PREV
5. Templates

NEXT
ngs and Regular Expressions

6. Library Overview

Why waste time learning
when ignorance is instantaneous?

– Hobbes

6.1. INTRODUCTION

No significant program is written in just a bare programming language. First, a set of libraries is developed. These then form the basis for further work. Most programs are tedious to write in the bare language, whereas just about any task can be rendered simple by the use of good libraries.

Continuing from Chapters 1- 5 , Chapters 6-13 give a quick tour of key standard-library facilities. I very briefly present useful standard-library types, such as `string`, `ostream`, `vector`, `map`, unique_ptr, thread, regex, and complex, as well as the most common ways of using them. As in Chapters 1-5, you are strongly encouraged not to be distracted or discouraged by an incomplete understanding of details. The purpose of this chapter is to convey a basic understanding of the most useful library facilities.

The specification of the standard library is almost two thirds of the ISO C++ standard. Explore it, and prefer it to home-made alternatives. Much thought has gone into its design, more still into its implementations, and much effort will go into its maintenance and extension.

The standard-library facilities described in this book are part of every complete C++ implementation. In addition to the standard-library components, most implementations offer "graphical user interface" systems (GUIs), Web interfaces, database interfaces, etc. Similarly, most application-development environments provide "foundation libraries" for corporate or industrial "standard" development and/or execution environments. Here, I do not describe such systems and libraries.

The intent is to provide a self-contained description of C++ as defined by the standard and to keep the examples portable. Naturally, a programmer is encouraged to explore the more extensive facilities available on most systems.

6.2. STANDARD-LIBRARY COMPONENTS

The facilities provided by the standard library can be classified like this:

• Run-time language support (e.g., for allocation and run-time type information).

• The C standard library (with very minor modifications to minimize violations of the type system).

• Strings (with support for international character sets and localization); see §7.2.

• Support for regular expression matching; see §7.3.

• I/O streams is an extensible framework for input and output to which users can add their own types, streams, buffering strategies, locales, and character sets.

• A framework of containers (such as vector and map) and algorithms (such as find(), sort(), and merge()); see Chapter 9 and Chapter 10. This framework, conventionally called the STL [Stepanov,1994], is extensible so users can add their own containers and algorithms.

• Support for numerical computation (such as standard mathematical functions, complex numbers, vectors with arithmetic operations, and random number generators); see §4.2.1 and Chapter 12.

• Support for concurrent programming, including threads and locks; see Chapter 13. The concurrency support is foundational so that users can add support for new models of concurrency as libraries.

• Utilities to support template metaprogramming (e.g., type traits; §11.6), STL-style generic programming (e.g., pair; §11.3.3), and general programming (e.g., clock; §11.4).

• "Smart pointers" for resource management (e.g., unique_ptr and shared_ptr; §11.2.1) and an interface to garbage collectors (§4.6.4).

• Special-purpose containers, such as array (§11.3.1), bitset (§11.3.2), and tuple (§11.3.3).

The main criteria for including a class in the library were that:

• it could be helpful to almost every C++ programmer (both novices and experts),

• it could be provided in a general form that did not add significant overhead compared to a simpler version of the same facility, and

• that simple uses should be easy to learn (relative to the inherent complexity of their task).

Essentially, the C++ standard library provides the most common fundamental data structures together with the fundamental algorithms used on them.

6.3. STANDARD-LIBRARY HEADERS AND NAMESPACE

Every standard-library facility is provided through some standard header. For example:

```
#include<string>
#include<list>
```

This makes the standard string and list available.

The standard library is defined in a namespace (§3.3) called std. To use standard library facilities, the std:: prefix can be used:

Click here to view code image

```
std::string s {"Four legs Good; two legs Baaad!"};
std::list<std::string> slogans {"War is Peace", "Freedom is Slavery'
```

For simplicity, I will rarely use the std:: prefix explicitly in examples. Neither will I always #include the necessary headers explicitly. To compile and run the program fragments here, you must #include the appropriate headers and make the names they declare accessible. For example:

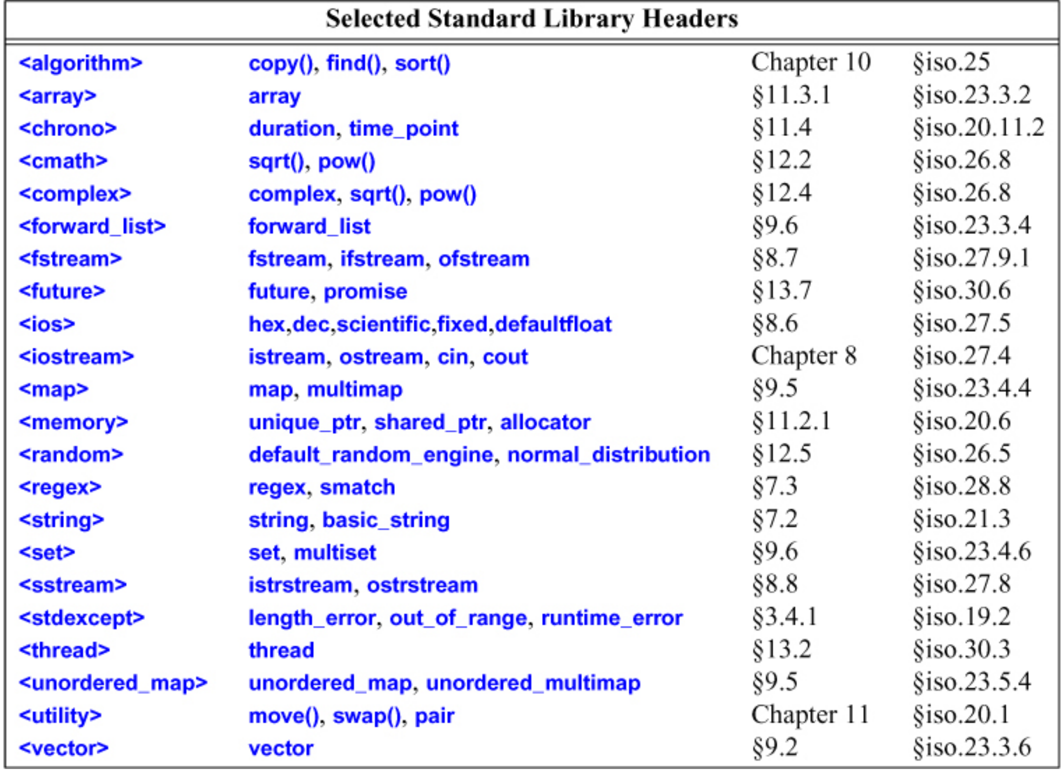
Click here to view code image

```
#include<string>                    // make the standard string faci
using namespace std;                 // make std names available wit

string s {"C++ is a general−purpose programming language"};
```

It is generally in poor taste to dump every name from a namespace into the global namespace. However, in this book, I use the standard library exclusively and it is good to know what it offers.

Here is a selection of standard-library headers, all supplying declarations in namespace std:


Selected Standard Library Headers

<algorithm>	copy(), find(), sort()	Chapter 10	§iso.25
<array>	array	§11.3.1	§iso.23.3.2
<chrono>	duration, time_point	§11.4	§iso.20.11.2
<cmath>	sqrt(), pow()	§12.2	§iso.26.8
<complex>	complex, sqrt(), pow()	§12.4	§iso.26.8
<forward_list>	forward_list	§9.6	§iso.23.3.4
<fstream>	fstream, ifstream, ofstream	§8.7	§iso.27.9.1
<future>	future, promise	§13.7	§iso.30.6
<ios>	hex,dec,scientific,fixed,defaultfloat	§8.6	§iso.27.5
<iostream>	istream, ostream, cin, cout	Chapter 8	§iso.27.4
<map>	map, multimap	§9.5	§iso.23.4.4
<memory>	unique_ptr, shared_ptr, allocator	§11.2.1	§iso.20.6
<random>	default_random_engine, normal_distribution	§12.5	§iso.26.5
<regex>	regex, smatch	§7.3	§iso.28.8
<string>	string, basic_string	§7.2	§iso.21.3
<set>	set, multiset	§9.6	§iso.23.4.6
<sstream>	istrstream, ostrstream	§8.8	§iso.27.8
<stdexcept>	length_error, out_of_range, runtime_error	§3.4.1	§iso.19.2
<thread>	thread	§13.2	§iso.30.3
<unordered_map>	unordered_map, unordered_multimap	§9.5	§iso.23.5.4
<utility>	move(), swap(), pair	Chapter 11	§iso.20.1
<vector>	vector	§9.2	§iso.23.3.6

This listing is far from complete.

Headers from the C standard library, such as <stdlib.h> are provided. For each such header there is also a version with its name prefixed by c and the .h removed. This version, such as <cstdlib> places its declarations in the std namespace.

6.4. ADVICE

[1] The material in this chapter roughly corresponds to what is described in much greater detail in Chapter 30 of [Stroustrup,2013].

[2] Don't reinvent the wheel; use libraries; §6.1.

[3] When you have a choice, prefer the standard library over other libraries; §6.1.

[4] Do not think that the standard library is ideal for everything; §6.1.

[5] Remember to #include the headers for the facilities you use; §6.3.

[6] Remember that standard-library facilities are defined in namespace std; §6.3.

PREV
5. Templates

NEXT
7. Strings and Regular Expressions