

C++ Dynamic Allocation of Arrays with Example

By Barbara Thompson ⓘ Updated January 1, 2022

What is a Dynamic Array?

A dynamic array is quite similar to a regular array, but its size is modifiable during program runtime. Dynamic array elements occupy a contiguous block of memory.

Once an array has been created, its size cannot be changed. However, a dynamic array is different. A dynamic array can expand its size even after it has been filled.

During the creation of an array, it is allocated a predetermined amount of memory. This is not the case with a dynamic array as it grows its memory size by a certain factor when there is a need.

In this C++ tutorial, you will learn

- [What is a Dynamic Array?](#)
- [Factors impacting performance of Dynamic Arrays](#)
- [The new Keyword](#)
- [Initializing dynamically allocated arrays](#)
- [Resizing Arrays](#)
- [Dynamically Deleting Arrays](#)

Factors impacting performance of Dynamic Arrays

The array's initial size and its growth factor determine its performance. Note the following points:

1. If an array has a small size and a small growth factor, it will keep on reallocating memory more often. This will reduce the performance of the array.
2. If an array has a large size and a large growth factor, it will have a huge chunk of unused memory. Due to this, resize operations may take longer. This will reduce the

The new Keyword

In C++, we can create a dynamic array using the new keyword. The number of items to be allocated is specified within a pair of square brackets. The type name should precede this. The requested number of items will be allocated.

Syntax:

The new keyword takes the following syntax:

```
pointer_variable = new data_type;
```

The `pointer_variable` is the name of the pointer variable.

The `data_type` must be a valid C++ data type.

The keyword then returns a pointer to the first item. After creating the dynamic array, we can delete it using the delete keyword.

Example 1:

```
#include<iostream>
using namespace std;
int main() {
    int x, n;
    cout << "Enter the number of items:" << "\n";
    cin >>n;
    int *arr = new int(n);
    cout << "Enter " << n << " items" << endl;
    for (x = 0; x < n; x++) {
        cin >> arr[x];
    }
    cout << "You entered: ";
    for (x = 0; x < n; x++) {
        cout << arr[x] << " ";
    }
}
```

```
        return 0;
    }
```

Output:

```
Enter the number of items:
5
Enter 5 items
1 4 5 6 7
You entered: 1 4 5 6 7
```

Here is a screenshot of the code:

```
ConsoleApplication2 (Global Scope)
1  #include<iostream>
2
3  using namespace std;
4
5  int main() {
6
7      int x, n;
8
9      cout << "Enter the number of items:" << "\n";
10     cin >> n;
11
12     int *arr = new int(n);
13
14     cout << "Enter " << n << " items" << endl;
15
16     for (x = 0; x < n; x++) {
17         cin >> arr[x];
18     }
19
20     cout << "You entered: ";
21
22     for (x = 0; x < n; x++) {
23         cout << arr[x] << " ";
24     }
25
26     return 0;
27 }
```

Code Explanation:

1. Include the `iostream` header file into our program to use its functions.

3. Call the main() function. The program logic should be added within the body of the function.
4. Declare two integer variables x and n.
5. Print some text on the console prompting the user to enter the value of variable n.
6. Read user input from the keyboard and assigning it to variable n.
7. Declare an array to hold a total of n integers and assigning it to pointer variable *arr.
8. Print a message prompting the user to enter n number of items.
9. Use a for loop to create a loop variable x to iterate over the items entered by the user.
10. Read the elements entered by the user and storing them in the array arr.
11. End of the body of the for loop.
12. Print some text on the console.
13. Use a for loop to create a loop variable x to iterate over the items of the array.
14. Print out the values contained in the array named arr on the console.
15. End of the body of the for loop.
16. The program must return value upon successful completion.
17. End of the body of the main() function.

NOTE: In the above example, the user is allowed to specify any size for the array during run time. This means the array's size is determined during runtime.

Initializing dynamically allocated arrays

It's easy to initialize a dynamic array to 0.

Syntax:

```
int *array{ new int[length]{} };
```

In the above syntax, the length denotes the number of elements to be added to the array. Since we need to initialize the array to 0, this should be left empty.

We can initialize a dynamic array using an initializer list. Let's create an example that demonstrates this.

Example 2:

```
#include <iostream>
using namespace std;

int main(void) {

    int x;

    int *array{ new int[5]{ 10, 7, 15, 3, 11 } };

    cout << "Array elements: " << endl;

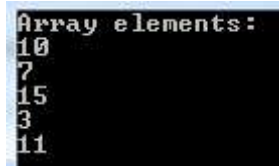
    for (x = 0; x < 5; x++) {

        cout << array[x] << endl;

    }

    return 0;
}
```

Output:

A screenshot of a terminal window showing the output of the C++ program. The text is white on a black background. It displays "Array elements:" followed by five lines of numbers: 10, 7, 15, 3, and 11, each on a new line.

```
Array elements:
10
7
15
3
11
```

Here is a screenshot of the code:

```

1  #include <iostream> 1
2
3  using namespace std; 2
4
5  int main(void) { 3
6
7      int x; 4
8
9      int *array{ new int[5]{ 10, 7, 15, 3, 11 } }; 5
10
11     cout << "Array elements: " << endl; 6
12
13     for (x = 0; x < 5; x++) { 7
14
15         cout << array[x] << endl; 8
16     } 9
17
18     return 0; 10
19 } 11
20
21
22

```

Code Explanation:

1. Include the iostream header file into our program to use its functions.
2. Include the std namespace in our program to use its classes without calling it.
3. Call the main() function. The program logic should be added within the body of the function.
4. Declare an integer variable named x.
5. Declare a dynamic array named array using an initializer list. The array will hold 5 integer elements. Note that we've not used the "=" operator between the array length and the initializer list.
6. Print some text on the console. The endl is a C++ keyword that means end line. It moves the cursor to the next sentence.
7. Use a for loop to iterate over the array elements.
8. Print the contents of the array named array on the console.
9. End of the body of the for loop.
10. The program must return value upon successful completion.
11. End of the body of the main() function.

Resizing Arrays

The length of a dynamic array is set during the allocation time.

However, C++ doesn't have a built-in mechanism of resizing an array once it has been allocated.

You can, however, overcome this challenge by allocating a new array dynamically, copying over the elements, then erasing the old array.

Note: that this technique is prone to errors, hence, try to avoid it.

Dynamically Deleting Arrays

A dynamic array should be deleted from the computer memory once its purpose is fulfilled. The delete statement can help you accomplish this. The released memory space can then be used to hold another set of data. However, even if you do not delete the dynamic array from the computer memory, it will be deleted automatically once the program terminates.

Note:

To delete a dynamic array from the computer memory, you should use delete[], instead of delete. The [] instructs the CPU to delete multiple variables rather than one variable. The use of delete instead of delete[] when dealing with a dynamic array may result in problems. Examples of such problems include memory leaks, data corruption, crashes, etc.

Example 3:

```
#include<iostream>
using namespace std;
int main() {
    int x, n;
    cout << "How many numbers will you type?" << "\n";
    cin >>n;
    int *arr = new int(n);
    cout << "Enter " << n << " numbers" << endl;
    for (x = 0; x < n; x++) {
        cin >> arr[x];
    }
    cout << "You typed: ";
    for (x = 0; x < n; x++) {
        cout << arr[x] << " ";
    }
}
```

```

    cout << endl;
    delete [] arr;
    return 0;
}

```

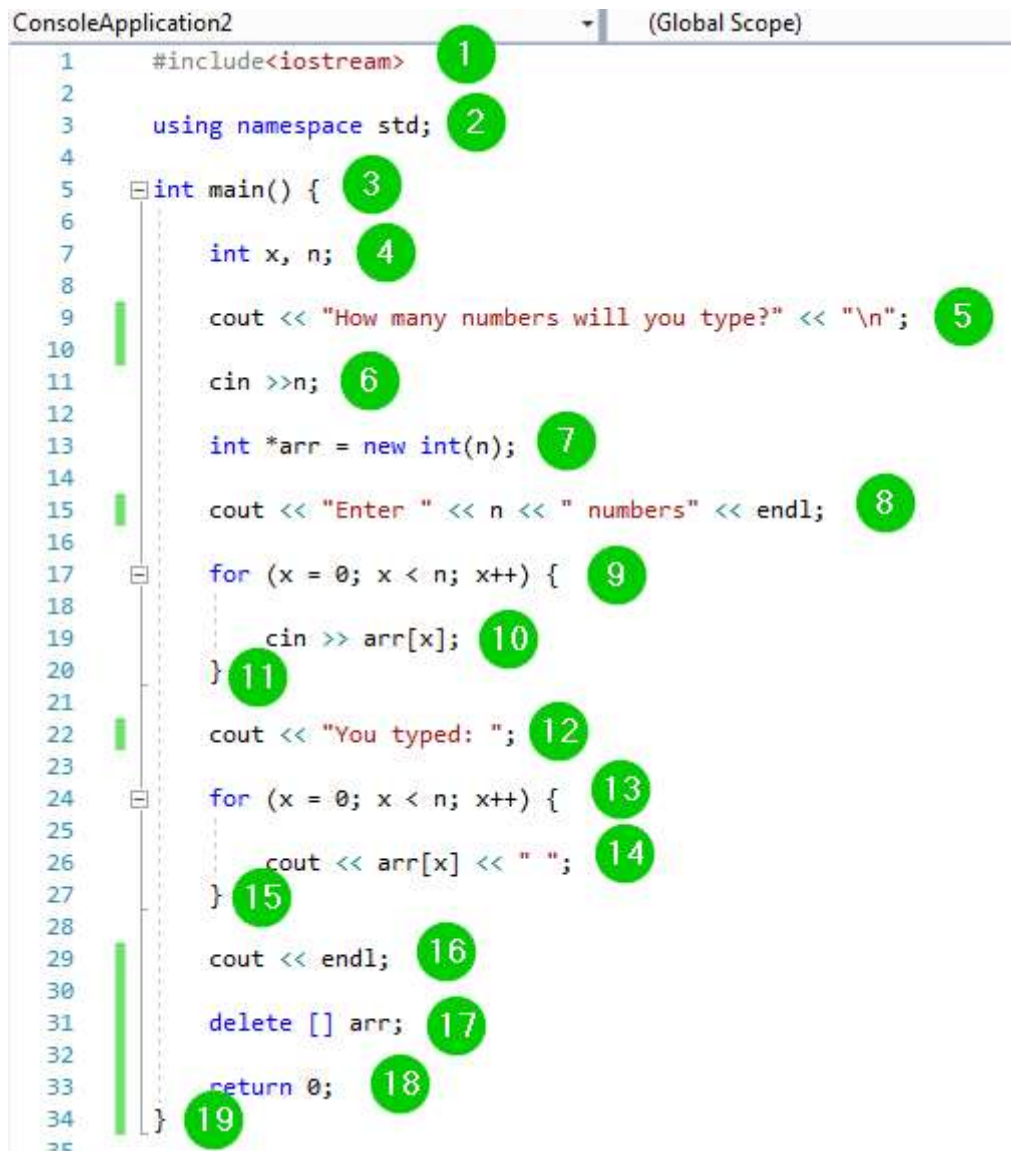
Output:

```

How many numbers will you type?
4
Enter 4 numbers
2 4 6 9
You typed: 2 4 6 9

```

Here is a screenshot of the code:



1. Include the `iostream` header file in our program in order to use its functions.
2. Include the `std` namespace in our program in order to use its classes without calling it.
3. Call the `main()` function. The program logic should be added within the body of the function.
4. Declare two variables `x` and `n` of the integer data type.
5. Print some text on the console. The text will ask the user to state the number of numbers they will enter.
6. Read user input from the keyboard. The input value will be assigned to variable `n`.
7. Declare a pointer variable `*arr`. The array `arr` will reserve some memory to store a total of `n` integers.
8. Print a message on the console prompting the user to enter `n` numbers.
9. Create a `for` loop and the loop variable `x` to iterate over the numbers entered by the user.
10. Read the numbers entered by the user and storing them in the array `arr`.
11. End of the body of the `for` loop.
12. Print some text on the console.
13. Use a `for` loop and the loop variable `x` to iterate over the contents of array `arr`.
14. Print out the values of the array `arr` on the console.
15. End of the body of the `for` loop.
16. Print an empty line on the console.
17. Free up the memory of the array `arr`.
18. The program will return value when it completes successfully.
19. End of the body of the `main()` function.

Summary:

- Regular arrays have a fixed size. You cannot modify their size once declared.
- With these types of arrays, the memory size is determined during compile time.
- Dynamic arrays are different. Their sizes can be changed during runtime.
- In dynamic arrays, the size is determined during runtime.
- Dynamic arrays in C++ are declared using the `new` keyword.
- We use square brackets to specify the number of items to be stored in the dynamic array.
- Once done with the array, we can free up the memory using the `delete` operator.
- Use the `delete` operator with `[]` to free the memory of all array elements.