C++ Standard Template Library

More info at:

cplusplus.com

cppreference.com

isocpp.org

learncpp.com

by NeptuneLuke

Index

Format output

endl	print a new line but also flushes the stream
	
	`#include <iostream>`
	
	`std::cout << ... << std::endl;`
		<iostream>
		<ostream>

flush	send what is on the output buffer directly to the device connected to thes stream
	
	`#include <iostream>`
	
	`cout << ... << std::flush;`
		<iostream>
		<ostream>

left
internal
right
	set the justification of the text
	
	`#include <iostream>`
	
	`cout << std::left << first_name;`
	`cout << std::internal << first_name;`
	`cout << std::right << first_name;`
		<iostream>
		<ios>

setw(int)	specifies the width of the next input/output field
	
	`#include <iomanip>`
	
	`cout << std::setw(15) << first_name <<`
	`        std::setw(3) << age;`
		<iomanip>

setfill(char)	set the specified character to fill empty spaces in out output for the current line
	
	`#include <iomanip>`
	
	`cout << std::fill('-') << std::setw(15) << first_name;`
		<iomanip>

setprecision(int)	changes the floating point precision of the next input/output field
	
	`#include <iomanip>`
	
	`const double PI = 3.141592653589793239;`
	`cout << std::setprecision(10) << PI;`
		<iomanip>

fixed
scientific
	changes output formatting used for floating point
	
	`#include <iomanip>`
	
	`const double PI = 3.141592653589793239;`
	`cout << std::fixed << PI;`
	`cout << std::scientific << PI;`
		<iostream>
		<ios>

showpoint
noshowpoint
	changes output formatting showing or not the decimal part of floating point numbers
	
	`#include <iostream>`
	
	`const double PI = 3.141592653589793239;`
	`cout << std::showpoint << PI;`
	`cout << std::noshowpoint << PI;`
		<iostream>
		<ios>

showpos
noshowpos
	changes output formatting showing or not the + sign in non negative numbers
	
	`#include <iostream>`
	
	`const double PI = 3.141592653589793239;`
	`cout << std::showpos << PI;`
	`cout << std::noshowpos << PI;`
		<iostream>
		<ios>

dec
oct
hex
 	
bitset
	changes the base of the next input/output field
	
	`#include <iostream>`
	
	`cout << std::dec << 42;`
	`cout << std::oct << 42;`
	`cout << std::hex << 42;`
	
	if we need binary representation we can use
	
	std::bitset<int> var_name {value};
	where <int> is the number of bits
	value is the number/char/string to be represented in binary
	
	`#include <bitset>`
	
	`cout << std::bitset<8>{42};`
		<iostream>
		<ios>
		
		<bitset>

uppercase
nouppercase
	changes output formatting showing or not the uppercase letters, can be useful with hex numbers
	
	`#include <iostream>`
	
	`cout << std::hex << std::uppercase << 2a;`
		<iostream>
		<ios>

showbase
noshowbase
	changes the showing of the base of the next input/output field
	
	`#include <iostream>`
	
	`cout << std::hex << std::showbase <<`
	`        std::uppercase << 2a;`
		<iostream>
		<ios>

boolalpha
noboolalpha
	changes the next input/output field by showing true as 1 and false as 0 or not
	
	`#include <iostream>`
	
	`cout << std::boolalpha << true;`
		<iostream>
		<ios>

cout.setf()
cout.unsetf()
	sets or unsets the format of the output by the values passed to
	
	`#include <iostream>`
	
	`const double PI = 3.141592653589793239;`
	`cout << std::fixed << PI;`
	`cout.unsetf(std::fixed);`
	
	you can set or unset more than one at a time
	
	`#include <<iostream>>`
	
	`const double PI = 3.141592653589793239;`
	`cout.setf(std::fixed | std::scientific);`
	`cout << PI;`
	`cout.unsetf(std::fixed | std::scientific);`
		<iostream>
		<ios>

Char

isalnum(int)
isalnum(unsigned char)
	checks if a int/char is alphanumeric	<cctype>
		<ctype.h>

isalpha(int)
isalpha(unsigned char)
	checks if a int/char is a letter of the alphabet	<cctype>
		<ctype.h>

isdigit(int)
 	
isxdigit(unsigned char)
	checks if a int/char is a decimal digit
	
	checks if a int/char is an hexadecimal digit
		<cctype>
		<ctype.h>

isspace(int)
 	
isblank(unsigned char)
	checks if a int/char is a whitespace type character
	
	checks if a int/char is a blank character
		<cctype>
		<ctype.h>

islower(int)
 	
isupper(unsigned char)
	checks if a int/char is a lower/uppercase character	<cctype>
		<ctype.h>

tolower(int)
 	
toupper(unsigned char)
	converts the int/char character to the lowercase/uppercase version	<cctype>
		<ctype.h>

Strings

length()
size()
 	
empty()
	returns the length of the string
	
	checks if the string is empty
		<string>

begin()
end()
	returns an iterator pointing to the first/last character of the string	<string>

push_back(char)
push_back(string)
	appends the char/string at the end of string	<string>

pop_back(char)	removes the last character of the string	<string>

insert(pos, char)
insert(pos, string)
	insert a char/string at the position pos	<string>

erase(pos, len)
 	
erase(p1, p2)
	erase a string of len length starting at the position pos
	
	erase the characters from p1 to p2
		<string>

clear()	clears the entire string	<string>
find(string)	returns the position of the first char of the substring if present, or if not present, a value >= of the length of the string	<string>
substr(pos, len)	returns the substring of length len found starting from position pos	<string>
find_first_of(char) find_first_of (string) find_last_of(char) find_last_of (string)	returns the position of the first/last occurence of the char/string, if present	<string>

find_first_of (, pos) find_last_of (, pos)	returns the position of the first/last occurence of the char/string starting from pos, if present	
compare(string)	returns a negative value if the string appears before the string in lexicographical order returns 0 if the strings are equal returns a positive value if the string appears after the string in lexicographical order	<string>

starts_with(char)
starts_with(string)
ends_with(char)
ends_with(string)
	returns true if the string begins/ends with the char/string	<string>

contains(char)
contains(string)
	checks if the string contains char/string	<string>

reverse(p1, p2)	reverse the string from p1 to p2	<string> <algorithm>
sort(p1, p2)	sorts the string from p1 to p2	<string> <algorithm>
to_string(num)	converts num to its string representation num can be any int, float, double, long	<string>

stoi()
stol()
stof()
stod()
	converts the string to an int, long, float, double	<string>

transform(p1, p2, p3, ::toupper) transform(p1, p2,	converts the string to uppercase/lowercase from p1 to p2 and stores the changes at p3 (where p3	<string> <algorithm> <cctype>

p3, ::tolower)	should ideally be string.begin() or another_string.begin())	

Vector

size()
 	
empty()
	returns the length of the vector
	
	checks if the vector is empty
		<vector>

begin()
end()
	returns an iterator pointing to the first/last element of the vector	<vector>

push_back(elem)	appends the element at the end of vector	<vector>
pop_back()	removes the last element of the vector	<vector>
insert(pos, elem)	insert the element at the position pos	<vector>

erase(pos)
erase(p1, p2)
	erase the element at the position pos
	erase the element between p1 and p2
		<vector>

clear()	clears the entire vector	<vector>
count(p1, p2, elem)	counts how many times elem is found between p1 and p2	<vector> <algorithm>
reverse(p1, p2)	reverse the vector from p1 to p2	<vector> <algorithm>
sort(p1, p2)	sorts the vector from p1 to p2	<vector> <algorithm>

Stack

`size()`
 	
`empty()`
	returns the length of the stack
	
	checks if the stack is empty
		`<stack>`

`push(elem)`	inserts the element at the top	`<stack>`
`pop()`	removes the top element	`<stack>`
`top()`	returns the top element without removing it	`<stack>`

Queue

`size()`
 	
`empty()`
	returns the length of the queue
	
	checks if the queue is empty
		`<queue>`

`push(elem)`	inserts the element at the end of the queue	`<queue>`
`pop()`	removes the first element of the queue	`<queue>`
`front()`	returns the first element without removing it	`<queue>`
`back()`	returns the last element without removing it	`<queue>`

Math

Basically all functions in **`<cmath>`** / **`<math.h>`** are written with float as the values passed as arguments and as return types.

floor(value)
ceil(value)
 	
round(value)
	rounds the value to floor/ceil
	
	rounds the value to the nearest integer
		<cmath>
		<math.h>

abs(value)	returns the absolute value of value	<cmath> <math.h>

exp(value)
pow(value, exp)
	returns e^value
	return value^exp
		<cmath>
		<math.h>

log(value)
log10(value)
lo2(value)
	returns natural log of value
	returns log base 10 of value
	returns log base 2 of value
		<cmath>
		<math.h>

sqrt(value)
cbrt(value)
	returns the square root of value
	returns the cubic root of value
		<cmath>
		<math.h>

sin(value)
asin(value)
	returns the sin of value
	return the arcsin of value
		<cmath>
		<math.h>

cos(value)
acos(value)
	returns the cos of value
	returns the arccos of value
		<cmath>
		<math.h>

tan(value)
atan(value)
	returns the tan of value
	returns the arctan of value
		<cmath>
		<math.h>