<functional>

Functions

These functions create objects of *wrapper classes* based on its arguments:

bind C++11	Bind function arguments (function template)
cref C++11	Construct reference_wrapper to const (function template)
mem_fn C++11	Convert member function to function object (function template)
not1	Return negation of unary function object (function template)
not2	Return negation of binary function object (function template)
ref C++11	Construct reference_wrapper (function template)

Classes

Wrapper classes

Wrapper classes are classes that hold an object and have an interface similar to that object, but adding or changing some of its features:

binary_negate	Negate binary function object class (class template)
function C++11	Function wrapper (class template)
reference_wrapper C++11	Reference wrapper (class template)
unary_negate	Negate unary function object class (class template)

Operator classes

Operator classes are classes that define functional objects that call operators:

bit_and C++11	Bitwise AND function object class (class template)
bit_or C++11	Bitwise OR function object class (class template)
bit_xor C++11	Bitwise XOR function object class (class template)
divides	Division function object class (class template)
equal_to	Function object class for equality comparison (class template)
greater	Function object class for greater-than inequality comparison (class template)
greater_equal	Function object class for greater-than-or-equal-to comparison (class template)
less	Function object class for less-than inequality comparison (class template)
less_equal	Function object class for less-than-or-equal-to comparison (class template)
logical_and	Logical AND function object class (class template)
logical_not	Logical NOT function object class (class template)
logical_or	Logical OR function object class (class template)
minus	Subtraction function object class (class template)
modulus	Modulus function object class (class template)
multiplies	Multiplication function object class (class template)
negate	Negative function object class (class template)
not_equal_to	Function object class for non-equality comparison (class template)
plus	Addition function object class (class template)

Other classes

bad_function_call C++11	Exception thrown on bad call (class)
hash C++11	Default hash function object class (class template)
is_bind_expression C++11	Is bind expression (class template)
is_placeholder C++11	Is placeholder (class template)

Namespaces

placeholders C++11	Bind argument placeholders (namespace)

<exception>

Standard exceptions

This header defines the base class for all exceptions thrown by the elements of the standard library: `exception`, along with several types and utilities to assist handling exceptions:

Types:

exception	Standard exception class (class)
bad_exception	Exception thrown by unexpected handler (class)
nested_exception C++11	Nested exception class (class)
exception_ptr C++11	Exception pointer (type)
terminate_handler	Type of terminate handler function (type)
unexpected_handler	Type of unexpected handler function (type)

fx Functions

terminate	Function handling termination on exception (function)
get_terminate C++11	Get terminate handler function (function)
set_terminate	Set terminate handler function (function)
unexpected	Function handling unexpected exceptions (function)
get_unexpected C++11	Get unexpected handler function (function)
set_unexpected	Set unexpected handler function (function)
uncaught_exception	Return exception status (function)
current_exception C++11	Get smart pointer to current exception (function)
rethrow_exception C++11	Rethrow exception (function)
make_exception_ptr C++11	Make exception_ptr (function template)
throw_with_nested C++11	Throw with nested (function)
rethrow_if_nested C++11	Rethrow if nested (function)

std::accumulate <numeric>

sum (1)
```
template <class InputIterator, class T>
   T accumulate (InputIterator first, InputIterator last, T init);
```

custom (2)
```
template <class InputIterator, class T, class BinaryOperation>
   T accumulate (InputIterator first, InputIterator last, T init,
                 BinaryOperation binary_op);
```

std::adjacent_difference <numeric>

difference (1)
```
template <class InputIterator, class OutputIterator>
   OutputIterator adjacent_difference (InputIterator first, InputIterator last,
                                       OutputIterator result);
```

custom (2)
```
template <class InputIterator, class OutputIterator, class BinaryOperation>
   OutputIterator adjacent_difference ( InputIterator first, InputIterator last,
                                        OutputIterator result, BinaryOperation binary_op );
```

std::inner_product <numeric>

sum/multiply (1)
```
template <class InputIterator1, class InputIterator2, class T>
   T inner_product (InputIterator1 first1, InputIterator1 last1,
                    InputIterator2 first2, T init);
```

custom (2)
```
template <class InputIterator1, class InputIterator2, class T,
          class BinaryOperation1, class BinaryOperation2>
   T inner_product (InputIterator1 first1, InputIterator1 last1,
                    InputIterator2 first2, T init,
                    BinaryOperation1 binary_op1,
                    BinaryOperation2 binary_op2);
```

std::partial_sum <numeric>

sum (1)
```
template <class InputIterator, class OutputIterator>
   OutputIterator partial_sum (InputIterator first, InputIterator last,
                               OutputIterator result);
```

custom (2)
```
template <class InputIterator, class OutputIterator, class BinaryOperation>
   OutputIterator partial_sum (InputIterator first, InputIterator last,
                               OutputIterator result, BinaryOperation binary_op);
```

std::iota C++11 <numeric>

```
template <class ForwardIterator, class T>
  void iota (ForwardIterator first, ForwardIterator last, T val);
```

<cctype> (ctype.h)

Character handling functions

This header declares a set of functions to classify and transform individual characters.

fx Functions

These functions take the `int` equivalent of one character as parameter and return an `int` that can either be another character or a value representing a boolean value: an `int` value of `0` means false, and an `int` value different from `0` represents true.

There are two sets of functions:

Character classification functions

They check whether the character passed as parameter belongs to a certain category:

isalnum	Check if character is alphanumeric (function)
isalpha	Check if character is alphabetic (function)
isblank C++11	Check if character is blank (function)
iscntrl	Check if character is a control character (function)
isdigit	Check if character is decimal digit (function)
isgraph	Check if character has graphical representation (function)
islower	Check if character is lowercase letter (function)
isprint	Check if character is printable (function)
ispunct	Check if character is a punctuation character (function)
isspace	Check if character is a white-space (function)
isupper	Check if character is uppercase letter (function)
isxdigit	Check if character is hexadecimal digit (function)

Character conversion functions

Two functions that convert between letter cases:

tolower	Convert uppercase letter to lowercase (function)
toupper	Convert lowercase letter to uppercase (function)

!!!not std!!!