

stringstream **802**
 see string stream
string termination character
 internationalized **857**
strlen() **855**
strstream **807**
<strstream> **807**
strstreambuf **807**
struct tm **158**
student_t_distribution **917, 924**
sub_match
 for regular expressions **720**
subscript operator
 see **[]**
substr()
 for strings **676, 711**
subtract_with_carry_engine **915**
suffix()
 for match results **720, 722**
sungetc()
 for input buffers **827, 839**
swap() **136**
 for shared_ptrs **93, 97**
 for arrays **263, 265**
 for container adapters **649**
 for containers **250, 255, 258, 407**
 for dequees **287**
 for forward lists **303**
 for lists **293**
 for maps and multimaps **336**
 for packaged tasks **977**
 for pairs **61**
 for promises **977**
 for sets and multisets **321**
 for strings **674, 702**
 for tuples **71**
 for unique_ptrs **111**
 for unordered containers **368**
 for vectors **271, 274**
swapping
 iterator values **446**
 values **136**
swapping elements **566**
swap_ranges()
 algorithm **566**

symbol
 monetary pattern **878**
sync()
 for output buffers **839**
 synchronization of threads **982**
sync_with_stdio()
 for streams **845**
system
 error category **50**
system_category() **50**
system_clock **149**
 duration **149**
 from_time_t() **158**
 is_steady **149**
 now() **149, 152**
 period **149**
 rep **149**
 time_point **149**
 to_time_t() **152, 158**
system_error **41, 43, 45**
 code() **48**
 error conditions **45**
<system_error> **44**

T

table()
 for ctype facets **895**
table_size
 for ctype facets **895**
tags
 for iterators **466**
tan()
 for complex **935, 940**
 global function **941**
tanh()
 for complex **935, 940**
 global function **941**
tellg()
 for streams **799**
tellp()
 for streams **799**
template
 >> **13**
 alias **27, 1024**

constructor **36**
copy constructor **62**
default parameter **33**
function **27**
member templates **34**
nested class **37**
nontype parameters **33**
typedef **27, 1024**
typename **34**
variadic **26, 68**
zero initialization **37**

tera ratio unit **142**
terminate() **162**
text_file_busy **47**
this_thread **981**
 get_id() **981**
 sleep_for() **947, 981**
 sleep_until() **981**
 yield() **955, 981**
thousands_sep()
 for moneypunct facets **874**
 for numpunct facets **870**

thread **964, 979**
 constructor **979**
 destructor **979**
 detach() **979**
 detached **967**
 get_id() **967, 979**
 hardwareConcurrency() **980**
 ID **967, 979**
 join() **979**
 joinable() **979**
 native_handle() **979**

<thread> **967, 981**
threads **945, 979**
 see concurrency, thread
 synchronization **982**

tie()
 for pairs **67**
 for streams **819**
 for tuples **70, 71, 72**

time
 conversion to/from time_point **158**

time() **158**
time locale category **884**

time_base **889**
dateorder **889**
dmy **889**
mdy **889**
no_order **889**
ydm **889**
ymd **889**

timed_mutex **994, 998**
timed_out **47**

time_get facet **887**
 date_order() **888**
 get() **888**
 get_date() **888**
 get_monthname() **888**
 get_weekday() **888**
 get_year() **888**

<time.h> **157**

timeout future status **954**

time_point **143, 152**
 +, - **155**
 +=, -= **155**
 ==, != **155**
 <, <=, >, >= **155**
 constructor **155**
 conversion to/from calendar time **158**
 current time **152**
 epoch **152**
 for clocks **149**
 max() **155**
 min() **155**
 time_point_cast **155**
 time_since_epoch() **155**

time_point_cast
 for timepoints **155**

time_put facet **884**
 put() **884**

timer **160, 947, 981**
 for locks **994**

time_since_epoch()
 for timepoints **155**

time_t **158, 886**

tinyness_before
 for numeric limits **118**

tm structure **158, 886**

to_bytes() for wstring_convert<> **901**

to_char_type()
 for char_traits **854**
to_int_type()
 for char_traits **854**
token iterator
 for regular expressions **727**
tolower() **684, 896**
 for ctype facets **891**
too_many_files_open **47**
too_many_files_open_in_system **47**
too_many_links **47**
too_many_symbolic_link_levels **47**
top()
 for container adapters **648**
to_string() **652**
 for strings **682, 713**
to_time_t() **757**
 for clocks **153**
 for system_clock **152, 158**
to_ullong() **652**
toupper() **684, 833, 896**
 for ctype facets **891**
to_wstring()
 for strings **682, 713**
TR1 **7**
 namespace **39**
traits
 for characters **689, 853**
 for iterators **466**
 for types **122**
 see type traits
traits_type
 for strings **693**
transaction safety **248**
transform()
 algorithm **225, 240, 563, 564, 684**
 for collate facets **904**
 versus for_each() **509**
transitive **315**
traps
 for numeric limits **118**
truename()
 for numpunct facets **870**
true_type **125, 142**
trunc stream flag **796**

try_lock
 for unique_locks **1000**
try_lock()
 for unique_locks **1000**
 for mutexes **994, 998**
 multiple locks **995**
 spurious failures **994**
try_lock_for() **994**
 for unique_locks **1000**
 for mutexes **160, 998**
try_lock_until() **994**
 for unique_locks **1000**
 for mutexes **160, 998**
try_to_lock **996**
tuple **806**
 and arrays **268**
 and initializer lists **72**
tuple **68**
= **71**
==, != **71**
<, <=, >, >= **71**
 and pair **75**
 constructor **69, 71**
 destructor **71**
 get() **74**
 ignore **72**
 I/O **74**
 make_tuple() **69, 70, 71**
 output **74**
 swap() **71**
 tie() **70, 71, 72**
 tuple_cat() **73**
 tuple_element **73**
 tuple_size **73**
<tuple> **66, 68**
tuple_cat()
 for tuples **73**
tuple_element
 for pairs **62**
 for tuples **73**
tuple interface
 for arrays **268**
tuple_size
 for pairs **62**
 for tuples **73**

type
 auto **14**
 deduction **14**
 of lambdas **31**
 relation traits **128**

type
 for integral_constant **125**
 for ratios **140**

typedef
 for templates **27, 1024**

typeid **42**

<typeinfo> **42**

typename **34, 380**

type traits **122**
 common_type **124**
 predicates **125**
 type modifiers **129**
 type relations **128**

<type_traits> **122, 125**

U

u16string **655, 664**
 see string

u32string **655, 664**
 see string

UCS-2 UCS-4 **851**

uflow()
 for input buffers **839**

unary_function **497**

unary predicate **226**

underflow()
 for input buffers **839**

underflow_error **41, 43**

underlying_type trait **131**

unexpected() **42**

unget()
 for input streams **770**

uniform initialization **15**

uniform_int_distribution **908, 917, 921, 947**

uniform_real_distribution **908, 917, 921**

uninitialized_copy() **1027, 1028**

uninitialized_copy_n() **1027**

uninitialized_fill() **1027**

uninitialized_fill_n() **1027, 1028**

union set **616**

unique()
 algorithm **578**
 for shared_ptrs **94**
 for forward lists **310, 312**
 for lists **297, 298, 421**

unique_copy()
 algorithm **580**

unique_future **975**

unique_lock **996, 1000**
 for condition variables **1004**

unique pointer
 see unique_ptr

unique_ptr **98, 822**
 * **111**
 -> **111**
 = **102, 111**
 ==, != **111**
 <, <=, >, >= **111**
 [] **111**
 and arrays **105**
 as member **103**
 bool() **100, 111**
 comparisons **112**
 constructor **111**
 deleter **107**
 destructor **111**
 get() **111**
 get_deleter() **111**
 initialization **100**
 ownership transfer **101**
 performance **114**
 release() **100, 111**
 reset() **111**
 swap() **111**

unitbuf **846**

unitbuf manipulator **789**

unitbuf stream flag **789**

unlock()
 for unique_locks **1000**
 for mutexes **989, 998**

unordered collection 167
 unordered container **167**, 180
 see container
 begin() for buckets **374**
 bucket() **374**
 bucket_count() **374**
 bucket interface **374**, 380, 429
 bucket_size() **374**
 end() for buckets **374**
 equivalence criterion **377**
 hash function **377**
 modifying access **221**
 order of duplicates 183
 terminology 168
 user-defined inserter **471**
unordered_map *183, 185, 355*
 see unordered container
 = **368**
 ==, != **367**
 [] **186, 374, 408**
 as associative array **185, 374**
 at() **186, 374, 408**
 begin() **369, 429**
 begin() for buckets **374**
 bucket() **429**
 bucket_count() **362, 380, 429**
 bucket interface **374**
 bucket_size() **429**
 cbegin() **369, 429**
 cend() **369, 430**
 clear() **370, 371**
 compare function **366**
 const_local_iterator **399**
 constructor **360**
 count() **368**
 crbegin() **369**
 crend() **369**
 destructor **360**
 element access with bind() **494**
 emplace() **370, 371**
 emplace_hint() **371**
 empty() **367**
 end() **369, 430**
 end() for buckets **374**
 equal_range() **368**
 equivalence criterion **357, 366, 377**
 erase() **370, 371**
 exception handling **375**
 find() **368, 373**
 hasher **399**
 hash function **363, 377**
 hash_function() **362, 427**
 header file **356**
 insert() **370, 371, 372, 382**
 iterators **368**
 key_eq() **362, 427**
 key_equal **399**
 lambda as equivalence criterion **379**
 lambda as hash function **379**
 load_factor() **362, 380, 427**
 local_iterator **399**
 max_bucket_count() **362, 429**
 max_load_factor() **362, 380, 383, 427, 429**
 max_size() **367**
 modifying access **221**
 piecewise construction **373**
 policy **359**
 rbegin() **369**
 rehash() **362, 428**
 removing elements **342**
 rend() **369**
 reserve() **362**
 size() **367, 380**
 swap() **368**
 value_type **356**
<unordered_map> **356**
unordered_multimap *355, 383*
 see unordered container
 = **368**
 ==, != **367**
 begin() **369, 429**
 begin() for buckets **374**
 bucket() **429**
 bucket_count() **362, 380, 429**
 bucket interface **374**
 bucket_size() **429**
 cbegin() **369, 429**
 cend() **369, 430**
 clear() **370, 371**

compare function 366
const_local_iterator 399
constructor 360, 383
count() 368
crbegin() 369
crend() 369
destructor 360
element access with bind() 494
emplace() 370, 371
emplace_hint() 371
empty() 367
end() 369, 430
end() for buckets 374
equal_range() 368
equivalence criterion 357, 366, 377
erase() 370, 371
exception handling 375
find() 368, 373
hasher 399
hash function 363, 377
hash_function() 362, 427
header file 356
insert() 370, 371, 372, 382
iterators 368, 383
key_eq() 362, 427
key_equal 399
lambda as equivalence criterion 379
lambda as hash function 379
load_factor() 362, 380, 427
local_iterator 399
max_bucket_count() 362, 429
max_load_factor() 362, 380, 383,
 427, 429
max_size() 367
modifying access 221
order of duplicates 183
piecewise construction 373
policy 359
rbegin() 369
rehash() 362, 428
removing elements 342
rend() 369
reserve() 362
size() 367, 380
stable order 183
swap() 368
value_type 356
unordered_multiset 182, 196, 355, 377
see unordered container
= 368
==, != 367
begin() 369, 429
begin() for buckets 374
bucket() 429
bucket_count() 362, 380, 429
bucket interface 374
bucket_size() 429
cbegin() 369, 429
cend() 369, 430
clear() 370, 371
compare function 366
const_local_iterator 399
constructor 360, 377
count() 368
crbegin() 369
crend() 369
destructor 360
emplace() 370, 371
emplace_hint() 371
empty() 367
end() 369, 430
end() for buckets 374
equal_range() 368
equivalence criterion 357, 366, 377
erase() 370, 371, 377
exception handling 375
find() 368, 373, 377
hasher 399
hash function 363, 377
hash_function() 362, 427
header file 356
insert() 370, 371, 372, 377, 382
iterators 368, 377
key_eq() 362, 427
key_equal 399
lambda as equivalence criterion 379
lambda as hash function 379
load_factor() 362, 380, 427
local_iterator 399
max_bucket_count() 362, 429

unordered_multiset (continued)

 max_load_factor() **362, 380, 383, 427, 429**
 max_size() **367**
 modifying access **221**
 order of duplicates **183**
 policy **359**
 rbegin() **369**
 rehash() **362, 428**
 rend() **369**
 reserve() **362**
 size() **367, 380**
 stable order **183**
 swap() **368**
 value_type **356**
unordered_set **198, 355, 375**
 see *unordered container*
 = **368**
 ==, != **367**
 begin() **369, 429**
 begin() for buckets **374**
 bucket() **429**
 bucket_count() **362, 380, 429**
 bucket interface **374**
 bucket_size() **429**
 cbegin() **369, 429**
 cend() **369, 430**
 clear() **370, 371**
 compare function **366**
 const_local_iterator **399**
 constructor **360, 375**
 count() **368**
 crbegin() **369**
 crend() **369**
 destructor **360**
 emplace() **370, 371**
 emplace_hint() **371**
 empty() **367**
 end() **369, 430**
 end() for buckets **374**
 equal_range() **368**
 equivalence criterion **357, 366, 377**
 erase() **370, 371, 375**
 exception handling **375**
 find() **368, 373, 375**

hasher **399**
 hash function **363, 377**
 hash_function() **362, 427**
 header file **356**
 insert() **370, 371, 372, 375, 382**
 iterators **368, 375**
 key_eq() **362, 427**
 key_equal **399**
 lambda as equivalence criterion **379**
 lambda as hash function **379**
 load_factor() **362, 380, 427**
 local_iterator **399**
 max_bucket_count() **362, 429**
 max_load_factor() **362, 380, 383, 427, 429**
 max_size() **367**
 modifying access **221**
 policy **359**
 rbegin() **369**
 rehash() **362, 428**
 rend() **369**
 reserve() **362**
 size() **367, 380**
 swap() **368**
 value_type **356**
<unordered_set> **356**
 unsetf() **688**
 for streams **779**
 unshift()
 for codecvt facets **898**
 unsynchronized data access **984**
 upper
 for ctype_base **894**
 upper_bound()
 algorithm **611**
 for containers **405**
 for maps and multimaps **335**
 for sets and multisets **319**
 uppercase manipulator **784**
 uppercase stream flag **784**
 uppercase string characters **684**
 US-ASCII **851**
 use_count()
 for shared_ptrs **94, 97**
 for weak_ptrs **89**

use_facet() 864, 867
user-defined
 <<, >> **810**
 algorithm 308, 468
 allocator 1024
 container **385**
 exception 635
 function object **495**
 inserter **471**
 iterator **471**
 manipulators **777**
 sorting criterion **228, 476**
 stream buffers 832
uses_allocator trait **128**
using declaration 40
using directive 40
UTF-8 UTF-16 UTF-32 **851**
 reading and writing **901, 903**
utilities **59**
<utility> 20, 60, 136, 138

V

valarray **943**
valarray **943**
valid()
 for futures 975
 for packaged tasks 977
valid range **203, 205**
value
 for integral_constant **125**
 monetary pattern 878
value()
 for exceptions **49**
value_comp()
 for associative containers **427**
 for maps and multimaps **335**
 for sets and multisets **318**
value_compare
 for associative containers **399**
value initialization **15**
value pair **60**
value semantics
 for containers **245**
value_too_large 47

value_type
 for allocators 1026
 for complex **935**
 for container adapters **645**
 for containers 260, **397**
 for insert() **341, 372**
 for integral_constant **125**
 for iterator_traits 467
 for maps **345**
 for maps and multimaps 331
 for strings **693**
 for unordered containers 356
variadic template **26, 68**
vector **169, 270, 279**
 see container
++, -- for iterators **440**
= **274**
==, != **273**
<, <=, >, >= **273**
[] **274**
as C-style array **278**
assign() **274**
at() **274**
back() **274**
begin() **276**
capacity() **270, 273, 427**
cbegin() **276**
cend() **276**
clear() **277**
constructor **272, 273, 1027**
contiguity of elements **278**
crbegin() **276**
crend() **276**
data() **278**
destructor **272, 273**
element access **274**
emplace() **277**
emplace_back() **277**
empty() **273**
end() **276**
erase() **277**
exception handling **278**
for bool **281**
front() **274**
header file **270**

vector (continued)
 insert() 277
 iterators 275
 max_size() 273
 pop_back() 277
 push_back() 277
 rbegin() 276
 reallocation 270
 removing elements 276
 rend() 276
 reserve() 271, 273, 1028
 resize() 277
 shrink capacity 271
 shrink_to_fit() 271, 273
 size() 270, 273
 swap() 271, 274
<vector> 270
vector<bool> 281
 and concurrency 985
 const_reference 282
 flip() 281, 282
 reference 282
 versions of C++ 7
void*
 I/O 756
volatile
 and concurrency 988, 998

W

wait()
 for condition variables 1004, 1009
 for futures 953, 975
wait_for() 160
 for condition variables 1009
 for futures 953, 975
wait_until() 160
 for condition variables 1009
 for futures 953, 975
wcerr 751
wchar_t 852, 858
 input 755
 numeric limits 116
wcin 751

wclog 751
wcout 751
wcregex_iterator 726
wcregex_token_iterator 727
 weak pointer
 see **weak_ptr**
weak_ptr 84, 96
 bad_weak_ptr 89
 lock() 88
 use_count() 89
weibull_distribution 917, 922
wfilebuf 791
wfstream 791
what()
 for exceptions 45, 52
 whitespace
 compressing 582
 wide-character format 850
widen()
 for ctype facets 891
 for streams 790
width()
 for streams 782, 811
wifstream 791
wios 750
wiostream 751
wistream 751
wistringstream 802
wofstream 791
wostream 751
wostream 802
wregex 719
write()
 for output streams 771
 global function 835, 837
 writing
 see output
wrong_protocol_type 47
 ws manipulator 746, 774
wsregex_iterator 726
wsregex_token_iterator 727
wstreambuf 750, 832
 see input buffer, output buffer
wstreampos 799

wstring **655**, 664

 see string

wstringbuf **802**

wstring_convert<> **901**

wstringstream **802**

X

xalloc()

 for streams **815**

xdigit

 for ctype_base **894**

xsgetn()

 for input buffers **840**

xsputn()

 for output buffers **832**

Y

ydm date order **889**

yield()

 for this_thread **955**, **981**

ymd date order **889**

yocto ratio unit **142**

yotta ratio unit **142**

Z

zepto ratio unit **142**

zero()

 for durations **147**

zero initialization **37**

zetta ratio unit **142**

最畅销的 C++ 资源

针对C++11更新

C++标准库提供了一组公共类和接口，极大地拓展了C++语言核心功能。然而，C++标准库并非不言自明。为了充分利用其组件，受益于其强大威力，真正能够满足你学习需要的材料绝不能仅仅列出那些类及其函数。

《C++标准库（第2版）》对现已并入新版ANSI/ISO C++语言标准（C++11）的标准库做了描述。本书对于每个标准库组件都提供了综合的文档，包括：关于其设计目的和设计方法的概览；对于复杂概念的清晰解读；为了有效利用所需的实用编程细节；存在的一些陷阱；实际运用中最重要的类和函数的确切签名和定义；以及大量实用代码范例。本书尤其着眼于标准模板库（STL），介绍了容器、迭代器、函数对象以及STL算法。

本书涵盖了所有全新的C++11标准库组件，包括：

- 并发
- 分数运算
- 时钟和定时器
- Tuple
- 新STL容器
- 新STL算法
- 新智能指针
- newlocale facet
- 随机数和分布
- Type trait和实用工具
- 正则表达式

本书还介绍了标准库中的新式C++编程风格及其影响。其中包括lambda表达式、基于区间的for循环、move语义以及可变参数模板。

Nicolai M. Josuttis是一名独立技术顾问，为电信、交通、金融和制造业设计过大中型软件系统。他还是C++标准委员会标准库工作组早期成员，因其权威著作而享有盛名。除了最为畅销的《C++标准库》（第1版出版于1999年），其著作还包括C++ Templates: The Complete Guide（与David Vandevoorde合著，由Addison-Wesley于2003年出版），以及SOA in Practice: The Art of Distributed System Design（由O'Reilly Media于2007年出版，简体中文版《SOA实践指南——分布式系统设计的艺术》由电子工业出版社于2008年出版）。



本书配套网站www.cppstdlib.com上面提供了所有书中代码的下载，也可访问博文视点网站下载（<http://www.broadview.com.cn/26089>）。

上架建议：编程语言>C++

PEARSON

www.pearson.com



策划编辑：张春雨

责任编辑：白 涛

封面设计：李 玲



ISBN 978-7-121-26089-6



9 787121 260896 >

定价：186.00元