**Accessing the values of an array.**

The values of any of the elements in an array can be accessed just like the value of a regular variable of the same type. The syntax is:

*name[index]*

Following the previous examples in which *foo* had 5 elements and each of those elements was of type int, the name which can be used to refer to each element is the following:

https://cplusplus.com/doc/tutorial/arrays/arrays5.png

For example, the following statement stores the value 75 in the third element of *foo*:

|  |  |  |
| --- | --- | --- |
|  | foo [2] = 75; |  |

and, for example, the following copies the value of the third element of *foo* to a variable called *x*:

|  |  |  |
| --- | --- | --- |
|  | x = foo[2]; |  |

Therefore, the expression *foo[2]* is itself a variable of type *int*.

Notice that the third element of *foo* is specified *foo[2]*, since the first one is *foo[0]*, the second one is *foo[1]*, and therefore, the third one is *foo[2]*. By this same reason, its last element is *foo[4]*. Therefore, if we write *foo[5]*, we would be accessing the sixth element of *foo*, and therefore actually exceeding the size of the array.

In C++, it is syntactically correct to exceed the valid range of indices for an array. This can create problems, since accessing out-of-range elements do not cause errors on compilation, but can cause errors on runtime. The reason for this being allowed will be seen in a later chapter when pointers are introduced.

At this point, it is important to be able to clearly distinguish between the two uses that square brackets [] have related to arrays. They perform two different tasks: one is to specify the size of arrays when they are declared; and the second one is to specify indices for concrete array elements when they are accessed. Do not confuse these two possible uses of brackets [] with arrays.

|  |  |  |
| --- | --- | --- |
| 1 2 | int foo[5]; // declaration of a new array  foo[2] = 75; // access to an element of the array. |  |

The main difference is that the declaration is preceded by the type of the elements, while the access is not.

Some other valid operations with arrays:

|  |  |  |
| --- | --- | --- |
| 1 2 3 4 | foo[0] = a;  foo[a] = 75;  b = foo [a+2];  foo[foo[a]] = foo[2] + 5; |  |

For example:

|  |  |  |  |
| --- | --- | --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 | // arrays example  #include <iostream>  using namespace std;  int foo [] = {16, 2, 77, 40, 12071};  int n, result=0;  int main ()  {  for (n=0; n<5 ; ++n)  {  result += foo[n];  }  cout << result;  return 0;  } | 12206 | [Edit & Run](https://cplusplus.com/doc/tutorial/arrays/) |