4.2. Header Files

The use of namespace `std` for all identifiers of the C++ standard library was introduced during the standardization process. This change is not backward compatible to old header files, in which identifiers of the C++ standard library are declared in the global scope. In addition, some interfaces of classes changed during the standardization process (however, the goal was to stay backward compatible if possible). So, a new style for the names of standard header files was introduced, thus allowing vendors to stay backward compatible by providing the old header files.

The definition of new names for the standard header files was a good opportunity to standardize the extensions of header files. Previously, several extensions for header files were used; for example, `.h`, `.hpp`, and `.hxx`. However, the new standard extension for header files might be a surprise: Standard headers no longer have extensions. Hence, `include` statements for standard header files look like this:

```
#include <iostream>
#include <string>
```

This convention also applies to header files assumed from the C standard. C header files now have the new prefix `c` instead of the old extension `.h`:

```
#include <cstdlib>       // was: <stdlib.h>
#include <cstring>       // was: <string.h>
```

Inside these header files, all identifiers are declared in namespace `std`.

One advantage of this naming scheme is that you can distinguish the old string header for `char*` C functions from the new string header for the standard C++ class `string`:

```
#include <string>       // C++ class string
#include <cstring>      // char* functions from C
```

The new naming scheme of header files does not necessarily mean that the filenames of standard header files have no extensions from the point of view of the operating system. How `include` statements for standard header files are handled is implementation defined. C++ systems might add an extension or even use built-in declarations without reading a file. In practice, however, most systems simply include the header from a file that has exactly the same name as used in the `include` statement. So, in most systems, C++ *standard* header files simply have no extension. In general, it is still a good idea to use a certain extension for your own header files to help identify them in a file system.

To maintain compatibility with C, the "old" standard C header files are still available. So if necessary, you can still use, for example:

```
#include <stdlib.h>
```

In this case, the identifiers are declared in both the global scope and namespace `std`. In fact, these headers behave as if they declare all identifiers in namespace `std`, followed by an explicit using declaration.

For the C++ header files in the "old" format, such as `<iostream.h>`, there is no specification in the standard. Hence, they are not supported. In practice, most vendors will probably provide them to enable backward compatibility. Note that there were more changes in the headers than just the introduction of namespace `std`. In general, you should either use the old names of header files or switch to the new standardized names.