Using Namespaces

Namespaces have to be written exactly as they are, however, the 'using' method allows us to make namespaces simpler.

If you use qualified names, you have to use them exactly as defined. For each namespace you must put the scope resolution operator `::`. More libraries of the C++ standard library use nested namespaces.

```
#include <iostream>
#include <chrono>
...
std::cout << "Hello world:" << std::endl;
auto timeNow= std::chrono::system_clock::now();
```

Unqualified Use of Names

You can use names in C++ with the `using` declaration and the `using` directive.

Using Declaration

A `using` declaration adds a name to the visibility scope, in which you applied the `using` declaration:

```
#include <iostream>
#include <chrono>
...
using std::cout;
using std::endl;
using std::chrono::system_clock;
```

```
...
cout << "Hello world:" << endl; // unqualified name
auto timeNow= now();            // unqualified name
```

The application of a `using` declaration has the following consequences:

- An ambiguous lookup and therefore a compiler error occurs if the same name was declared in the same visibility scope.
- If the same name was declared in a surrounding visibility scope, it will be hidden by the `using` declaration.

Using Directive

The `using` directive permits it to use all names of a namespace without qualification.

```
#include <iostream>
#include <chrono>
...
using namespace std;
...
cout << "Hello world:" << endl;           // unqualified name
auto timeNow= chrono::system_clock::now(); // partially qualified name
```

A `using` directive adds no name to the current visibility scope; it only makes the name accessible. That implies:

- An ambiguous lookup and therefore a compiler error occurs if the same name was declared in the same visibility scope.
- A name in the local namespace hides a name declared in a surrounding namespace.
- An ambiguous lookup and therefore a compiler error occurs if the same name gets visible from different namespaces or if a name in the namespace hides a name in the global scope.

> **i Use `using` directives with great care in source files**
> `using` directives should be used with great care in source files, because by the directive `using namespace std` all names from `std` becomes visible. That includes names, which accidentally hide names in the local or surrounding namespace.
>
> Don't use `using` directives in header files. If you include a header with `using namespace std` directive, all names from `std` become visible.

Namespace Alias

A namespace alias defines a synonym for a namespace. It's often convenient to use an alias for a long namespace or nested namespaces:

```
#include <chrono>
...
namespace sysClock= std::chrono::system_clock;
auto nowFirst= sysClock::now();
auto nowSecond= std::chrono::system_clock::now();
```

Because of the namespace alias, you can address the `now` function qualified and with the alias. A namespace alias must not hide a name.

Now, let's talk about the final step of using libraries in C++ – building executables.