`<algorithm>` -- (STL) for defining numerous templates that implement useful algorithms

`<bitset>` -- for defining a template class that administers sets of bits

`<cassert>` -- for enforcing assertions when functions execute

`<cctype>` -- for classifying characters

`<cerrno>` -- for testing error codes reported by library functions

`<cfloat>` -- for testing floating-point type properties

`<ciso646>` -- for programming in ISO 646 variant character sets

`<climits>` -- for testing integer type properties

`<clocale>` -- for adapting to different cultural conventions

`<cmath>` -- for computing common mathematical functions

`<complex>` -- for defining a template class that supports complex arithmetic

`<csetjmp>` -- for executing nonlocal *goto* statements

`<csignal>` -- for controlling various exceptional conditions

`<cstdarg>` -- for accessing a varying number of arguments

`<cstddef>` -- for defining several useful types and macros

`<cstdio>` -- for performing input and output

`<cstdlib>` -- for performing a variety of operations

`<cstring>` -- for manipulating several kinds of strings

`<ctime>` -- for converting between various time and date formats

`<cwchar>` -- for manipulating wide streams and several kinds of strings

`<cwctype>` -- for classifying wide characters

`<deque>` -- (STL) for defining a template class that implements a deque container

`<exception>` -- for defining several functions that control exception handling

`<fstream>` -- for defining several iostreams template classes that manipulate exteral files

`<functional>` -- (STL) for defining several templates that help construct predicates for the templates defined in `<algorithm>` and `<numeric>`

`<iomanip>` -- for declaring several iostreams manipulators that take an argument

`<ios>` -- for defining the template class that serves as the base for many iostreams classes

`<iosfwd>` -- for declaring several iostreams template classes before they are necessarily defined

`<iostream>` -- for declaring the iostreams objects that manipulate the standard streams

`<istream>` -- for defining the template class that performs extractions

`<iterator>` -- (STL) for defining several templates that help define and manipulate iterators

`<limits>` -- for testing numeric type properties

`<list>` -- (STL) for defining a template class that implements a list container

`<locale>` -- for defining several classes and templates that control locale-specific behavior, as in the iostreams classes

`<map>` -- (STL) for defining template classes that implement associative containers

`<memory>` -- (STL) for defining several templates that allocate and free storage for various container classes

`<new>` -- for declaring several functions that allocate and free storage

`<numeric>` -- (STL) for defining several templates that implement useful numeric functions

`<ostream>` -- for defining the template class that performs insertions

`<queue>` -- (STL) for defining a template class that implements a queue container

`<set>` -- (STL) for defining template classes that implement associative containers with unique elements

`<sstream>` -- for defining several iostreams template classes that manipulate string containers

`<stack>` -- (STL) for defining a template class that implements a stack container

`<stdexcept>` -- for defining several classes useful for reporting exceptions

`<streambuf>` -- for defining template classes that buffer iostreams operations

`<string>` -- for defining a template class that implements a string container

`<strstream>` -- for defining several iostreams classes that manipulate in-memory character sequences

`<typeinfo>` -- for defining class `type_info`, the result of the `typeid` operator

`<utility>` -- (STL) for defining several templates of general utility

`<valarray>` -- for defining several classes and template classes that support value-oriented arrays

`<vector>` -- (STL) for defining a template class that implements a vector container

The Standard C++ library also includes the 18 headers from the **Standard C library**, sometimes with small alterations:

`<assert.h>` -- for enforcing assertions when functions execute

`<ctype.h>` -- for classifying characters

`<errno.h>` -- for testing error codes reported by library functions

`<float.h>` -- for testing floating-point type properties

`<iso646.h>` -- for programming in ISO 646 variant character sets

`<limits.h>` -- for testing integer type properties

`<locale.h>` -- for adapting to different cultural conventions

`<math.h>` -- for computing common mathematical functions

`<setjmp.h>` -- for executing nonlocal *goto* statements

`<signal.h>` -- for controlling various exceptional conditions

`<stdarg.h>` -- for accessing a varying number of arguments

`<stddef.h>` -- for defining several useful types and macros

`<stdio.h>` -- for performing input and output

`<stdlib.h>` -- for performing a variety of operations

`<string.h>` -- for manipulating several kinds of strings

`<time.h>` -- for converting between various time and date formats

`<wchar.h>` -- for manipulating wide streams and several kinds of strings

`<wctype.h>` -- for classifying wide characters

Finally, in this implementation, the Standard C++ library also includes four headers for compatibility with traditional C++ libraries:

`<fstream.h>` -- for defining several iostreams template classes that manipulate exteral files

`<iomanip.h>` -- for declaring several iostreams manipulators that take an argument

`<iostream.h>` -- for declaring the iostreams objects that manipulate the standard streams

<new.h> -- for declaring several functions that allocate and free storage

<stl.h> -- for declaring several template classes that aid migration from older versions of the Standard