**Initializing arrays.**

By default, regular arrays of *local scope* (for example, those declared within a function) are left uninitialized. This means that none of its elements are set to any particular value; their contents are undetermined at the point the array is declared.

But the elements in an array can be explicitly initialized to specific values when it is declared, by enclosing those initial values in braces {}. For example:

|  |  |  |
| --- | --- | --- |
|  | int foo [5] = { 16, 2, 77, 40, 12071 }; |  |

This statement declares an array that can be represented like this:

https://cplusplus.com/doc/tutorial/arrays/arrays2.png

The number of values between braces {} shall not be greater than the number of elements in the array. For example, in the example above, *foo* was declared having 5 elements (as specified by the number enclosed in square brackets, []), and the braces {} contained exactly 5 values, one for each element. If declared with less, the remaining elements are set to their default values (which for fundamental types, means they are filled with zeroes). For example:

|  |  |  |
| --- | --- | --- |
|  | int bar [5] = {10, 20, 30}; |  |

Will create an array like this:

https://cplusplus.com/doc/tutorial/arrays/arrays3.png

The initializer can even have no values, just the braces:

|  |  |  |
| --- | --- | --- |
|  | int baz [5] = {}; |  |

This creates an array of five int values, each initialized with a value of zero:

https://cplusplus.com/doc/tutorial/arrays/arrays4.png

When an initialization of values is provided for an array, C++ allows the possibility of leaving the square brackets empty []. In this case, the compiler will assume automatically a size for the array that matches the number of values included between the braces {}:

|  |  |  |
| --- | --- | --- |
|  | int foo [] = { 16, 2, 77, 40, 12071 }; |  |

After this declaration, array *foo* would be 5 int long, since we have provided 5 initialization values.

Finally, the evolution of C++ has led to the adoption of *universal initialization* also for arrays. Therefore, there is no longer need for the equal sign between the declaration and the initializer. Both these statements are equivalent:

|  |  |  |
| --- | --- | --- |
| 1 2 | int foo[] = { 10, 20, 30 };  int foo[] { 10, 20, 30 }; |  |

Static arrays, and those declared directly in a namespace (outside any function), are always initialized. If no explicit initializer is specified, all the elements are default-initialized (with zeroes, for fundamental types).