**Reference operator (&).**

The address of a variable can be obtained by preceding the name of a variable with an ampersand sign (&), known as reference operator, and which can be literally or even metaphorically transliterated as *address-of operator*. For example:

|  |  |  |
| --- | --- | --- |
|  | foo = &myvar; |  |

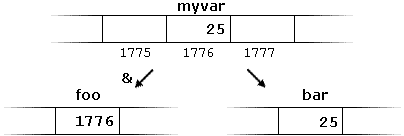
This would assign the address of variable *myvar* to *foo*; by preceding the name of the variable *myvar* with the *address-of operator (reference operator)* (&), we are no longer assigning the content of the variable itself to *foo*, but its address.

The actual address of a variable in memory cannot be known before runtime, but let's assume, in order to help clarify some concepts, that *myvar* is placed during runtime in the memory address 1776.

In this case, consider the following code fragment:

|  |  |  |
| --- | --- | --- |
| 1 2 3 | myvar = 25;  foo = &myvar;  bar = myvar; |  |

The values contained in each variable after the execution of this are shown in the following diagram:



First, we have assigned the value 25 to *myvar* (a variable whose address in memory we assumed to be 1776).

The second statement assigns *foo* the address of *myvar*, which we have assumed to be 1776.

Finally, the third statement, assigns the value contained in *myvar* to *bar*. This is a standard assignment operation, as already done many times in earlier chapters.

The main difference between the second and third statements is the appearance of the *reference operator* (&) (*address-of operator*).

The variable that stores the address of another variable (like *foo* in the previous example) is what in C++ is called a *pointer*. Pointers are a very powerful feature of the language that has many uses in lower level programming. A bit later, we will see how to declare and use pointers.

Abovementioned operations:

|  |  |  |  |
| --- | --- | --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12  13  14  15  16  17 | #include "iostream"  using namespace std;  int main()  {  auto myvar = 25;  auto foo = &myvar;  decltype(myvar) bar = myvar;  auto &bak = myvar;  cout << myvar << endl;  cout << foo << endl;  cout << bar << endl;  cout << bak << endl;  cout << &bak << endl;  cout << &myvar << endl;  cout << &foo << endl;  return 0;  } | 25  012FF738  25  25  012FF738  012FF738  012FF730 | [Edit & Run](https://www32.cplusplus.com/doc/tutorial/ntcs/) |

\*Notice how *foo* and *&foo* differ. *foo* stores the address of *myvar*, whereas *&foo* stores the address of *foo* itself.