header

**<string>**

Strings

This header introduces string types, character traits and a set of converting functions:

**Class templates**

[**basic\_string**](http://www.cplusplus.com/reference/string/basic_string/)

Generic string class (class template )

[**char\_traits**](http://www.cplusplus.com/reference/string/char_traits/)

Character traits (class template )

**Class instantiations**

[**string**](http://www.cplusplus.com/reference/string/string/)

String class (class )

[**u16string**](http://www.cplusplus.com/reference/string/u16string/)

String of 16-bit characters (class )

[**u32string**](http://www.cplusplus.com/reference/string/u32string/)

String of 32-bit characters (class )

[**wstring**](http://www.cplusplus.com/reference/string/wstring/)

Wide string (class )

**Functions**

**Convert from strings**

[**stoi**](http://www.cplusplus.com/reference/string/stoi/)

Convert string to integer (function template )

[**stol**](http://www.cplusplus.com/reference/string/stol/)

Convert string to long int (function template )

[**stoul**](http://www.cplusplus.com/reference/string/stoul/)

Convert string to unsigned integer (function template )

[**stoll**](http://www.cplusplus.com/reference/string/stoll/)

Convert string to long long (function template )

[**stoull**](http://www.cplusplus.com/reference/string/stoull/)

Convert string to unsigned long long (function template )

[**stof**](http://www.cplusplus.com/reference/string/stof/)

Convert string to float (function template )

[**stod**](http://www.cplusplus.com/reference/string/stod/)

Convert string to double (function template )

[**stold**](http://www.cplusplus.com/reference/string/stold/)

Convert string to long double (function template )

**Convert to strings**

[**to\_string**](http://www.cplusplus.com/reference/string/to_string/)

Convert numerical value to string (function )

[**to\_wstring**](http://www.cplusplus.com/reference/string/to_wstring/)

Convert numerical value to wide string (function )

**Range access**

[**begin**](http://www.cplusplus.com/reference/iterator/begin/)

Iterator to beginning (function template )

[**end**](http://www.cplusplus.com/reference/iterator/end/)

Iterator to end (function template )

function template

<string>

**std::stoi**

int stoi (const string& str, size\_t\* idx = 0, int base = 10);

int stoi (const wstring& str, size\_t\* idx = 0, int base = 10);

Convert string to integer

Parses *str* interpreting its content as an integral number of the specified *base*, which is returned as an int value.  
  
If *idx* is not a null pointer, the function also sets the value of *idx* to the position of the first character in *str* after the number.  
  
The function uses [strtol](http://www.cplusplus.com/strtol) (or [wcstol](http://www.cplusplus.com/wcstol)) to perform the conversion (see [strtol](http://www.cplusplus.com/strtol) for more details on the process).

**Parameters**

str

String object with the representation of an integral number.

idx

Pointer to an object of type [size\_t](http://www.cplusplus.com/size_t), whose value is set by the function to position of the next character in *str* after the numerical value.  
This parameter can also be a null pointer, in which case it is not used.

base

Numerical base (radix) that determines the valid characters and their interpretation.  
If this is 0, the base used is determined by the format in the sequence (see [strtol](http://www.cplusplus.com/strtol) for details). Notice that by default this argument is 10, not 0.

**Return Value**

On success, the function returns the converted integral number as an int value.

**Example**

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 | *// stoi example*  *#include <iostream> // std::cout*  *#include <string> // std::string, std::stoi*  *int* main ()  {  std::string str\_dec = "2001, A Space Odyssey";  std::string str\_hex = "40c3";  std::string str\_bin = "-10010110001";  std::string str\_auto = "0x7f";  std::string::size\_type sz; *// alias of size\_t*  *int* i\_dec = std::stoi (str\_dec,&sz);  *int* i\_hex = std::stoi (str\_hex,*nullptr*,16);  *int* i\_bin = std::stoi (str\_bin,*nullptr*,2);  *int* i\_auto = std::stoi (str\_auto,*nullptr*,0);  std::cout << str\_dec << ": " << i\_dec << " and [" << str\_dec.substr(sz) << "]\n";  std::cout << str\_hex << ": " << i\_hex << '\n';  std::cout << str\_bin << ": " << i\_bin << '\n';  std::cout << str\_auto << ": " << i\_auto << '\n';  *return* 0;  } |

Output:

|  |
| --- |
| 2001, A Space Odyssey: 2001 and [, A Space Odyssey]  40c3: 16579  -10010110001: -1201  0x7f: 127 |

**Complexity**

Unspecified, but generally linear in the number of characters interpreted.

**Data races**

Modifies the value pointed by *idx* (if not zero).

**Exceptions**

If no conversion could be performed, an [invalid\_argument](http://www.cplusplus.com/invalid_argument) exception is thrown.  
If the value read is out of the range of representable values by an int, an [out\_of\_range](http://www.cplusplus.com/out_of_range) exception is thrown.  
  
An invalid *idx* causes *undefined behavior*.

function template

<string>

**std::stol**

long stol (const string& str, size\_t\* idx = 0, int base = 10);

long stol (const wstring& str, size\_t\* idx = 0, int base = 10);

Convert string to long int

Parses *str* interpreting its content as an integral number of the specified *base*, which is returned as a value of type long int.  
  
If *idx* is not a null pointer, the function also sets the value of *idx* to the position of the first character in *str* after the number.  
  
The function uses [strtol](http://www.cplusplus.com/strtol) (or [wcstol](http://www.cplusplus.com/wcstol)) to perform the conversion (see [strtol](http://www.cplusplus.com/strtol) for more details on the process).

**Parameters**

str

String object with the representation of an integral number.

idx

Pointer to an object of type [size\_t](http://www.cplusplus.com/size_t), whose value is set by the function to position of the next character in *str* after the numerical value.  
This parameter can also be a null pointer, in which case it is not used.

base

Numerical base (radix) that determines the valid characters and their interpretation.  
If this is 0, the base used is determined by the format in the sequence (see [strtol](http://www.cplusplus.com/strtol) for details). Notice that by default this argument is 10, not 0.

**Return Value**

On success, the function returns the converted integral number as a value of type long int.

**Example**

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 | *// stol example*  *#include <iostream> // std::cout*  *#include <string> // std::string, std::stol*  *int* main ()  {  std::string str\_dec = "1987520";  std::string str\_hex = "2f04e009";  std::string str\_bin = "-11101001100100111010";  std::string str\_auto = "0x7fffff";  std::string::size\_type sz; *// alias of size\_t*  *long* li\_dec = std::stol (str\_dec,&sz);  *long* li\_hex = std::stol (str\_hex,*nullptr*,16);  *long* li\_bin = std::stol (str\_bin,*nullptr*,2);  *long* li\_auto = std::stol (str\_auto,*nullptr*,0);  std::cout << str\_dec << ": " << li\_dec << '\n';  std::cout << str\_hex << ": " << li\_hex << '\n';  std::cout << str\_bin << ": " << li\_bin << '\n';  std::cout << str\_auto << ": " << li\_auto << '\n';  *return* 0;  } |

Output:

|  |
| --- |
| 1987520: 1987520  2f04e009: 788848649  -11101001100100111010: -956730  0x7fffff: 8388607 |

**Complexity**

Unspecified, but generally linear in the number of characters interpreted.

**Data races**

Modifies the value pointed by *idx* (if not zero).

**Exceptions**

If no conversion could be performed, an [invalid\_argument](http://www.cplusplus.com/invalid_argument) exception is thrown.  
If the value read is out of the range of representable values by a long int, an [out\_of\_range](http://www.cplusplus.com/out_of_range) exception is thrown.  
  
An invalid *idx* causes *undefined behavior*.

function template

<string>

**std::stoul**

unsigned long stoul (const string& str, size\_t\* idx = 0, int base = 10);

unsigned long stoul (const wstring& str, size\_t\* idx = 0, int base = 10);

Convert string to unsigned integer

Parses *str* interpreting its content as an integral number of the specified *base*, which is returned as an unsigned long value.  
  
If *idx* is not a null pointer, the function also sets the value of *idx* to the position of the first character in *str* after the number.  
  
The function uses [strtoul](http://www.cplusplus.com/strtoul) (or [wcstoul](http://www.cplusplus.com/wcstoul)) to perform the conversion (see [strtol](http://www.cplusplus.com/strtol) for more details on the process).

**Parameters**

str

String object with the representation of an integral number.

idx

Pointer to an object of type [size\_t](http://www.cplusplus.com/size_t), whose value is set by the function to position of the next character in *str* after the numerical value.  
This parameter can also be a null pointer, in which case it is not used.

base

Numerical base (radix) that determines the valid characters and their interpretation.  
If this is 0, the base used is determined by the format in the sequence (see [strtol](http://www.cplusplus.com/strtol) for details). Notice that by default this argument is 10, not 0.

**Return Value**

On success, the function returns the converted integral number as an unsigned long value.

**Example**

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 | *// stoul example*  *#include <iostream> // std::cin, std::cout*  *#include <string> // std::string, std::stoul, std::getline*  *int* main ()  {  std::string str;  std::cout << "Enter an unsigned number: ";  std::getline (std::cin,str);  *unsigned* *long* ul = std::stoul (str,*nullptr*,0);  std::cout << "You entered: " << ul << '\n';  *return* 0;  } |

**Complexity**

Unspecified, but generally linear in the number of characters interpreted.

**Data races**

Modifies the value pointed by *idx* (if not zero).

**Exceptions**

If no conversion could be performed, an [invalid\_argument](http://www.cplusplus.com/invalid_argument) exception is thrown.  
If the value read is out of the range of representable values by an unsigned int, an [out\_of\_range](http://www.cplusplus.com/out_of_range) exception is thrown.  
  
An invalid *idx* causes *undefined behavior*.

function template

<string>

**std::stoll**

long long stoll (const string& str, size\_t\* idx = 0, int base = 10);

long long stoll (const wstring& str, size\_t\* idx = 0, int base = 10);

Convert string to long long

Parses *str* interpreting its content as an integral number of the specified *base*, which is returned as a value of type long long.  
  
If *idx* is not a null pointer, the function also sets the value of *idx* to the position of the first character in *str* after the number.  
  
The function uses [strtoll](http://www.cplusplus.com/strtoll) (or [wcstoll](http://www.cplusplus.com/wcstoll)) to perform the conversion (see [strtol](http://www.cplusplus.com/strtol) for more details on the process).

**Parameters**

str

String object with the representation of an integral number.

idx

Pointer to an object of type [size\_t](http://www.cplusplus.com/size_t), whose value is set by the function to position of the next character in *str* after the numerical value.  
This parameter can also be a null pointer, in which case it is not used.

base

Numerical base (radix) that determines the valid characters and their interpretation.  
If this is 0, the base used is determined by the format in the sequence (see [strtol](http://www.cplusplus.com/strtol) for details). Notice that by default this argument is 10, not 0.

**Return Value**

On success, the function returns the converted integral number as a value of type long long.

**Example**

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 | *// stoll example*  *#include <iostream> // std::cout*  *#include <string> // std::string, std::stoll*  *int* main ()  {  std::string str = "8246821 0xffff 020";  std::string::size\_type sz = 0; *// alias of size\_t*  *while* (!str.empty()) {  *long* *long* ll = std::stoll (str,&sz,0);  std::cout << str.substr(0,sz) << " interpreted as " << ll << '\n';  str = str.substr(sz);  }  *return* 0;  } |

Output:

|  |
| --- |
| 8246821 interpreted as 8246821  0xffff interpreted as 65535  020 interpreted as 16 |

**Complexity**

Unspecified, but generally linear in the number of characters interpreted.

**Data races**

Modifies the value pointed by *idx* (if not zero).

**Exceptions**

If no conversion could be performed, an [invalid\_argument](http://www.cplusplus.com/invalid_argument) exception is thrown.  
If the value read is out of the range of representable values by a long long, an [out\_of\_range](http://www.cplusplus.com/out_of_range) exception is thrown.  
  
An invalid *idx* causes *undefined behavior*.

function template

<string>

**std::stoull**

unsigned long long stoull (const string& str, size\_t\* idx = 0, int base = 10);

unsigned long long stoull (const wstring& str, size\_t\* idx = 0, int base = 10);

Convert string to unsigned long long

Parses *str* interpreting its content as an integral number of the specified *base*, which is returned as a value of type unsigned long long.  
  
If *idx* is not a null pointer, the function also sets the value of *idx* to the position of the first character in *str* after the number.  
  
The function uses [strtoull](http://www.cplusplus.com/strtoull) (or [wcstoull](http://www.cplusplus.com/wcstoull)) to perform the conversion (see [strtol](http://www.cplusplus.com/strtol) for more details on the process).

**Parameters**

str

String object with the representation of an integral number.

idx

Pointer to an object of type [size\_t](http://www.cplusplus.com/size_t), whose value is set by the function to position of the next character in *str* after the numerical value.  
This parameter can also be a null pointer, in which case it is not used.

base

Numerical base (radix) that determines the valid characters and their interpretation.  
If this is 0, the base used is determined by the format in the sequence (see [strtol](http://www.cplusplus.com/strtol) for details). Notice that by default this argument is 10, not 0.

**Return Value**

On success, the function returns the converted integral number as a value of type unsigned long long.

**Example**

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 | *// stoull example*  *#include <iostream> // std::cout*  *#include <string> // std::string, std::stoull*  *int* main ()  {  std::string str = "8246821 0xffff 020 -1";  std::string::size\_type sz = 0; *// alias of size\_t*  *while* (!str.empty()) {  *unsigned* *long* *long* ull = std::stoull (str,&sz,0);  std::cout << str.substr(0,sz) << " interpreted as " << ull << '\n';  str = str.substr(sz);  }  *return* 0;  } |

Possible output:

|  |
| --- |
| 8246821 interpreted as 8246821  0xffff interpreted as 65535  020 interpreted as 16  -1 interpreted as 18446744073709551615 |

**Complexity**

Unspecified, but generally linear in the number of characters interpreted.

**Data races**

Modifies the value pointed by *idx* (if not zero).

**Exceptions**

If no conversion could be performed, an [invalid\_argument](http://www.cplusplus.com/invalid_argument) exception is thrown.  
If the value read is out of the range of representable values by a unsigned long long, an [out\_of\_range](http://www.cplusplus.com/out_of_range) exception is thrown.  
  
An invalid *idx* causes *undefined behavior*.

function template

<string>

**std::stof**

float stof (const string& str, size\_t\* idx = 0);

float stof (const wstring& str, size\_t\* idx = 0);

Convert string to float

Parses *str* interpreting its content as a floating-point number, which is returned as a value of type float.  
  
If *idx* is not a null pointer, the function also sets the value of *idx* to the position of the first character in *str* after the number.  
  
The function uses [strtod](http://www.cplusplus.com/strtod) (or [wcstod](http://www.cplusplus.com/wcstod)) to perform the conversion (see [strtod](http://www.cplusplus.com/strtod) for more details on the process).

**Parameters**

str

String object with the representation of a floating-point number.

idx

Pointer to an object of type [size\_t](http://www.cplusplus.com/size_t), whose value is set by the function to position of the next character in *str* after the numerical value.  
This parameter can also be a null pointer, in which case it is not used.

**Return Value**

On success, the function returns the converted floating-point number as a value of type float.

**Example**

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 | *// stof example*  *#include <iostream> // std::cout*  *#include <string> // std::string, std::stof*  *int* main ()  {  std::string orbits ("686.97 365.24");  std::string::size\_type sz; *// alias of size\_t*  *float* mars = std::stof (orbits,&sz);  *float* earth = std::stof (orbits.substr(sz));  std::cout << "One martian year takes " << (mars/earth) << " Earth years.\n";  *return* 0;  } |

Possible output:

|  |
| --- |
| One martian year takes 1.88087 Earth years. |

**Complexity**

Unspecified, but generally linear in the number of characters interpreted.

**Data races**

Modifies the value pointed by *idx* (if not zero).

**Exceptions**

If no conversion could be performed, an [invalid\_argument](http://www.cplusplus.com/invalid_argument) exception is thrown.  
If the value read is out of the range of representable values by a float (in some library implementations, this includes underflows), an [out\_of\_range](http://www.cplusplus.com/out_of_range) exception is thrown.  
  
An invalid *idx* causes *undefined behavior*.

function template

<string>

**std::stod**

double stod (const string& str, size\_t\* idx = 0);

double stod (const wstring& str, size\_t\* idx = 0);

Convert string to double

Parses *str* interpreting its content as a floating-point number, which is returned as a value of type double.  
  
If *idx* is not a null pointer, the function also sets the value of *idx* to the position of the first character in *str* after the number.  
  
The function uses [strtod](http://www.cplusplus.com/strtod) (or [wcstod](http://www.cplusplus.com/wcstod)) to perform the conversion (see [strtod](http://www.cplusplus.com/strtod) for more details on the process).

**Parameters**

str

String object with the representation of a floating-point number.

idx

Pointer to an object of type [size\_t](http://www.cplusplus.com/size_t), whose value is set by the function to position of the next character in *str* after the numerical value.  
This parameter can also be a null pointer, in which case it is not used.

**Return Value**

On success, the function returns the converted floating-point number as a value of type double.

**Example**

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 | *// stod example*  *#include <iostream> // std::cout*  *#include <string> // std::string, std::stod*  *int* main ()  {  std::string orbits ("365.24 29.53");  std::string::size\_type sz; *// alias of size\_t*  *double* earth = std::stod (orbits,&sz);  *double* moon = std::stod (orbits.substr(sz));  std::cout << "The moon completes " << (earth/moon) << " orbits per Earth year.\n";  *return* 0;  } |

Possible output:

|  |
| --- |
| The moon completes 12.3684 orbits per Earth year. |

**Complexity**

Unspecified, but generally linear in the number of characters interpreted.

**Data races**

Modifies the value pointed by *idx* (if not zero).

**Exceptions**

If no conversion could be performed, an [invalid\_argument](http://www.cplusplus.com/invalid_argument) exception is thrown.  
If the value read is out of the range of representable values by a double (in some library implementations, this includes underflows), an [out\_of\_range](http://www.cplusplus.com/out_of_range) exception is thrown.  
  
An invalid *idx* causes *undefined behavior*.

function template

<string>

**std::stold**

long double stold (const string& str, size\_t\* idx = 0);

long double stold (const wstring& str, size\_t\* idx = 0);

Convert string to long double

Parses *str* interpreting its content as a floating-point number, which is returned as a value of type long double.  
  
If *idx* is not a null pointer, the function also sets the value of *idx* to the position of the first character in *str* after the number.  
  
The function uses [strtold](http://www.cplusplus.com/strtold) (or [wcstold](http://www.cplusplus.com/wcstold)) to perform the conversion (see [strtod](http://www.cplusplus.com/strtod) for more details on the process).

**Parameters**

str

String object with the representation of a floating-point number.

idx

Pointer to an object of type [size\_t](http://www.cplusplus.com/size_t), whose value is set by the function to position of the next character in *str* after the numerical value.  
This parameter can also be a null pointer, in which case it is not used.

**Return Value**

On success, the function returns the converted floating-point number as a value of type long double.

**Example**

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 | *// stold example*  *#include <iostream> // std::cout*  *#include <string> // std::string, std::stod*  *int* main ()  {  std::string orbits ("90613.305 365.24");  std::string::size\_type sz; *// alias of size\_t*  *long* *double* pluto = std::stod (orbits,&sz);  *long* *double* earth = std::stod (orbits.substr(sz));  std::cout << "Pluto takes " << (pluto/earth) << " years to complete an orbit.\n";  *return* 0;  } |

Possible output:

|  |
| --- |
| Pluto takes 248.093 years to complete an orbit. |

**Complexity**

Unspecified, but generally linear in the number of characters interpreted.

**Data races**

Modifies the value pointed by *idx* (if not zero).

**Exceptions**

If no conversion could be performed, an [invalid\_argument](http://www.cplusplus.com/invalid_argument) exception is thrown.  
If the value read is out of the range of representable values by a long double (in some library implementations, this includes underflows), an [out\_of\_range](http://www.cplusplus.com/out_of_range) exception is thrown.  
  
An invalid *idx* causes *undefined behavior*.

function

<string>

**std::to\_string**

string to\_string (int val);

string to\_string (long val);

string to\_string (long long val);

string to\_string (unsigned val);

string to\_string (unsigned long val);

string to\_string (unsigned long long val);

string to\_string (float val);

string to\_string (double val);

string to\_string (long double val);

Convert numerical value to string

Returns a [string](http://www.cplusplus.com/string) with the representation of *val*.  
  
The format used is the same that [printf](http://www.cplusplus.com/printf) would print for the corresponding type:

|  |  |  |
| --- | --- | --- |
| **type of *val*** | **printf equivalent** | **description** |
| int | "%d" | Decimal-base representation of *val*. The representations of negative values are preceded with a minus sign (-). |
| long | "%ld |
| long long | "%lld |
| unsigned | "%u" | Decimal-base representation of *val*. |
| unsigned long | "%lu |
| unsigned long long | "%llu |
| float | "%f" | As many digits are written as needed to represent the integral part, followed by the decimal-point character and six decimal digits. inf (or infinity) is used to represent *infinity*. nan (followed by an optional sequence of characters) to represent NaNs (*Not-a-Number*). The representations of negative values are preceded with a minus sign (-). |
| double | "%f |
| long double | "%Lf |

**Parameters**

val

Numerical value.

**Return Value**

A [string](http://www.cplusplus.com/string) object containing the representation of *val* as a sequence of characters.

**Example**

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 | *// to\_string example*  *#include <iostream> // std::cout*  *#include <string> // std::string, std::to\_string*  *int* main ()  {  std::string pi = "pi is " + std::to\_string(3.1415926);  std::string perfect = std::to\_string(1+2+4+7+14) + " is a perfect number";  std::cout << pi << '\n';  std::cout << perfect << '\n';  *return* 0;  } |

Possible output:

|  |
| --- |
| pi is 3.141593  28 is a perfect number |

**Exceptions**

The [string constructor](http://www.cplusplus.com/string::string) may throw.

Top of Form

Bottom of Form

[[hide](javascript:donate.hide())]

function

<string>

**std::to\_wstring**

wstring to\_wstring (int val);

wstring to\_wstring (long val);

wstring to\_wstring (long long val);

wstring to\_wstring (unsigned val);

wstring to\_wstring (unsigned long val);

wstring to\_wstring (unsigned long long val);

wstring to\_wstring (float val);

wstring to\_wstring (double val);

wstring to\_wstring (long double val);

Convert numerical value to wide string

Returns a [wstring](http://www.cplusplus.com/wstring) with the representation of *val*.  
  
The format used is the same that [wprintf](http://www.cplusplus.com/wprintf) would print for the corresponding type:

|  |  |  |
| --- | --- | --- |
| **type of *val*** | **wprintf equivalent** | **description** |
| int | L"%d" | Decimal-base representation of *val*. The representations of negative values are preceded with a minus sign (-). |
| long | L"%ld |
| long long | L"%lld |
| unsigned | "L%u" | Decimal-base representation of *val*. |
| unsigned long | L"%lu |
| unsigned long long | L"%llu |
| float | L"%f" | As many digits are written as needed to represent the integral part, followed by the decimal-point character and six decimal digits. inf (or infinity) is used to represent *infinity*. nan (followed by an optional sequence of characters) to represent NaNs (*Not-a-Number*). The representations of negative values are preceded with a minus sign (-). |
| double | L"%f |
| long double | L"%Lf |

**Parameters**

val

Numerical value.

**Return Value**

A [wstring](http://www.cplusplus.com/wstring) object containing the representation of *val* as a sequence of characters.

**Example**

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 | *// to\_wstring example*  *#include <iostream> // std::wcout*  *#include <string> // std::wstring, std::to\_wstring*  *int* main ()  {  std::wstring pi = L"pi is " + std::to\_wstring(3.1415926);  std::wstring perfect = std::to\_wstring(1+2+4+7+14) + L" is a perfect number";  std::wcout << pi << L'\n';  std::wcout << perfect << L'\n';  *return* 0;  } |

Possible output:

|  |
| --- |
| pi is 3.141593  28 is a perfect number |

**Exceptions**

The [wstring constructor](http://www.cplusplus.com/basic_string::basic_string) may throw.

function template

<iterator> <array> <deque> <forward\_list> <list> <map> <regex> <set> <string> <unordered\_map> <unordered\_set> <vector>

**std::begin**

|  |  |
| --- | --- |
| **container (1)** | template <class Container>  auto begin (Container& cont) -> decltype (cont.begin());  template <class Container>  auto begin (const Container& cont) -> decltype (cont.begin()); |
| **array (2)** | template <class T, size\_t N>  T\* begin (T(&arr)[N]); |

Iterator to beginning

Returns an iterator pointing to the first element in the sequence:

(1) Container

The function returns cont.begin().

(2) Array

The function returns the *array-to-pointer conversion* of its argument.

If the sequence is *empty*, the returned value shall not be *dereferenced*.  
  
These function templates are defined in multiple headers: Each of these headers includes the generic templates for all container and array types and not simply a specific overload. The headers are: [<iterator>](http://www.cplusplus.com/%3Citerator%3E), [<array>](http://www.cplusplus.com/%3Carray%3E), [<deque>](http://www.cplusplus.com/%3Cdeque%3E), [<forward\_list>](http://www.cplusplus.com/%3Cforward_list%3E), [<list>](http://www.cplusplus.com/%3Clist%3E), [map](http://www.cplusplus.com/%3Cmap%3E), [<regex>](http://www.cplusplus.com/%3Cregex%3E), [<set>](http://www.cplusplus.com/%3Cset%3E), [<string>](http://www.cplusplus.com/%3Cstring%3E), [<unordered\_map>](http://www.cplusplus.com/%3Cunordered_map%3E), [<unordered\_set>](http://www.cplusplus.com/%3Cunordered_set%3E) and [<vector>](http://www.cplusplus.com/%3Cvector%3E).  
  
Conversely, [begin](http://www.cplusplus.com/begin) is overloaded (with a different definition) in headers [<initializer\_list>](http://www.cplusplus.com/%3Cinitializer_list%3E) and [<valarray>](http://www.cplusplus.com/%3Cvalarray%3E).

**Parameters**

cont

An object of a class type for which member begin is defined.

arr

An array.

**Return Value**

For *(1)*, the same as returned by cont.begin().  
For *(2)*, a pointer to the first element in the array.

**Example**

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 | *// std::begin / std::end example*  *#include <iostream> // std::cout*  *#include <vector> // std::vector, std::begin, std::end*  *int* main () {  *int* foo[] = {10,20,30,40,50};  std::vector<*int*> bar;  *// iterate foo: inserting into bar*  *for* (*auto* it = std::begin(foo); it!=std::end(foo); ++it)  bar.push\_back(\*it);  *// iterate bar: print contents:*  std::cout << "bar contains:";  *for* (*auto* it = std::begin(bar); it!=std::end(bar); ++it)  std::cout << ' ' << \*it;  std::cout << '\n';  *return* 0;  } |

Output:

|  |
| --- |
| bar contains: 10 20 30 40 50 |

**Data races**

The argument is accessed but not modified.  
None of the elements in the sequence are accessed by the call, but the iterator returned can be used to access or modify them.

**Exception safety**

Provides the same level of guarantees as the operation performed on the argument (for standard containers and arrays this is a no-throw guarantee).

function template

<iterator> <array> <deque> <forward\_list> <list> <map> <regex> <set> <string> <unordered\_map> <unordered\_set> <vector>

**std::end**

|  |  |
| --- | --- |
| **container (1)** | template <class Container>  auto end (Container& cont) -> decltype (cont.end());  template <class Container>  auto end (const Container& cont) -> decltype (cont.end()); |
| **array (2)** | template <class T, size\_t N>  T\* end (T(&arr)[N]); |

Iterator to end

Returns an iterator pointing to the *past-the-end* element in the sequence:

(1) Container

The function returns cont.end().

(2) Array

The function returns arr+N.

If the sequence is *empty*, the returned value compares equal to the one returned by [begin](http://www.cplusplus.com/begin) with the same argument.  
  
These function templates are defined in multiple headers: Each of these headers includes the generic templates for all container and array types and not simply a specific overload. The headers are: [<iterator>](http://www.cplusplus.com/%3Citerator%3E), [<array>](http://www.cplusplus.com/%3Carray%3E), [<deque>](http://www.cplusplus.com/%3Cdeque%3E), [<forward\_list>](http://www.cplusplus.com/%3Cforward_list%3E), [<list>](http://www.cplusplus.com/%3Clist%3E), [map](http://www.cplusplus.com/%3Cmap%3E), [<regex>](http://www.cplusplus.com/%3Cregex%3E), [<set>](http://www.cplusplus.com/%3Cset%3E), [<string>](http://www.cplusplus.com/%3Cstring%3E), [<unordered\_map>](http://www.cplusplus.com/%3Cunordered_map%3E), [<unordered\_set>](http://www.cplusplus.com/%3Cunordered_set%3E) and [<vector>](http://www.cplusplus.com/%3Cvector%3E).  
  
Conversely, [end](http://www.cplusplus.com/end) is overloaded (with a different definition) in headers [<initializer\_list>](http://www.cplusplus.com/%3Cinitializer_list%3E) and [<valarray>](http://www.cplusplus.com/%3Cvalarray%3E).

**Parameters**

cont

An object of a class type for which member end is defined.

arr

An array.

**Return Value**

For *(1)*, the same as returned by cont.end().  
For *(2)*, a pointer to the element that would follow the last element in the array.

**Example**

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 | *// std::begin / std::end example*  *#include <iostream> // std::cout*  *#include <vector> // std::vector, std::begin, std::end*  *int* main () {  *int* foo[] = {10,20,30,40,50};  std::vector<*int*> bar;  *// iterate foo: inserting into bar*  *for* (*auto* it = std::begin(foo); it!=std::end(foo); ++it)  bar.push\_back(\*it);  *// iterate bar: print contents:*  std::cout << "bar contains:";  *for* (*auto* it = std::begin(bar); it!=std::end(bar); ++it)  std::cout << ' ' << \*it;  std::cout << '\n';  *return* 0;  } |

Output:

|  |
| --- |
| bar contains: 10 20 30 40 50 |

**Data races**

The argument is accessed but not modified.  
None of the elements in the sequence are accessed by the call, but the iterator returned can be used to access or modify them.

**Exception safety**

Provides the same level of guarantees as the operation performed on the argument (for standard containers and arrays this is a no-throw guarantee).