IAR Systems

STANDARD C++ LIBRARY

The Standard C++ library as revised through 2014 (C++14) consists of the following C++ system headers:

Header	Description
`<algorithm>`	for defining numerous templates that implement useful algorithms
`<array>`	for defining a fixed-size array with a container-like interface
`<atomic>`	for defining classes and objects that manage atomic data operations
`<bitset>`	for defining a template class that administers sets of bits
`<cassert>`	for enforcing assertions when functions execute
`<ccomplex>`	for performing complex arithmetic
`<cctype>`	for classifying characters
`<cerrno>`	for testing error codes reported by library functions
`<cfenv>`	for controlling IEEE-style floating-point arithmetic
`<cfloat>`	for testing floating-point type properties
`<chrono>`	for defining time points and durations
`<cinttypes>`	for converting various integer types
`<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
`<codecvt>`	for defining locale facets that convert between different Unicode encodings
`<complex>`	for defining a template class that supports complex arithmetic
`<condition_variable>`	IAR Systems does not support `<condition_variable>`
`<csetjmp>`	for executing nonlocal goto statements
`<csignal>`	for controlling various exceptional conditions
`<cstdalign>`	for ensuring that `alignas` is defined

<cstdarg>	for accessing a varying number of arguments
<cstdatomic>	for managing atomic data operations
<cstdbool>	for defining a convenient Boolean type name and constants
<cstddef>	for defining several useful types and macros
<cstdint>	for defining various integer types with size constraints
<cstdio>	for performing input and output
<cstdlib>	for performing a variety of operations
<cstdnoreturn>	for defining the function specifier noreturn
<cstring>	for manipulating several kinds of strings
<ctgmath>	for defining generic forms of math functions
<cthreads>	IAR Systems does not support <cthreads>
<ctime>	for converting between various time and date formats
<cuchar>	for manipulating 16-bit and 32-bit UNICODE wide characters
<cwchar>	for manipulating wide streams and several kinds of strings
<cwctype>	for classifying wide characters
<deque>	for defining a template class that implements a deque container
<exception>	for defining several functions that control exception handling
<forward_list>	for defining a template class that implements a small-footprint singly linked list container
<fstream>	for defining several iostreams template classes that manipulate external files
<functional>	for defining several templates that help construct predicates for the templates defined in <algorithm> and <numeric>
<future>	IAR Systems does not support <future>
<hash_map>	for defining template classes that implement hashed associative containers that map keys to values (also includes an STLport-compatible adapter)
<hash_set>	for defining template classes that implement hashed associative containers (also includes an STLport-compatible adapter)
<initializer_list>	for describing a brace-enclosed initializer list

<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>	for defining several templates that help define and manipulate iterators
<limits>	for testing numeric type properties
<list>	for defining a template class that implements a doubly linked list container
<locale>	for defining several classes and templates that control locale-specific behavior, as in the iostreams classes
<map>	for defining template classes that implement associative containers that map keys to values
<memory>	for defining several templates that allocate and free storage for various container classes
<mutex>	IAR Systems does not support <mutex>
<new>	for declaring several functions that allocate and free storage
<numeric>	for defining several templates that implement useful numeric functions
<ostream>	for defining the template class that performs insertions
<queue>	for defining a template class that implements a queue container
<random>	for defining random number generators
<ratio>	for defining templates that implement compile-time rational numbers
<regex>	for defining a regular-expression matcher
<scoped_allocator>	for wrapping a nest of allocators
<set>	for defining template classes that implement associative containers
<shared_mutex>	IAR Systems does not support <shared_mutex>
<slist>	for defining a template class that implements a

	singly linked list container
`<sstream>`	for defining several iostreams template classes that manipulate string containers
`<stack>`	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
`<system_error>`	for wrapping low-level system errors
`<thread>`	IAR Systems does not support `<thread>`
`<tuple>`	for defining an ordered collection of subobjects
`<typeindex>`	for defining classes and functions to index `typeinfo` objects
`<typeinfo>`	for defining class `type_info`, the result of the `typeid` operator
`<type_traits>`	for accessing detailed type information at compile time to support generic programming
`<unordered_map>`	for defining template classes that implement unordered associative containers that map keys to values
`<unordered_set>`	for defining template classes that implement unordered associative containers
`<utility>`	for defining several templates of general utility
`<valarray>`	for defining several classes and template classes that support value-oriented arrays
`<vector>`	for defining a template class that implements a vector container