

Declare Arrays in C++

Here is the general form to declare an array in C++ :

```
type array_name[array_size];
```

Above is the general form of the single or one dimensional array declaration. Here type is any valid type of C++, array_name is a valid identifier which is the name of the array given by the programmer, and the array_size represents the number of elements that this array can hold. Here is an example declaring the array named arr of type int and size 20 :

```
int arr[20];
```

Here is the general form to declare two-dimensional array in C++.

```
type array_name[array_size1][array_size2];
```

Here is the general form to declare multi-dimensional array in C++

```
type array_name[array_size1][array_size2][array_size3]...[array_sizeN];
```

Initialize Arrays in C++

Here is the general form to initialize arrays in C++:

```
type array_name[array_size] = {values_list};
```

Here is an example, initializing values to the array named arr in C++:

```
int arr[5] = {97, 69, 18, 46, 83};
```

Note - The number of values between braces { } can not be larger than the array size (here 5) or the number of elements (here 5).

If you omit the array size, then the array becomes big enough to hold the initialization is created. Therefore, if you write:

```
int arr[] = {97, 69, 18, 46, 83};
```

You will create exactly the same array as you did in the previous example. So, you can also initialize more or less than 5 values like this:

```
int arr[] = {10, 12, 23};
```

or

```
int arr[] = {12, 23, 34, 35, 45, 33, 10, 2, 54};
```

Assign Values to Specific Array Element in C++

Here is the general form to assign values to specific array elements:

```
array_name[index_number] = value;
```

Here is an example, assigning a value, 20, to the 5th element in the array named arr:

```
arr[4] = 20;
```

Note - Since, index always starts from 0, so index number 4, corresponds to the 5th element in the array.

Access Array Elements in C++

An element can be accessed by indexing the array name. This is done by placing the index number of the element within the square brackets after the name of the array. Here is the general form to access the array element in C++:

```
type variable_name = array_name[index_number];
```

Here is an example, accessing the values present at the index number 4 of the array arr:

```
int num = arr[4];
```

So, if value at arr[4] (value present at index 4, in the array arr) is 20, then 20 will initialize to the variable num.

C++ Arrays Example

Here are some example demonstrating the concept of arrays in C++ practically.

```
/* C++ Arrays - C++ Arrays Example Program */
```

```
#include<iostream.h>
#include<conio.h>
void main()
{
    clrscr();
    int arr1[5] = {1, 2, 3, 4, 5};
    int arr2[] = {10, 20, 30, 40, 50};
    int i;
    cout<<"Array arr1 contains:\n";
    for(i=0; i<5; i++)
    {
        cout<<arr1[i]<<"\t";
    }
    cout<<"\n\n";
    cout<<"Array arr2 contains:\n";
    for(i=0; i<5; i++)
    {
        cout<<arr2[i]<<"\t";
    }

    getch();
}
```

Here is the sample output of the above C++ program:



Here is an another array example.

```
/* C++ Arrays - C++ Arrays Example Program */
```

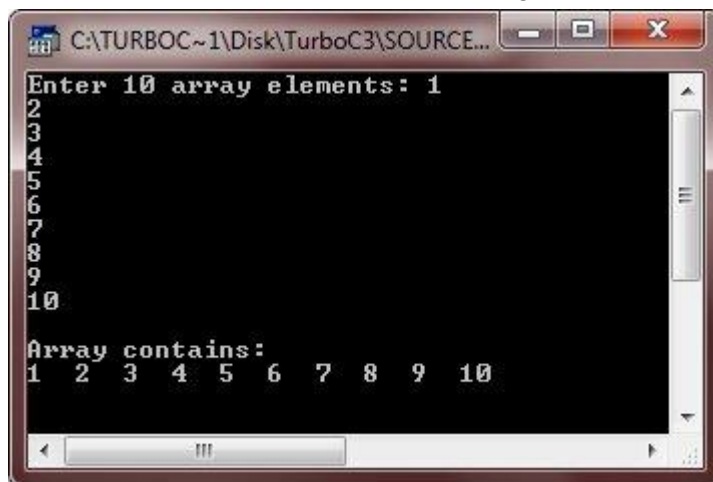
```
#include<iostream.h>
```

```

#include<conio.h>
void main()
{
    clrscr();
    int arr[10];
    int i;
    cout<<"Enter 10 array elements: ";
    for(i=0; i<10; i++)
    {
        cin>>arr[i];
    }
    cout<<"\nArray contains:\n";
    for(i=0; i<10; i++)
    {
        cout<<arr[i]<<" ";
    }
    getch();
}

```

Below is the sample run of this C++ program:



Let's make above C++ program more user-friendly.

/* C++ Arrays - C++ Arrays Example Program */

```

#include<iostream.h>
#include<conio.h>
void main()
{
    clrscr();
    int arr[100];
    int i, count=0;
    cout<<"Enter array elements(max 100) Press -1 to stop: ";
    for(i=0; i<100; i++)
    {
        cin>>arr[i];
        if(arr[i] == -1)
        {

```

```

        getch();
        break;
    }
    count++;
}
cout<<"\nArray contains:\n";
for(i=0; i<count; i++)
{
    cout<<arr[i]<<" ";
}
getch();
}

```

Here are the two sample runs of the above C++ program:

```

C:\TURBOC~1\Disk\TurboC3\SOURCE\1.EXE
Enter array elements(max 100) Press -1 to stop: 1
2
3
4
5
6
7
-1

Array contains:
1 2 3 4 5 6 7

```

```

C:\TURBOC~1\Disk\TurboC3\SOURCE\1.EXE
Enter array elements(max 100) Press -1 to stop: 1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
-1

Array contains:
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

```

In this article, you will learn and get code on one-dimensional (1D) array in C++. For example:

```
int arr[5] = {10, 20, 30, 40, 50};
```

Note - Array such type of variable that can hold upto many values of same type. Same type means, whatever the variable's (array's variable) type was declared. For example, in above array, arr can hold upto 5 integer values.

From above declaration, all the 5 integer values are stored in arr[] in following way:

- arr[0]=10
- arr[1]=20
- arr[2]=30
- arr[3]=40
- arr[4]=50

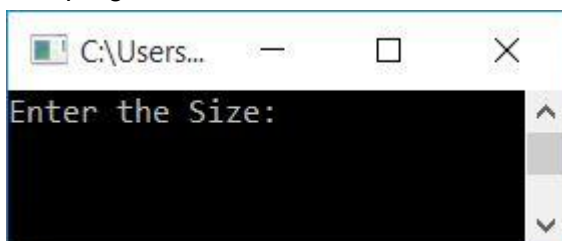
Note - Here 0, 1, 2, 3, 4 are called as array's index. Indexing in array always starts with 0.

One Dimensional Array Program in C++

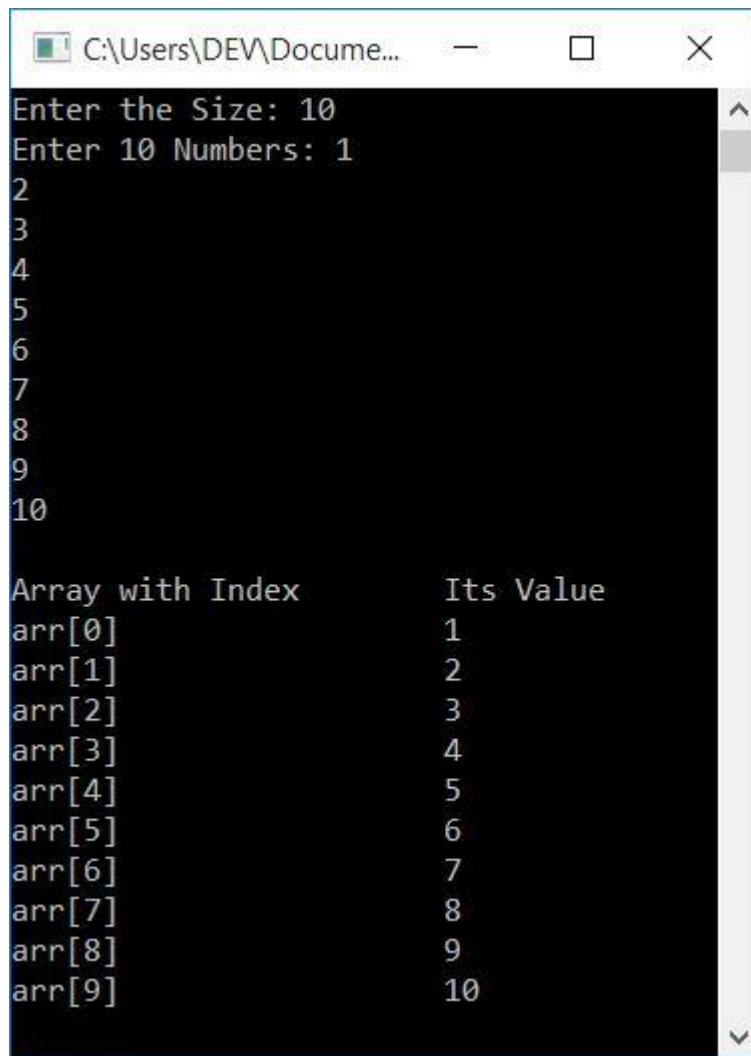
To print [one dimensional array](#) in C++ programming, you have to ask from user to enter the size and elements of [array](#). Then print it back on output with all its detail as shown in the program given below:

```
#include<iostream>
using namespace std;
int main()
{
    int arr[50], tot, i;
    cout<<"Enter the Size: ";
    cin>>tot;
    cout<<"Enter "<<tot<<" Numbers: ";
    for(i=0; i<tot; i++)
        cin>>arr[i];
    cout<<"\nArray with Index\tIts Value\n";
    for(i=0; i<tot; i++)
        cout<<"arr["<<i<<"]"<<"\t\t"<<arr[i]<<endl;
    cout<<endl;
    return 0;
}
```

This program was build and run under Code::Blocks IDE. Here is its sample run:



Now supply inputs say 10 as size and 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 as its elements or numbers to store. Here is the sample run with exactly these inputs:



```
Enter the Size: 10
Enter 10 Numbers: 1
2
3
4
5
6
7
8
9
10

Array with Index      Its Value
arr[0]                1
arr[1]                2
arr[2]                3
arr[3]                4
arr[4]                5
arr[5]                6
arr[6]                7
arr[7]                8
arr[8]                9
arr[9]                10
```

In above program, the following statement:

```
cout<<"arr["<<i<<" "<<"\t\t\t"<<arr[i]<<endl;
```

If we break the things from above statement and divide into static and dynamic output (print), here are the list:

- arr[- static output. Prints the same
- i - dynamic output. Prints whatever the value of i
- \t\t\t - static output. Prints the same, that is three tabs
- arr[i] - dynamic output. Prints whatever the value of arr[i]
- endl - static output. Used as newline. That is, next thing starts from newline

Note - Here dynamic output means, the output gets changed with change in value of i and arr[i]. Things inside "" is considered as static output

Previous array is of type int (integer). Therefore that array can hold upto 50 integer values with same variable arr[] just by changing its index. Now let's try array with its type as char (character) to store characters.

```
#include<iostream>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    char str[50];
```

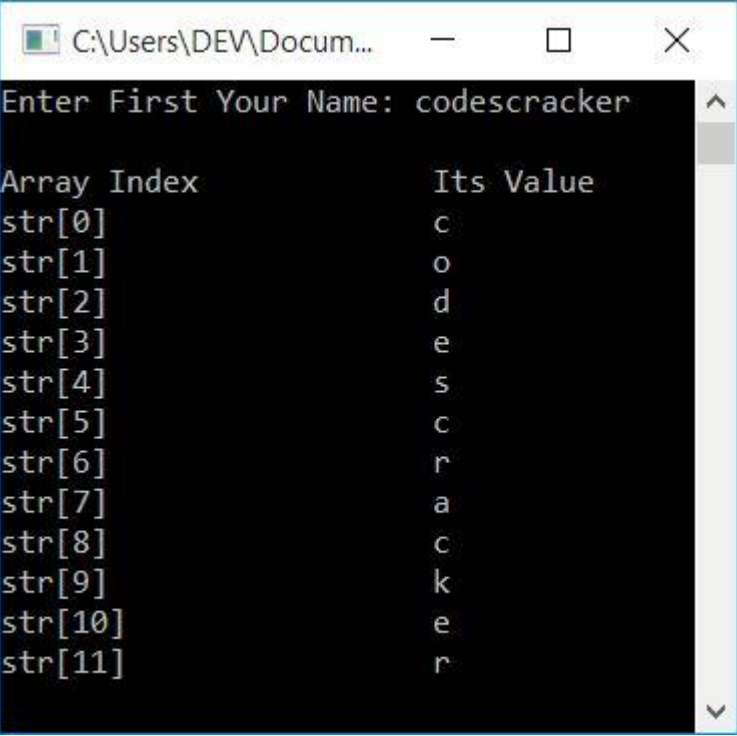
```
    int i=0;
```

```

cout<<"Enter First Your Name: ";
cin>>str;
cout<<"\nArray Index\t\tIts Value\n";
while(str[i])
{
    cout<<"str["<<i<<"]"<<"\t\t"<<str[i]<<endl;
    i++;
}
cout<<endl;
return 0;
}

```

Here is its sample run with user input, codescracker:



```

C:\Users\DEV\Docum...
Enter First Your Name: codescracker

Array Index      Its Value
str[0]           c
str[1]           o
str[2]           d
str[3]           e
str[4]           s
str[5]           c
str[6]           r
str[7]           a
str[8]           c
str[9]           k
str[10]          e
str[11]          r

```

Linear Search in C++

To search any element present inside the [array](#) in C++ programming using linear search technique, you have to ask from user to enter any 10 numbers as 10 array elements and then ask to enter a number to search as shown in the program given below.

This program doesn't allow user to define the size of an array. Later on, you will go through the program that allows user to define the size and also prints all the indexes of an element, if found multiple times.

This is the simplest program to implement linear search in C++.

```

#include<iostream>
using namespace std;
int main()
{
    int arr[10], i, num, index;
    cout<<"Enter 10 Numbers: ";

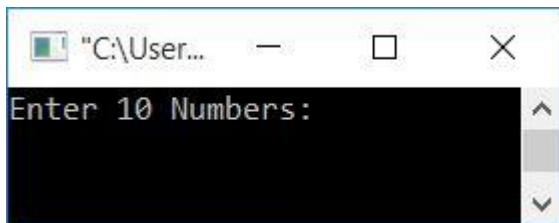
```

```

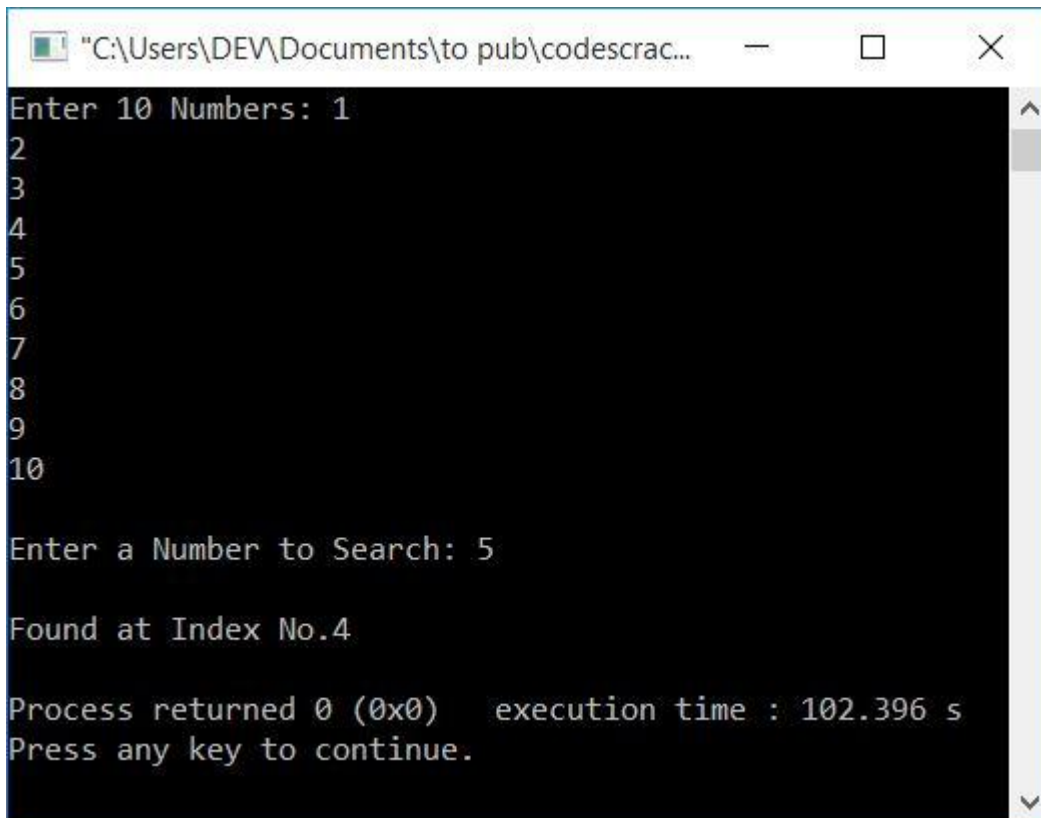
for(i=0; i<10; i++)
    cin>>arr[i];
cout<<"\nEnter a Number to Search: ";
cin>>num;
for(i=0; i<10; i++)
{
    if(arr[i]==num)
    {
        index = i;
        break;
    }
}
cout<<"\nFound at Index No."<<index;
cout<<endl;
return 0;
}

```

This program was build and run under Code::Blocks IDE. Here is its sample run:



Now supply any 10 numbers say 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, and a number say 5 to search it from the given array. Here is the sample output produced after providing these inputs:



Note - If user enters a number that doesn't exist in the array, then the program given above is not a correct one with this type of inputs. Therefore we've another program for you, as given below.

Note - If user enters a number that presents more than one time in the array, then in this case also, above program is not a correct program for you. Because that program will only print its index of first occurrence only. Rest of its index gets skipped. Therefore go with the second program, as given below.

Linear Search with Duplicate Element

This program has many extra features than previous program:

- Allows user to define the size of array
- Handles that type of inputs, when user enters a number that doesn't exist in the array
- Prints all the index numbers, if entered number found in repeated order

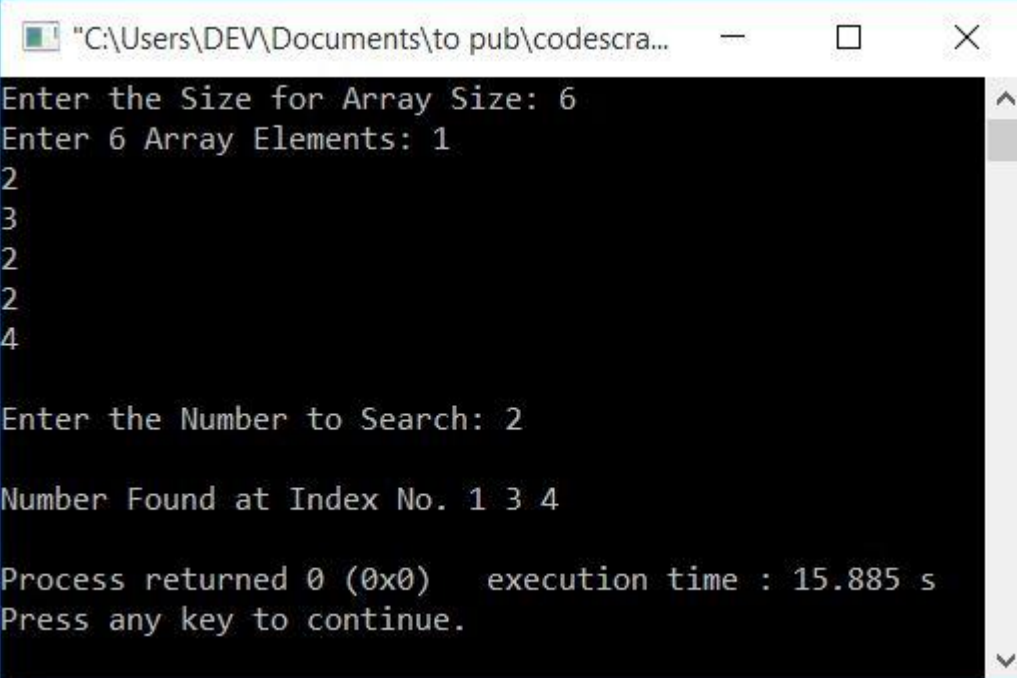
Let's have a look at the program and its sample run:

```
#include<iostream>
using namespace std;
int main()
{
    int arr[100], tot, i, num, arrTemp[50], j=0, chk=0;
    cout<<"Enter the Size for Array Size: ";
    cin>>tot;
    cout<<"Enter "<<tot<<" Array Elements: ";
    for(i=0; i<tot; i++)
        cin>>arr[i];
    cout<<"\nEnter the Number to Search: ";
    cin>>num;
    for(i=0; i<tot; i++)
    {
        if(arr[i]==num)
        {
            arrTemp[j] = i;
            j++;
            chk++;
        }
    }
    if(chk>0)
    {
        cout<<"\nNumber Found at Index No. ";
        tot = chk;
        for(i=0; i<tot; i++)
            cout<<arrTemp[i]<<" ";
    }
    else
        cout<<"\nNumber doesn't Found!";
    cout<<endl;
    return 0;
}
```

Here is its sample run with following user inputs:

- 6 as size for array
- 1, 2, 3, 2, 2, 4 as 6 array elements
- 2 as number to search

After providing these inputs, here is the sample output you will see:

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 following sequence of input and output:

```
Enter the Size for Array Size: 6
Enter 6 Array Elements: 1
2
3
2
2
4

Enter the Number to Search: 2

Number Found at Index No. 1 3 4

Process returned 0 (0x0)   execution time : 15.885 s
Press any key to continue.
```

Binary Search in C++

To search an element from an array using binary search technique in [C++](#) programming, you have to ask from user to enter any 10 elements for the [array](#) and then enter the element or number to be search.

After searching the element using binary search technique, if it is available in the list, then display the position of the element. Following C++ program asks from user to enter any 10 array elements and an element to be search. Here is the simple binary search program:

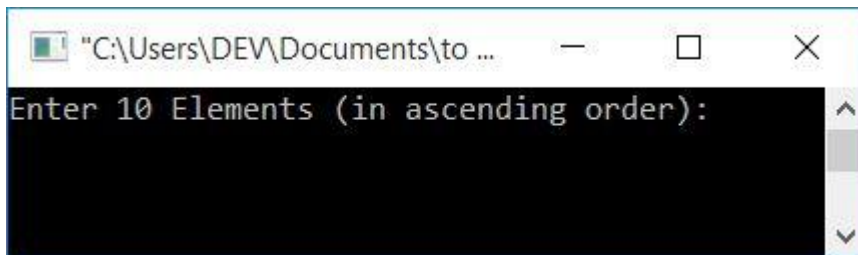
```
#include<iostream>
using namespace std;
int main()
{
    int i, arr[10], num, first, last, middle;
    cout<<"Enter 10 Elements (in ascending order): ";
    for(i=0; i<10; i++)
        cin>>arr[i];
    cout<<"\nEnter Element to be Search: ";
    cin>>num;
    first = 0;
    last = 9;
    middle = (first+last)/2;
    while(first <= last)
```

```

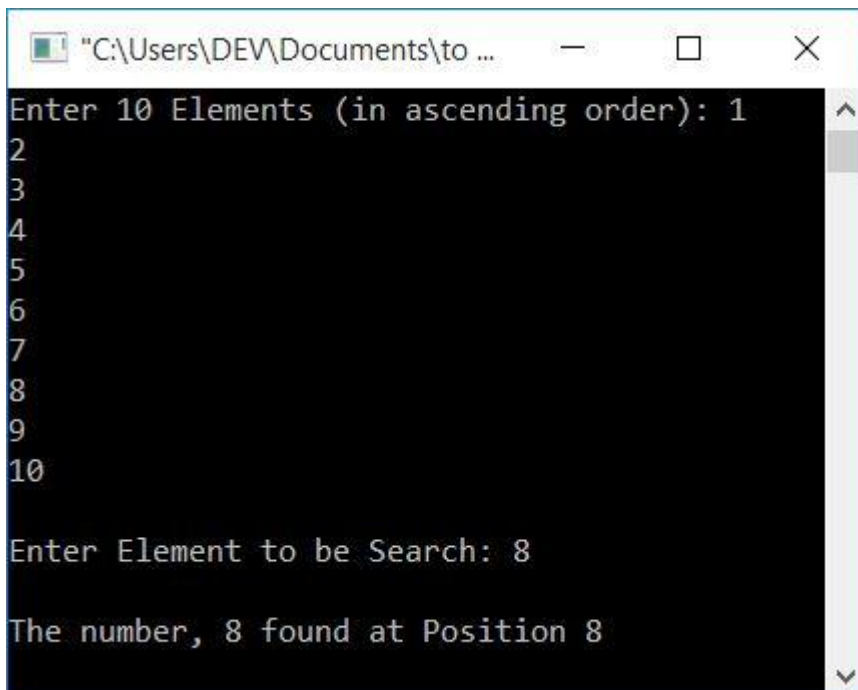
{
    if(arr[middle]<num)
        first = middle+1;
    else if(arr[middle]==num)
    {
        cout<<"\nThe number, "<<num<<" found at Position "<<middle+1;
        break;
    }
    else
        last = middle-1;
    middle = (first+last)/2;
}
if(first>last)
    cout<<"\nThe number, "<<num<<" is not found in given Array";
cout<<endl;
return 0;
}

```

This program was build and run under Code::Blocks IDE. Here is the initial snapshot of sample run:



Now enter any 10 numbers (in ascending order) as array elements say 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 and then a number that is going to be search from the list, say 8. Here is the final snapshot of sample run:



If user enters 10 elements as 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 and element to be search as 8, then dry run of above program goes like:

- 10 elements gets stored in the array arr[] in a way that, arr[0]=1, arr[1]=2, arr[2]=3, and so on
- The number to be search say 8 gets initialized to num. Therefore, num=8
- Initially first=0 and last=9
- $(\text{first}+\text{last})/2$ or $0+9/2$ or 4 gets initialized to middle. Therefore middle=4
- Because the value of first (0) is less than the value of last (9), so the condition of while loop evaluates to be true. Therefore program flow goes inside the loop
- Checks whether arr[middle]<num evaluates to be true or not. On putting the value of variables middle and num, arr[4]<num or 5<8 evaluates to be true. Therefore program flow goes inside the if block and middle+1 or 4+1 or 5 gets initialized to first
- Because, the condition of if block evaluates to be true, therefore else if and else block's statement(s) will not get executed.
- At last, middle = $(\text{first}+\text{last})/2$ or middle = $(5+9)/2$ or middle=7
- Program flow goes back to the condition of while loop. The condition first <= last or 5<=9 again evaluates to be true, therefore again program flow goes inside the loop
- Checks the condition of if block, that is arr[middle]<num evaluates to be true or not, that is arr[7]<num or 8<8 evaluates to be false. Therefore, program flow goes to else if block, and checks its condition
- The condition arr[middle]==num or 8==8 evaluates to be true. Therefore program flow goes inside this else if block and executes its statements
- That is, prints the position of number. Because the number is found at index number 7. Therefore I've printed its position as 8, since indexing in arrays starts from 0, not from 1
- After printing the position of element, use the break keyword to break out from the while loop
- After exiting from the while loop, checks the condition using if block, that is whether first's value is greater than the value of last or not
- If condition evaluates to be true, then print a message saying that the number is not found in the list

Allow User to Define Size

This program allows user to define the size of array. It also doesn't cares about the way user enters the data, that is, either in ascending or in random order. Because, after receiving all the numbers, we've created a block of code that sorts the list in ascending order (before performing the search operation). And then ask to enter an element to be search from the sorted list as shown in the program given below:

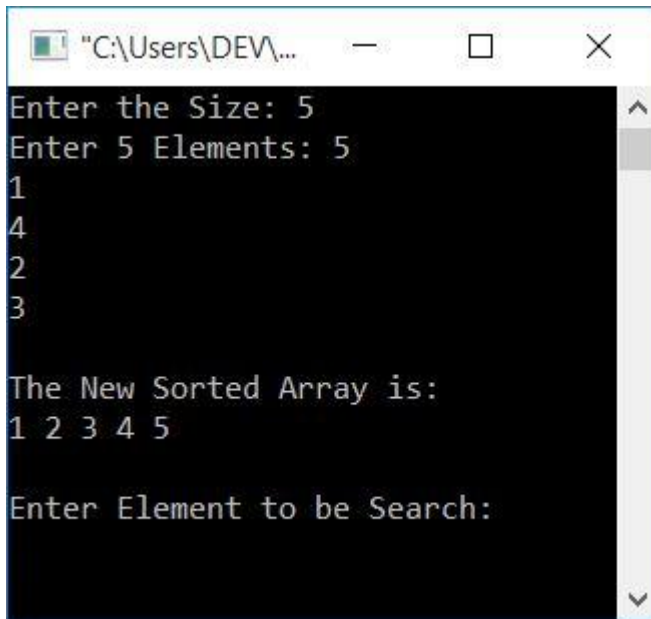
```
#include<iostream>
using namespace std;
int main()
{
    int len, i, arr[50], num, j, temp, first, last, middle;
    cout<<"Enter the Size: ";
    cin>>len;
    cout<<"Enter "<<len<<" Elements: ";
    for(i=0; i<len; i++)
```

```

        cin>>arr[i];
// sort the array first
for(i=0; i<(len-1); i++)
{
    for(j=0; j<(len-i-1); j++)
    {
        if(arr[j]>arr[j+1])
        {
            temp = arr[j];
            arr[j] = arr[j+1];
            arr[j+1] = temp;
        }
    }
}
// print the sorted array
cout<<"\nThe New Sorted Array is:\n";
for(i=0; i<len; i++)
    cout<<arr[i]<<" ";
// now back to binary search
cout<<"\n\nEnter Element to be Search: ";
cin>>num;
first = 0;
last = (len-1);
middle = (first+last)/2;
while(first <= last)
{
    if(arr[middle]<num)
        first = middle+1;
    else if(arr[middle]==num)
    {
        cout<<"\nThe number, "<<num<<" found at Position "<<middle+1;
        break;
    }
    else
        last = middle-1;
    middle = (first+last)/2;
}
if(first>last)
    cout<<"\nThe number, "<<num<<" is not found in given Array";
cout<<endl;
return 0;
}

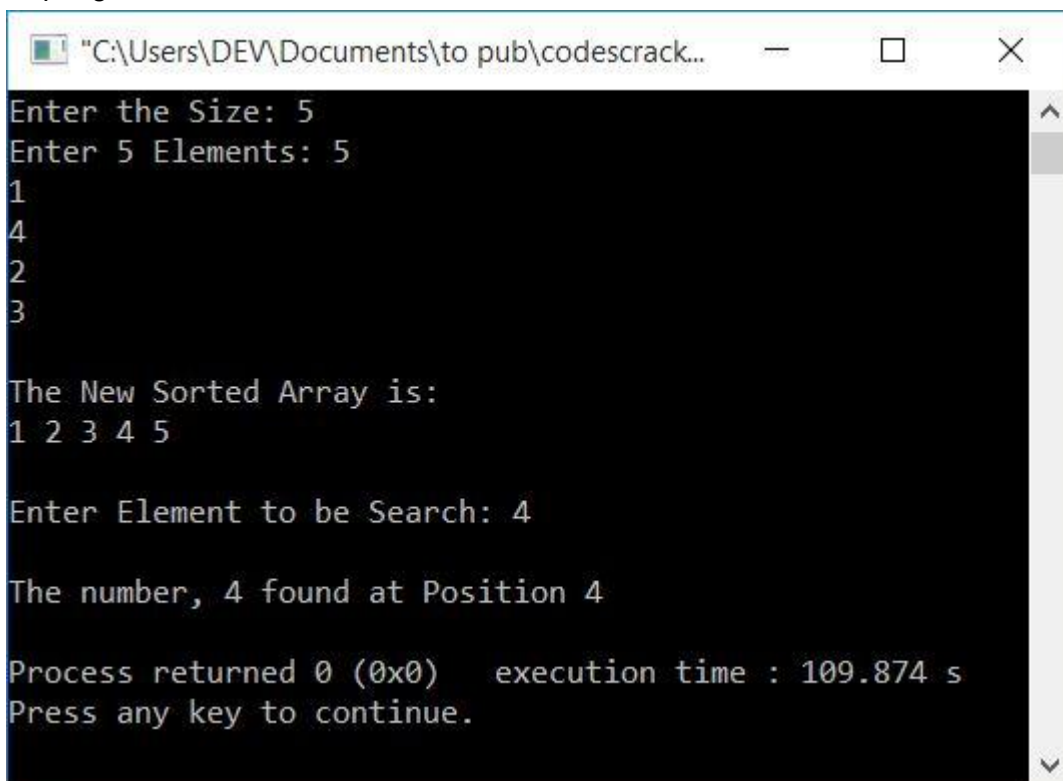
```

Here is its sample run supposing that user enters the array size as 5 and its element as 5, 1, 4, 2, 3:



```
"C:\Users\DEV\..."  
Enter the Size: 5  
Enter 5 Elements: 5  
1  
4  
2  
3  
  
The New Sorted Array is:  
1 2 3 4 5  
  
Enter Element to be Search:
```

Now enter an element say 4 to search it from the list and prints its position as shown in the output given below:



```
"C:\Users\DEV\Documents\to pub\codescrack..."  
Enter the Size: 5  
Enter 5 Elements: 5  
1  
4  
2  
3  
  
The New Sorted Array is:  
1 2 3 4 5  
  
Enter Element to be Search: 4  
  
The number, 4 found at Position 4  
  
Process returned 0 (0x0)   execution time : 109.874 s  
Press any key to continue.
```

To learn more about sorting an array, you can follow and apply any of the below sorting method in the program:

- [Bubble Sort](#)
- [Selection Sort](#)
- [Insertion Sort](#)

Using user-defined Function

This program is created using a user-defined function `binSearRecFun()` that receives three arguments. The first one is the array, second is the number to be search, and third is the size of array. This function either returns the position of element in the list or 0 that indicates the number is not available in the list.

A function named `sortArray()` is also implemented here to sort the given array in ascending order before going for the binary search.

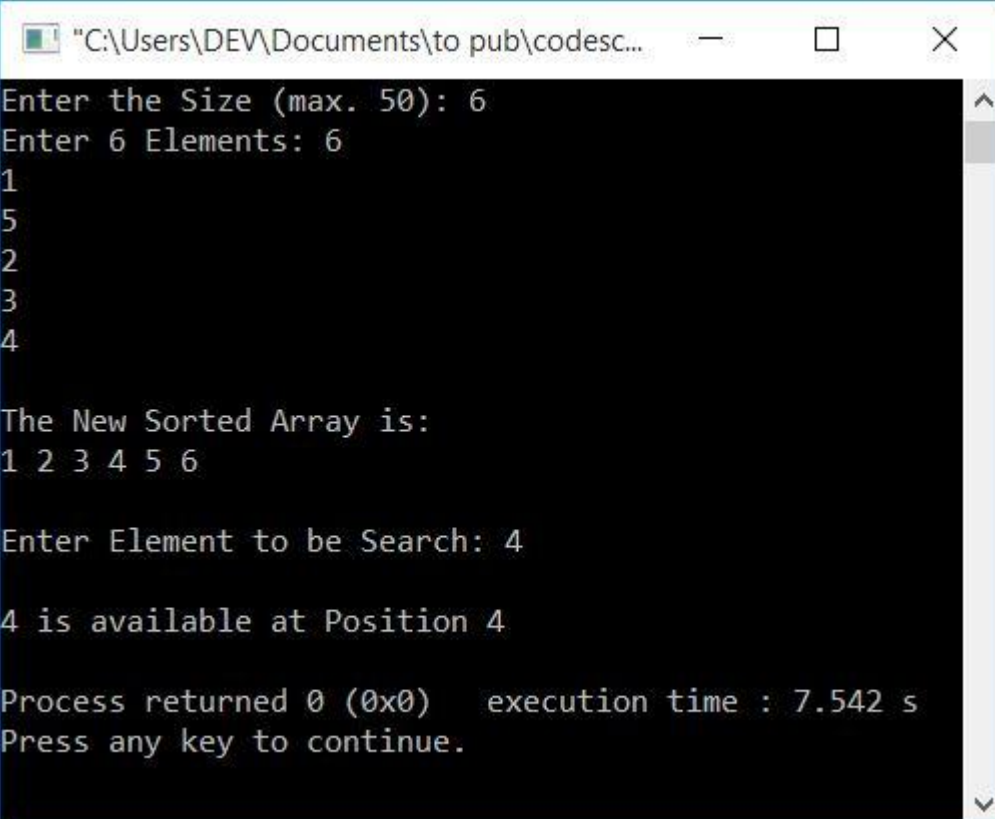
```
#include<iostream>
using namespace std;
void sortArray(int [], int);
int binSearchFun(int [], int, int);
int main()
{
    int len, i, arr[50], num, pos;
    cout<<"Enter the Size (max. 50): ";
    cin>>len;
    cout<<"Enter "<<len<<" Elements: ";
    for(i=0; i<len; i++)
        cin>>arr[i];
    // sort the array first
    sortArray(arr, len);
    // print the sorted array
    cout<<"\nThe New Sorted Array is:\n";
    for(i=0; i<len; i++)
        cout<<arr[i]<<" ";
    cout<<"\n\nEnter Element to be Search: ";
    cin>>num;
    // search element using binary search
    pos = binSearchFun(arr, num, len);
    if(pos==0)
        cout<<endl<<num<<" isn't available in the list";
    else
        cout<<endl<<num<<" is available at Position "<<pos;
    cout<<endl;
    return 0;
}
void sortArray(int arr[], int len)
{
    int i, j, temp;
    for(i=0; i<(len-1); i++)
    {
        for(j=0; j<(len-i-1); j++)
        {
            if(arr[j]>arr[j+1])
            {
                temp = arr[j];
                arr[j] = arr[j+1];
            }
        }
    }
}
```

```

        arr[j+1] = temp;
    }
}
}
}
int binSearchFun(int arr[], int num, int len)
{
    int first, last, middle;
    first = 0;
    last = (len-1);
    middle = (first+last)/2;
    while(first <= last)
    {
        if(arr[middle]<num)
            first = middle+1;
        else if(arr[middle]==num)
            return (middle+1);
        else
            last = middle-1;
        middle = (first+last)/2;
    }
    return 0;
}

```

Here is its sample run supposing that user enters size as 6, its elements as 6, 1, 5, 2, 3, 4, and the number to be search as 4.:



```

"C:\Users\DEV\Documents\to pub\codesc...
Enter the Size (max. 50): 6
Enter 6 Elements: 6
1
5
2
3
4

The New Sorted Array is:
1 2 3 4 5 6

Enter Element to be Search: 4

4 is available at Position 4

Process returned 0 (0x0)   execution time : 7.542 s
Press any key to continue.

```

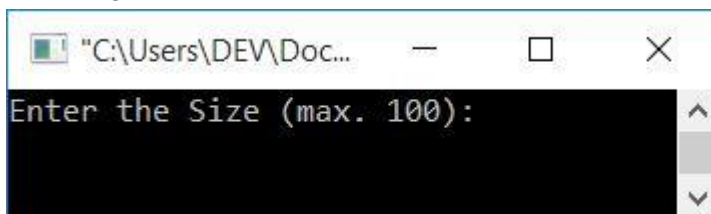

Find Largest Element in an Array without Function

To find the largest element in an [array in C++](#) programming, you have to ask from user to enter the size for array and again to enter elements of given size. Then find and print the largest number or element from given list as shown in the program given below.

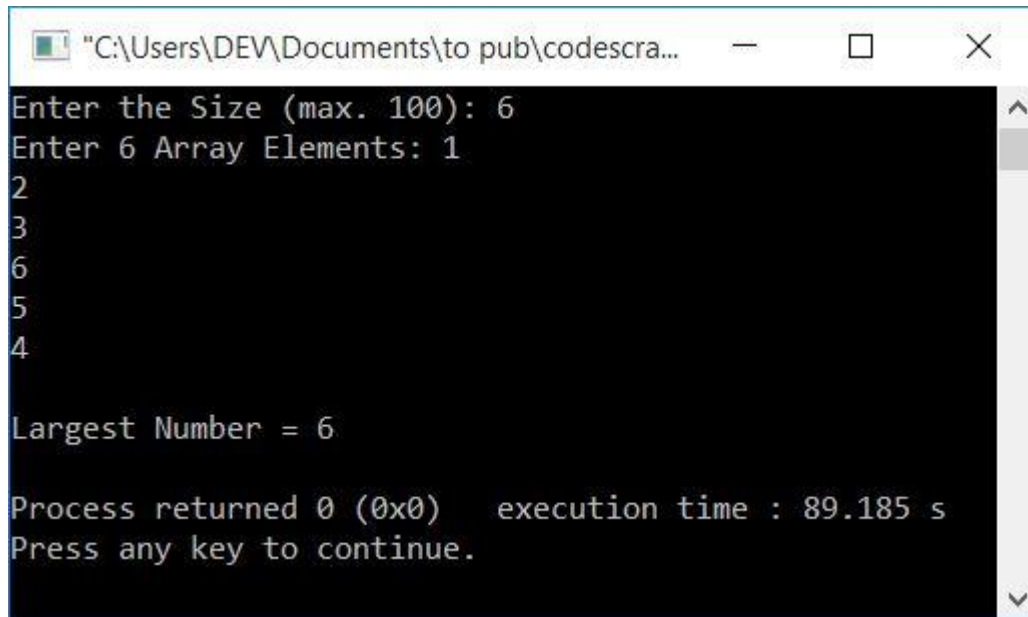
The question is, write a program in C++ to find and print largest number in an array. Here is its answer:

```
#include<iostream>
using namespace std;
int main()
{
    int arr[100], tot, larg, i;
    cout<<"Enter the Size (max. 100): ";
    cin>>tot;
    cout<<"Enter "<<tot<<" Array Elements: ";
    for(i=0; i<tot; i++)
        cin>>arr[i];
    larg = arr[0];
    for(i=1; i<tot; i++)
    {
        if(larg<arr[i])
            larg = arr[i];
    }
    cout<<"\nLargest Number = "<<larg;
    cout<<endl;
    return 0;
}
```

This program was build and run under Code::Blocks IDE. Here is its sample run:



Now enter the size for array say 6. And then enter any 6 numbers or elements. After supplying all the inputs, press ENTER key to find and print the largest number from the given list of numbers as shown in the snapshot given below:



```
"C:\Users\DEV\Documents\to pub\codescra...
Enter the Size (max. 100): 6
Enter 6 Array Elements: 1
2
3
6
5
4

Largest Number = 6

Process returned 0 (0x0)   execution time : 89.185 s
Press any key to continue.
```

The statement:

```
larg = arr[0];
```

states that, it is assumed that the largest element is present at very first index (that is, 0th) of the array. Or it is supposed that the largest element is the first element of array. From next index, we have compared the value of larg with each and every value present at rest indexes. While comparing, if the value of larg is found lesser, then the new value (value which is greater than larg's value) gets initialized to it (larg variable).

The dry run of above program with user input 6 as size, and 1, 2, 3, 6, 5, 4 as 6 array elements, goes like:

- Because array size is entered as 6. So tot=6
- And when user enters 6 array elements, then it gets stored in arr[], in this way:
 - arr[0]=1
 - arr[1]=2
 - arr[2]=3
 - arr[3]=6
 - arr[4]=5
 - arr[5]=4
- Now value at 0th index, that is 1 gets initialized to larg. So larg=1
- The execution of for loop starts. That is its initialization part gets executed at first and only at once. So i=1
- With i=1, the condition, i<tot or 1<6 evaluates to be true, therefore program flow goes inside the loop
- Inside the loop, the condition larg<arr[i] or 1<arr[1] or 1<2 evaluates to be true, therefore program flow goes inside the if's body and the value of arr[i] gets initialized to larg. So larg=2
- Now the program flow goes to update part of for loop and increments the value of i, evaluates the condition again.
- That is, i=2 and the condition, i<tot or 2<6 again evaluates to be true, therefore program flow again goes inside the loop
- This process continues, until the condition evaluates to be false

- After exiting from the loop, the variable larg holds the largest element from given array
- So print the value of larg as largest number/element.

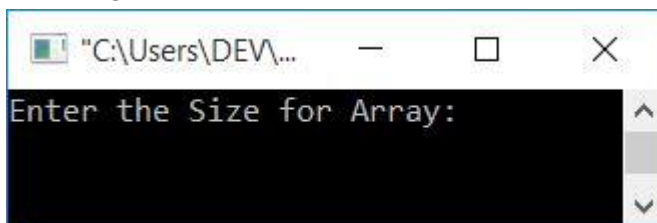
Find Smallest Number in an Array

To find the smallest element or number in an [array in C++](#) programming, you have to ask from user to enter the size and elements for array. Now find and print the smallest one as shown in the program given below:

The question is, write a program in C++ to find and print smallest number in an array. Here is its answer:

```
#include<iostream>
using namespace std;
int main()
{
    int arr[100], tot, i, s;
    cout<<"Enter the Size for Array: ";
    cin>>tot;
    cout<<"Enter "<<tot<<" Array Elements: ";
    for(i=0; i<tot; i++)
        cin>>arr[i];
    s = arr[0];
    for(i=1; i<tot; i++)
    {
        if(s>arr[i])
            s = arr[i];
    }
    cout<<"\nSmallest Number = "<<s;
    cout<<endl;
    return 0;
}
```

This program was build and run under Code::Blocks IDE. Here is its sample run:



Now supply any number say 5 as array size, and then enter 5 numbers as 5 array elements. After supplying all these things, press ENTER key to find and print the smallest number from the array as shown in the snapshot given below:

The following statement:

```
s = arr[0];
```

supposed that the number at 0th (very first) index is the smallest number. And then we've compared all the number at remaining indexes one by one with the number at s. If s's value is found greater than any number at any index, then we've initialized that number as new value of s.

The dry run of above program goes like:

- When user enters the size say 5 for the array, then it gets stored in tot. So tot=5
- Now we've created a for loop that executes 5 times. That is, the loop variable, i starts from 0 to 4
- So at first time, the first number gets stored in arr[0], second number in arr[1] and so on
- Therefore, if user enters 5 numbers as 5, 4, 3, 1, 2, then it gets stored in arr[] in this way:
 - arr[0]=5
 - arr[1]=4
 - arr[2]=3
 - arr[3]=1
 - arr[4]=2
- Now the statement that is used to suppose the first number as the smallest number. That is, arr[0] or 5 gets initialized to s
- Now using the second for loop, we've compared the value of s with each and every element of arr[]
- That is, at first execution of for loop, 1 gets initialized to i, and the condition i<tot or 1<5 evaluates to be true, therefore program flow goes inside the loop, and the condition s>arr[i] or 5>arr[1] or 5>4 evaluates to be true, therefore program flow goes inside the if's body and the value of arr[i] or arr[1] or 4 gets initialized to s
- Now program flow goes to update part of for loop and increments the value of i. So i=2
- Now the condition of for loop again gets evaluated. Every time before entering into the body of loop, condition must evaluated to be true
- This time again the condition i<tot or 2<5 evaluates to be true, therefore program flow again goes inside the loop

- This process continues until the condition evaluates to be false
- After exiting from the loop, the variable s holds the smallest number from given list of numbers stored in arr[]
- So just print the value of s as smallest number from an array entered by user

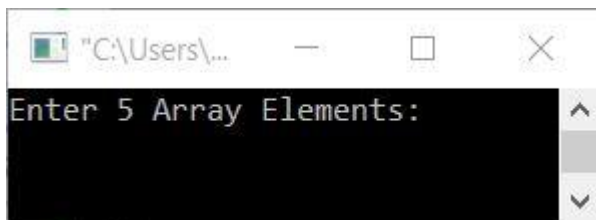
Insert Element at the End of an Array

This program asks from user to enter 5 numbers or elements for an array and then further ask to enter the element to insert it at the end of given array.

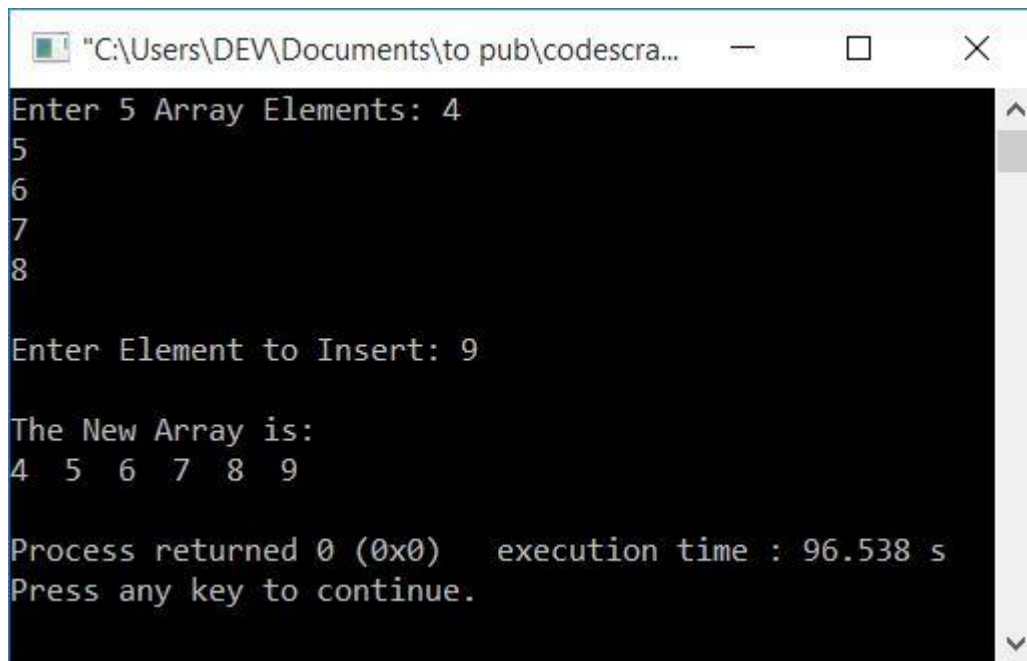
The question is, write a program in C++ to insert an element at the end of an array. Here is its answer:

```
#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;
}
```

This program was build and run under Code::Blocks IDE. Here is its initial output:



Now supply 5 inputs as 5 numbers or elements for the array say 4, 5, 6, 7, 8 and then enter a number or element say 9 to insert in the array. Here is the sample output produced by above program after providing these inputs:

A screenshot of a Windows command prompt window titled "C:\Users\DEV\Documents\to pub\codescra...". The window has a black background with white text. The text shows the program's execution: it prompts for 5 array elements (4, 5, 6, 7, 8), then prompts for an element to insert (9), displays the new array (4 5 6 7 8 9), and shows the process return code and execution time (96.538 s).

```
"C:\Users\DEV\Documents\to pub\codescra...  
Enter 5 Array Elements: 4  
5  
6  
7  
8  
  
Enter Element to Insert: 9  
  
The New Array is:  
4 5 6 7 8 9  
  
Process returned 0 (0x0)   execution time : 96.538 s  
Press any key to continue.
```

This programs receives 5 array elements using for loop, one by one. That is, if user enters 5 array elements as 4, 5, 6, 7, 8, then these elements gets stored in `arr[]` in following way:

- `arr[0]=4`
- `arr[1]=5`
- `arr[2]=6`
- `arr[3]=7`
- `arr[4]=8`

After receiving all the 5 elements for the array, the value of `i` is 5 now. So just receive another input, that is the element to insert and initialize it to `arr[i]` or `arr[5]`. Finally print the new array.

Insert Element in Array at a Specific Position

To insert an element in an [array](#) in C++ programming, you have to ask from user to enter the size and elements for 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 desired position, don't forgot 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 ? ";
```

```

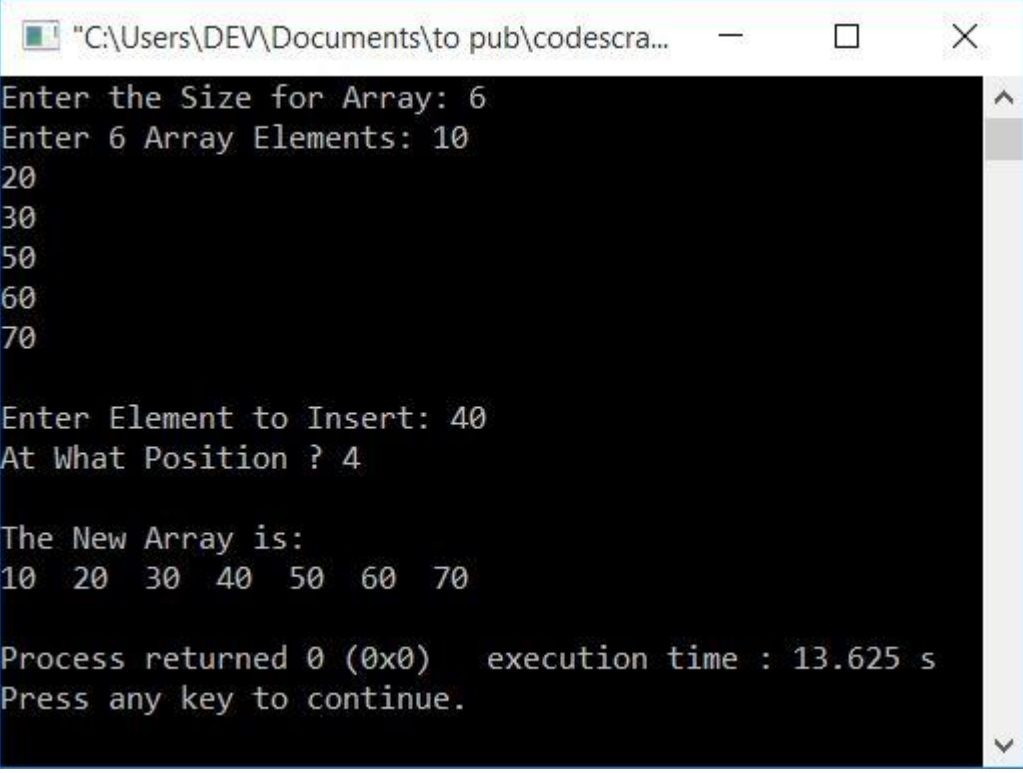
    cin>>pos;
    for(i=tot; i>=pos; i--)
        arr[i] = arr[i-1];
    arr[i] = elem;
    tot++;
    cout<<"\n\nThe New Array is:\n\n";
    for(i=0; i<tot; i++)
        cout<<arr[i]<<" ";
    cout<<endl;
    return 0;
}

```

Here is its sample run with following user input:

- 6 as array size
- 10, 20, 30, 50, 60, 70 as 6 array elements
- 40 as 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 has been done on the output screen:



```

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 above program, with the help of pos's value (desired position where user wants to insert the new element), we've shifted all the element from this position to one index forth (forward). After doing this task, the previous element at this position gets moved to its one index forth. Therefore, we're free to initialize the new element at this position

Don't forgot to increment the value of tot (size of array) after inserting the new element to it. Now print the new array that contains newly inserted element at desired or specific position as shown in the output given above.

For example, the dry run of following block of code:

```
for(i=tot; i>=pos; i--)
```

```
arr[i] = arr[i-1];  
arr[i] = elem;
```

with same user input as provided in above sample run, goes like:

- Here are the values before program flow come at this block of code:
 - tot=6
 - arr[0]=10, arr[1]=20, arr[2]=30, arr[3]=50, arr[4]=60, arr[5]=70
 - elem=40
 - pos=4
- Now the execution of this small block of code, starts with for loop
- That is, its initialization part executes at first but only at once. So the value of tot gets initialized to i. Therefore, i=6
- Now the condition, i>=pos or 6>=4 evaluates to be true, therefore program flow goes inside the loop
- Inside the loop, arr[i-1] or arr[6-1] or arr[5] or 70 gets initialized to arr[i] or arr[6]. So arr[6]=70. As you can see that the number at last index (fifth index) gets moved to one index forth (that is at sixth index)
- Now the program flow goes to the update part of for loop decrements the value of i. So i=5, and evaluates the condition again
- That is the condition i>=pos or 5>=4 evaluates to be true again, therefore program flow again goes inside the loop
- This process continues, until the condition of for loop evaluates to be false. Before its condition evaluated to be false, here are the new values of arr[]:
 - arr[6]=70
 - arr[5]=60
 - arr[4]=50
- Now the element or number 50 which was present at fourth position (third index) gets moved to fifth position (forth index). Because indexing starts from 0, therefore we're considering the element at fourth index as fifth position's element
- Therefore the fourth position (third index) is free to initialize any new element
- So just initialize the new element to it. Therefore, the value of elem, that is 40 gets initialized to arr[i] or arr[3]