](#), we wrote the size of the array in the loop condition (`i < 5`). This is not ideal, since it will only work for arrays of a specified size.

However, by using the `sizeof()` approach from the example above, we can now make loops that work for arrays of any size, which is more sustainable.

Instead of writing:

```
int myNumbers[5] = {10, 20, 30, 40, 50};  
for (int i = 0; i < 5; i++) {  
    cout << myNumbers[i] << "\n";  
}
```

It is better to write:

## Example

```
int myNumbers[5] = {10, 20, 30, 40, 50};  
for (int i = 0; i < sizeof(myNumbers) / sizeof(int); i++) {  
    cout << myNumbers[i] << "\n";  
}
```

Note that, in C++ version 11 (2011), you can also use the ["for-each" loop](#):

## Example

```
int myNumbers[5] = {10, 20, 30, 40, 50};
for (int i : myNumbers) {
    cout << i << "\n";
}
```

## Multi-Dimensional Arrays

A multi-dimensional array is an array of arrays.

To declare a multi-dimensional array, define the variable type, specify the name of the array followed by square brackets which specify how many elements the main array has, followed by another set of square brackets which indicates how many elements the sub-arrays have:

```
string letters[2][4];
```

As with ordinary arrays, you can insert values with an array literal - a comma-separated list inside curly braces. In a multi-dimensional array, each element in an array literal is another array literal.

```
string letters[2][4] = {
    { "A", "B", "C", "D" },
    { "E", "F", "G", "H" }
};
```

Each set of square brackets in an array declaration adds another **dimension** to an array. An array like the one above is said to have two dimensions.

Arrays can have any number of dimensions. The more dimensions an array has, the more complex the code becomes. The following array has three dimensions:

```
string letters[2][2][2] = {
    {
        { "A", "B" },
        { "C", "D" }
    },
    {
        { "E", "F" },
        { "G", "H" }
    }
};
```

```
    { "G", "H" }  
  }  
};
```

## Access the Elements of a Multi-Dimensional Array

To access an element of a multi-dimensional array, specify an index number in each of the array's dimensions.

This statement accesses the value of the element in the **first row (0)** and **third column (2)** of the **letters** array.

### Example

```
string letters[2][4] = {  
    { "A", "B", "C", "D" },  
    { "E", "F", "G", "H" }  
};  
  
cout << letters[0][2]; // Outputs "C"
```

## Change Elements in a Multi-Dimensional Array

To change the value of an element, refer to the index number of the element in each of the dimensions:

### Example

```
string letters[2][4] = {  
    { "A", "B", "C", "D" },  
    { "E", "F", "G", "H" }  
};  
letters[0][0] = "Z";
```

```
cout << letters[0][0]; // Now outputs "Z" instead of "A"
```

## Loop Through a Multi-Dimensional Array

To loop through a multi-dimensional array, you need one loop for each of the array's dimensions.

The following example outputs all elements in the **letters** array:

### Example

```
string letters[2][4] = {
    { "A", "B", "C", "D" },
    { "E", "F", "G", "H" }
};

for (int i = 0; i < 2; i++) {
    for (int j = 0; j < 4; j++) {
        cout << letters[i][j] << "\n";
    }
}
```

This example shows how to loop through a three-dimensional array:

### Example

```
string letters[2][2][2] = {
    {
        { "A", "B" },
        { "C", "D" }
    },
    {
        { "E", "F" },
        { "G", "H" }
    }
};
```

```

for (int i = 0; i < 2; i++) {
    for (int j = 0; j < 2; j++) {
        for (int k = 0; k < 2; k++) {
            cout << letters[i][j][k] << "\n";
        }
    }
}

```

## C++ program for linear search in array

```

#include <iostream>
using namespace std;

int main(){
    int input[100], count, i, num;

    cout << "Enter Number of Elements in Array\n";
    cin >> count;

    cout << "Enter " << count << " numbers \n";

    // Read array elements
    for(i = 0; i < count; i++){
        cin >> input[i];
    }

    cout << "Enter a number to search in Array\n";
    cin >> num;

    // search num in inputArray from index 0 to elementCount-1
    for(i = 0; i < count; i++){
        if(input[i] == num){
            cout << "Element found at index " << i;
            break;
        }
    }

    if(i == count){
        cout << "Element Not Present in Input Array\n";
    }

    return 0;
}

```

## Output

```
Enter Number of Elements in Array
6
Enter 6 numbers
8 4 7 1 3 9
Enter a number to serach in Array
3
Element found at index 4
```

## Add an Element to the End of an Array

This program prompts the user to enter 5 numbers or elements for an array, followed by the element to insert at the end of the given array.

The question is: write a program in C++ to insert an element at the end of an array. Here is its answer:

```
#include<iostream>
using namespace std;
int main()
{
    int arr[6], i, elem;
    cout<<"Enter 5 Array Elements: ";
    for(i=0; i<5; i++)
        cin>>arr[i];
    cout<<"\nEnter Element to Insert: ";
    cin>>elem;
    arr[i] = elem;
    cout<<"\nThe New Array is:\n";
    for(i=0; i<6; i++)
        cout<<arr[i]<<" ";
    cout<<endl;
    return 0;
}
```

## Insert an Array Element at a Specific Position

To insert an element in an array in C++ programming, you have to ask the user to enter the size and elements of the array. And then ask to enter the element to insert and at what position, as shown in the program given below:

After inserting the element at the desired position, don't forget to display the new array on the screen.

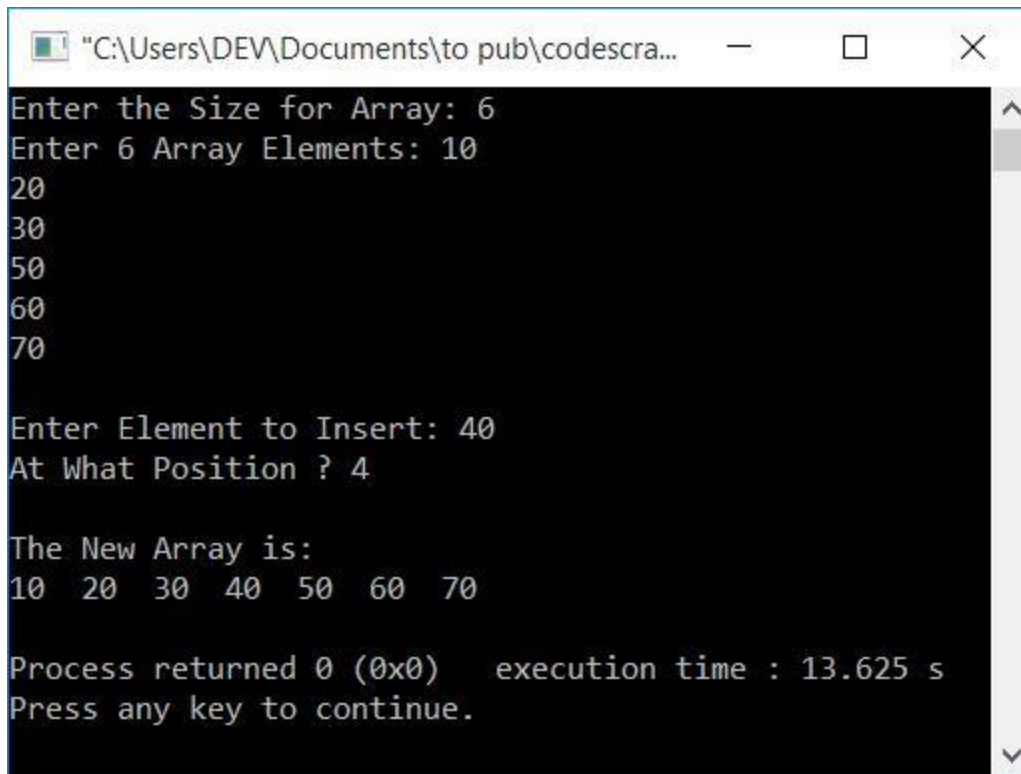
```
#include<iostream>
using namespace std;
int main()
{
    int arr[50], i, elem, pos, tot;
    cout<<"Enter the Size for Array: ";
    cin>>tot;
    cout<<"Enter "<<tot<<" Array Elements: ";
    for(i=0; i<tot; i++)
        cin>>arr[i];
    cout<<"\nEnter Element to Insert: ";
    cin>>elem;
    cout<<"At What Position ? ";
    cin>>pos;
    for(i=tot; i>=pos; i--)
        arr[i] = arr[i-1];
    arr[i] = elem;
    tot++;
    cout<<"\nThe New Array is:\n";
    for(i=0; i<tot; i++)
        cout<<arr[i]<<" ";
    cout<<endl;
    return 0;
}
```

Here is its sample run with the following user input:

- 6 as the array size
- 10, 20, 30, 50, 60, and 70 as 6 array elements
- 40 as the element to insert
- 4 as the position to insert the element.

After supplying these inputs, here is the snapshot that shows all the operations that have been done on the output screen:



A screenshot of a Windows command prompt window titled "C:\Users\DEV\Documents\to pub\codescra...". The window has standard Windows window controls (minimize, maximize, close). The text inside the window shows the execution of a C++ program. It starts with "Enter the Size for Array: 6", followed by "Enter 6 Array Elements: 10". Then, six numbers are entered on separate lines: 20, 30, 50, 60, 70. Next, it asks "Enter Element to Insert: 40" and "At What Position ? 4". It then displays "The New Array is:" followed by the array elements "10 20 30 40 50 60 70". At the bottom, it shows "Process returned 0 (0x0) execution time : 13.625 s" and "Press any key to continue.".

```
"C:\Users\DEV\Documents\to pub\codescra...  
Enter the Size for Array: 6  
Enter 6 Array Elements: 10  
20  
30  
50  
60  
70  
  
Enter Element to Insert: 40  
At What Position ? 4  
  
The New Array is:  
10 20 30 40 50 60 70  
  
Process returned 0 (0x0) execution time : 13.625 s  
Press any key to continue.
```

## In C++, delete an element from an array.

To delete an element from an array in C++ programming, you have to ask the user to enter the array's 10 elements first. And then ask for the element that has to be deleted. Now search for that number or element and delete it if found. Otherwise, print a message such as "Element not found."

The question is, "Write a program in C++ that deletes an element from an array." Here is the answer to this question:

```
#include<iostream>  
using namespace std;  
int main()  
{  
    int arr[10], tot=10, i, elem, j, found=0;  
    cout<<"Enter 10 Array Elements: ";  
    for(i=0; i<tot; i++)  
        cin>>arr[i];  
    cout<<"\nEnter Element to Delete: ";  
    cin>>elem;  
    for(i=0; i<tot; i++)  
    {
```

```

        if(arr[i]==elem)
        {
            for(j=i; j<(tot-1); j++)
                arr[j] = arr[j+1];
            found++;
            i--;
            tot--;
        }
    }
    if(found==0)
        cout<<"\nElement doesn't found in the Array!";
    else
        cout<<"\nElement Deleted Successfully!";
    cout<<endl;
    return 0;
}

```

## Delete array element in given index range [L – R] in C++ Program

```

#include <bits/stdc++.h>
using namespace std;
int deleteElementsInRange(int arr[], int n, int l, int r) {
    int i, newIndex = 0;
    for (i = 0; i < n; i++) {
        // adding updating element if it is not in the given range
        if (i < l-1 || i > r-1) {
            arr[newIndex] = arr[i];
            newIndex++;
        }
    }
    // returning the updated index
    return newIndex;
}
int main() {
    int n = 9, l = 1, r = 6;
    int arr[] = { 1, 2, 3, 4, 5, 6, 7, 8, 9 };
    int updatedArrayLength = deleteElementsInRange(arr, n, l, r);
}

```

```
for (int i = 0; i < updatedArrayLength; i++) {  
    cout << arr[i] << " ";  
}  
cout << endl;  
return 0;  
}
```

aarray

Reverse an array

Prefix sum

## C++ Program to get the subarray from an array using a specified range of indices

```
#include <iostream>
```

```
# define Z 50
```

```
using namespace std;
```

```
void displayArr(int arr[], int n){
```

```
    for( int i = 0; i < n; i++) {
```

```
        cout << arr[ i ] << " , ";
```

```
    }
```

```
    cout << endl;
```

```
}
```

```
void pickSubarray( int A[], int n, int sub[], int &m, int i, int j ) {
```

```
    int ind = 0;
```

```

for( int k = i; k < j; k++, ind++ ) {
    sub[ ind ] = A[ k ];
    m += 1;
}
}

int sub[ Z ];
int m = 0;

cout << "Given Array: ";
displayArr( A, n );

cout << "Sub array from index 3 to 9: ";
pickSubarray( A, n, sub, m, 3, 9);
displayArr( sub, m );
}

```

## Maximum Sum Subsequence

```

// C++ program to implement
// the above approach
#include <bits/stdc++.h>
using namespace std;

// Function to print the maximum
// non-empty subsequence sum
int MaxNonEmpSubSeq(int a[], int n)
{
    // Stores the maximum non-empty

```

```

// subsequence sum in an array
int sum = 0;


// Traverse the array
for (int i = 0; i < n; i++) {

    // If a[i] is greater than 0
    if (a[i] > 0) {

        // Update sum
        sum += a[i];
    }
}
return sum;
}


// Driver Code
int main()
{
    int arr[] = { -2, 11, -4, 2, -3, -10 };
    int N = sizeof(arr) / sizeof(arr[0]);

    cout << MaxNonEmpSubSeq(arr, N);

    return 0;
}

```

## Counting frequencies of array elements

```
// CPP program to count frequencies of array items
#include <bits/stdc++.h>
using namespace std;

void countFreq(int arr[], int n)
{
    // Mark all array elements as not visited
    vector<bool> visited(n, false);

    // Traverse through array elements and
    // count frequencies
    for (int i = 0; i < n; i++) {

        // Skip this element if already processed
        if (visited[i] == true)
            continue;

        // Count frequency
        int count = 1;
        for (int j = i + 1; j < n; j++) {
            if (arr[i] == arr[j]) {
                visited[j] = true;
                count++;
            }
        }
        cout << arr[i] << " " << count << endl;
    }
}

int main()
{
    int arr[] = { 10, 20, 20, 10, 10, 20, 5, 20 };
    int n = sizeof(arr) / sizeof(arr[0]);
    countFreq(arr, n);
    return 0;
}
```

Prefix sum

```
#include <iostream>

using namespace std;

int n, q;

int v[111];

int PrefixSum[111];

int main() {

    cin >>n;

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

        cin >>v[i];

    }

    //PrefixSum[0]=0 from the declartion

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

        PrefixSum[i]=PrefixSum[i-1]+v[i];

    }

    cin >>q;

    for (int i=0;i<q;i++){

        int l, r;

        cin >>l>>r;

        cout <<PrefixSum[r]-PrefixSum[l-1]<<endl;

    }

    return 0;

}
```