**Using namespace std.**

If you have seen C++ code before, you may have seen *cout* being used instead of *std::cout*. Both name the same object: the first one uses its *unqualified name* (*cout*), while the second qualifies it directly within the *namespace* std (as *std::cout*).

*cout* is part of the standard library, and all the elements in the standard C++ library are declared within what is called a *namespace*: the namespace std.

In order to refer to the elements in the std namespace a program shall either qualify each and every use of elements of the library (as we have done by prefixing *cout* with *std::*), or introduce visibility of its components. The most typical way to introduce visibility of these components is by means of *using declarations*:

|  |  |  |
| --- | --- | --- |
|  | using namespace std; |  |

The above declaration allows all elements in the std namespace to be accessed in an *unqualified* manner (without the std:: prefix).

With this in mind, the last example can be rewritten to make unqualified uses of *cout* as:

|  |  |  |  |
| --- | --- | --- | --- |
| 1 2 3 4 5 6 7 8 9 | // my second program in C++  #include <iostream>  using namespace std;  int main ()  {  cout << "Hello World! ";  cout << "I'm a C++ program";  } | Hello World! I'm a C++ program | [Edit & Run](https://cplusplus.com/doc/tutorial/program_structure/) |

Both ways of accessing the elements of the *std namespace* (explicit qualification and *using* declarations) are valid in C++ and produce the exact same behavior. For simplicity, and to improve readability, the examples in these tutorials will more often use this latter approach with *using* declarations, although note that *explicit qualification* is the only way to guarantee that name collisions never happen.

Namespaces are explained in more detail in a later chapter.