

```

int *volatile vip;    // vip 是一个 volatile 指针，它指向 int
volatile int *ivp;   // ivp 是一个指针，它指向一个 volatile int
// vivp 是一个 volatile 指针，它指向一个 volatile int
volatile int *volatile vivp;

int *ip = &v;          // 错误：必须使用指向 volatile 的指针
ivp = &v;              // 正确：ivp 是一个指向 volatile 的指针
vivp = &v;             // 正确：vivp 是一个指向 volatile 的 volatile 指针

```

和 `const` 一样，我们只能将一个 `volatile` 对象的地址（或者拷贝一个指向 `volatile` 类型的指针）赋给一个指向 `volatile` 的指针。同时，只有当某个引用是 `volatile` 的时，我们才能使用一个 `volatile` 对象初始化该引用。

合成的拷贝对 `volatile` 对象无效

`const` 和 `volatile` 的一个重要区别是我们不能使用合成的拷贝/移动构造函数及赋值运算符初始化 `volatile` 对象或从 `volatile` 对象赋值。合成的成员接受的形参类型是（非 `volatile`）常量引用，显然我们不能把一个非 `volatile` 引用绑定到一个 `volatile` 对象上。

如果一个类希望拷贝、移动或赋值它的 `volatile` 对象，则该类必须自定义拷贝或移动操作。例如，我们可以将形参类型指定为 `const volatile` 引用，这样我们就能利用任意类型的 `Foo` 进行拷贝或赋值操作了：

```

class Foo {
public:
    Foo(const volatile Foo&); // 从一个 volatile 对象进行拷贝
    // 将一个 volatile 对象赋值给一个非 volatile 对象
    Foo& operator=(volatile const Foo&);
    // 将一个 volatile 对象赋值给一个 volatile 对象
    Foo& operator=(volatile const Foo&) volatile;
    // Foo 类的剩余部分
};

```

尽管我们可以为 `volatile` 对象定义拷贝和赋值操作，但是一个更深层次的问题是拷贝 `volatile` 对象是否有意义呢？不同程序使用 `volatile` 的目的各不相同，对上述问题的回答与具体的使用目的密切相关。

19.8.3 链接指示：`extern "C"`

C++程序有时需要调用其他语言编写的函数，最常见的是调用 C 语言编写的函数。像所有其他名字一样，其他语言中的函数名字也必须在 C++中进行声明，并且该声明必须指定返回类型和形参列表。对于其他语言编写的函数来说，编译器检查其调用的方式与处理普通 C++函数的方式相同，但是生成的代码有所区别。C++使用链接指示（linkage directive）指出任意非 C++函数所用的语言。



要想把 C++代码和其他语言（包括 C 语言）编写的代码放在一起使用，要求我们必须有权访问该语言的编译器，并且这个编译器与当前的 C++编译器是兼容的。

声明一个非 C++ 的函数

链接指示可以有两种形式：单个的或复合的。链接指示不能出现在类定义或函数定义的内部。同样的链接指示必须在函数的每个声明中都出现。

举个例子，接下来的声明显示了 `cstring` 头文件的某些 C 函数是如何声明的：

```
// 可能出现在 C++头文件<cstring>中的链接指示
// 单语句链接指示
extern "C" size_t strlen(const char *);
// 复合语句链接指示
extern "C" {
    int strcmp(const char*, const char*);
    char *strcat(char*, const char*);
}
```

链接指示的第一种形式包含一个关键字 `extern`，后面是一个字符串字面值常量以及一个“普通的”函数声明。

其中的字符串字面值常量指出了编写函数所用的语言。编译器应该支持对 C 语言的链接指示。此外，编译器也可能会支持其他语言的链接指示，如 `extern "Ada"`、`extern "FORTRAN"` 等。

链接指示与头文件

我们可以令链接指示后面跟上花括号括起来的若干函数的声明，从而一次性建立多个链接。花括号的作用是将适用于该链接指示的多个声明聚合在一起，否则花括号就会被忽略，花括号中声明的函数名字就是可见的，就好像在花括号之外声明的一样。

多重声明的形式可以应用于整个头文件。例如，C++ 的 `cstring` 头文件可能形如：

```
// 复合语句链接指示
extern "C" {
#include <string.h>           // 操作 C 风格字符串的 C 函数
}
```

当一个 `#include` 指示被放置在复合链接指示的花括号中时，头文件中的所有普通函数声明都被认为是由链接指示的语言编写的。链接指示可以嵌套，因此如果头文件包含带自带链接指示的函数，则该函数的链接不受影响。



C++从 C 语言继承的标准库函数可以定义成 C 函数，但并非必须：决定使用 C 还是 C++实现 C 标准库，是每个 C++实现的事情。

<859

指向 `extern "C"` 函数的指针

编写函数所用的语言是函数类型的一部分。因此，对于使用链接指示定义的函数来说，它的每个声明都必须使用相同的链接指示。而且，指向其他语言编写的函数的指针必须与函数本身使用相同的链接指示：

```
// pf 指向一个 C 函数，该函数接受一个 int 返回 void
extern "C" void (*pf)(int);
```

当我们使用 `pf` 调用函数时，编译器认定当前调用的是一个 C 函数。

指向 C 函数的指针与指向 C++ 函数的指针是不一样的类型。一个指向 C 函数的指针

不能用在执行初始化或赋值操作后指向 C++ 函数，反之亦然。就像其他类型不匹配的问题一样，如果我们试图在两个链接指示不同的指针之间进行赋值操作，则程序将发生错误：

```
void (*pf1)(int); // 指向一个 C++ 函数
extern "C" void (*pf2)(int); // 指向一个 C 函数
pf1 = pf2; // 错误：pf1 和 pf2 的类型不同
```



WARNING

有的 C++ 编译器会接受之前的这种赋值操作并将其作为对语言的扩展，尽管从严格意义上来看它是非法的。

链接指示对整个声明都有效

当我们使用链接指示时，它不仅对函数有效，而且对作为返回类型或形参类型的函数指针也有效：

```
// f1 是一个 C 函数，它的形参是一个指向 C 函数的指针
extern "C" void f1(void(*)(int));
```

这条声明语句指出 f1 是一个不返回任何值的 C 函数。它有一个类型是函数指针的形参，其中的函数接受一个 int 形参返回为空。这个链接指示不仅对 f1 有效，对函数指针同样有效。当我们调用 f1 时，必须传给它一个 C 函数的名字或者指向 C 函数的指针。

因为链接指示同时作用于声明语句中的所有函数，所以如果我们希望给 C++ 函数传入一个指向 C 函数的指针，则必须使用类型别名（参见 2.5.1 节，第 60 页）：

```
860 // fC 是一个指向 C 函数的指针
extern "C" typedef void FC(int);
// f2 是一个 C++ 函数，该函数的形参是指向 C 函数的指针
void f2(FC *);
```

导出 C++ 函数到其他语言

通过使用链接指示对函数进行定义，我们可以令一个 C++ 函数在其他语言编写的程序中可用：

```
// calc 函数可以被 C 程序调用
extern "C" double calc(double dparam) { /* ... */ }
```

编译器将为该函数生成适合于指定语言的代码。

值得注意的是，可被多种语言共享的函数的返回类型或形参类型受到很多限制。例如，我们不太可能把一个 C++ 类的对象传给 C 程序，因为 C 程序根本无法理解构造函数、析构函数以及其他类特有的操作。

对链接到 C 的预处理器的支持

有时需要在 C 和 C++ 中编译同一个源文件，为了实现这一目的，在编译 C++ 版本的程序时预处理器定义 `__cplusplus`（两个下画线）。利用这个变量，我们可以在编译 C++ 程序的时候有条件地包含进来一些代码：

```
#ifdef __cplusplus
// 正确：我们正在编译 C++ 程序
extern "C"
#endif
int strcmp(const char*, const char*);
```

重载函数与链接指示

链接指示与重载函数的相互作用依赖于目标语言。如果目标语言支持重载函数，则为该语言实现链接指示的编译器很可能也支持重载这些 C++ 的函数。

C 语言不支持函数重载，因此也就不难理解为什么一个 C 链接指示只能用于说明一组重载函数中的某一个了：

```
// 错误：两个 extern "C" 函数的名字相同
extern "C" void print(const char*);
extern "C" void print(int);
```

如果在一组重载函数中有一个是 C 函数，则其余的必定都是 C++ 函数：

```
class SmallInt { /* ... */ };
class BigNum { /* ... */ };
// C 函数可以在 C 或 C++ 程序中调用
// C++ 函数重载了该函数，可以在 C++ 程序中调用
extern "C" double calc(double);
extern SmallInt calc(const SmallInt&);
extern BigNum calc(const BigNum&);
```

861

C 版本的 calc 函数可以在 C 或 C++ 程序中调用，而使用了类类型形参的 C++ 函数只能在 C++ 程序中调用。上述性质与声明的顺序无关。

19.8.3 节练习

练习 19.26：说明下列声明语句的含义并判断它们是否合法：

```
extern "C" int compute(int *, int);
extern "C" double compute(double *, double);
```

862 小结

C++为解决某些特殊问题设置了一系列特殊的处理机制。

有的程序需要精确控制内存分配过程，它们可以通过在类的内部或在全局作用域中自定义 `operator new` 和 `operator delete` 来实现这一目的。如果应用程序为这两个操作定义了自己的版本，则 `new` 和 `delete` 表达式将优先使用应用程序定义的版本。

有的程序需要在运行时直接获取对象的动态类型，运行时类型识别（RTTI）为这种程序提供了语言级别的支持。RTTI 只对定义了虚函数的类有效；对没有定义虚函数的类，虽然也可以得到其类型信息，但只是静态类型。

当我们定义指向类成员的指针时，在指针类型中包含了该指针所指成员所属类的类型信息。成员指针可以绑定到该类当中任意一个具有指定类型的成员上。当我们解引用成员指针时，必须提供获取成员所需的对象。

C++定义了另外几种聚集类型：

- 嵌套类，定义在其他类的作用域中，嵌套类通常作为外层类的实现类。
- `union`，是一种特殊的类，它可以定义几个数据成员但是在任意时刻只有一个成员有值，`union` 通常嵌套在其他类的内部。
- 局部类，定义在函数的内部，局部类的所有成员都必须定义在类内，局部类不能含有静态数据成员。

C++支持几种固有的不可移植的特性，其中位域和 `volatile` 使得程序更容易访问硬件；链接指示使得程序更容易访问用其他语言编写的代码。

术语表

匿名 union (anonymous union) 未命名的 `union`，不能用于定义对象。匿名 `union` 的成员也是外层作用域的成员。匿名 `union` 不能包含成员函数，也不能包含私有成员或受保护的成员。

位域 (bit-field) 特殊的类成员，该成员含有一个整型值以指定为其分配的二进制位数。如果可能的话，在类中连续定义的位域将被压缩在一个普通的整数值当中。

判别式 (discriminant) 是一种使用一个对象判断 `union` 的当前值类型的编程技术。

dynamic_cast 是一个运算符，执行从基类向派生类的带检查的强制类型转换。当基类中至少含有一个虚函数时，该运算符负责检查指针或引用所绑定的对象的动态类型。如果对象类型与目标类型（或其派生类）一致，则类型转换完成。否则，指针

转换将返回一个值为 0 的指针；引用转换将抛出一个异常。

枚举类型 (enumeration) 将一组整型常量命名后聚合在一起形成的类型。

枚举成员 (enumerator) 是枚举类型的成员。枚举成员是常量，可以用在任何需要整型常量的地方。

free 是定义在 `cstdlib` 中的低层函数，负责释放内存。`free` 只能释放由 `malloc` 分配的内存。

链接指示 (linkage directive) 支持 C++ 程序调用其他语言编写的函数的一种机制。所有编译器都应支持调用 C++ 和 C 函数，至于是否支持其他语言则由编译器决定。

局部类 (local class) 定义在函数中的类。局部类只有在其外层函数内可见。局部类

的所有成员都必须定义在类的内部。局部类不能含有静态成员。局部类成员不能访问外层函数的非静态变量，只能访问类型名字、静态变量或枚举成员。

malloc 是定义在 `cstdlib` 中的低层函数，负责分配内存。**malloc** 分配的内存必须由 `free` 释放。

mem_fn 是一个标准库类模板，根据指向成员函数的指针生成一个可调用对象。

嵌套类 (nested class) 定义在其他类内部的类，嵌套类定义在它的外层作用域中：在外层类的作用域中嵌套类的名字必须唯一，在外层类之外可以被重用。在外层类之外访问嵌套类需要用作用域运算符指明嵌套类所属的范围。

嵌套类型 (nested type) “嵌套类”的同义词。

不可移植 (nonportable) 固有的与机器有关的特性，当程序转移到其他机器或编译器上时需要修改代码。

operator delete 是一个标准库函数，用于释放由 `operator new` 分配的未指明类型的、未构造的内存空间。相应的，`operator delete[]` 释放由 `operator new[]` 为数组分配的内存。

operator new 是一个标准库函数，用于分配一个给定大小的、未指明类型的、未构造的内存空间。标准库函数 `operator new[]` 为数组分配原始内存。与 `allocator` 类相比，这两个标准库函数提供的内存分配机制更低级。现代的 C++ 程序应该使用 `allocator` 而不是这两个函数。

定位 new 表达式 (placement new expression) 是 `new` 的一种特殊形式，在给定的内存中构造对象。它不分配内存，而是根据实参指定在哪儿构造对象。它是对 `allocator` 类的 `construct` 成员的行为的一种低级模拟。

成员指针 (pointer to member) 其中既包含类类型，也包含指针所指的成员类型。

成员指针的定义必须同时指定类的名字以及指针所指的成员类型：

```
T C::*pmem = &C::member;
```

该语句将 `pmem` 定义为一个指针，它可以指向类 `C` 的成员，并且该成员的类型是 `T`，然后初始化 `pmem` 令其指向类 `C` 的名为 `member` 的成员。要使用该指针，我们必须提供 `C` 的一个对象或指针：

```
classobj.*pmem;
```

```
classptr->*pmem;
```

从 `classptr` 所指的对象 `classobj` 中获取 `member`。 ◀864

运行时类型识别 (run-time type identification) 是 C++ 的一种特性，允许在运行时获取指针或引用的动态类型。RTTI 运算符包括 `typeid` 和 `dynamic_cast`，为含有虚函数的类的指针或引用提供动态类型。当作用于其他类型时，返回的结果是指针或引用的静态类型。

限定作用域的枚举类型 (scoped enumeration) 是一种新的枚举类型，它的枚举成员不能被外层作用域直接访问。

typeid 运算符 (typeid operator) 是一个一元运算符，返回标准库类型 `type_info` 的引用，表示给定表达式的类型。当表达式是某个含有虚函数的类型的对象时，返回表达式的动态类型；此类表达式在运行时求值。如果表达式的类型是指针、引用或其他未定义虚函数的类型，则返回指针、引用或对象的静态类型；此类表达式不会被求值。

type_info typeid 运算符返回的标准库类型。`type_info` 的细节因机器而异，但是必须提供一组操作，其中名为 `name` 的函数负责返回一个表示类型名字的字符串。`type_info` 对象不能被拷贝、移动或赋值。

联合 (union) 是一种和类有些相似的类型。可以包含多个数据成员，但是同一时刻只能有一个成员有值。联合可以有包括

构造函数和析构函数在内的成员函数。联合不能被用作基类。在 C++11 新标准中，联合可以含有类类型的成员，前提是这些类自定义了拷贝控制成员。对于这样的联合来说，如果它们没有定义自己的拷贝控制成员，则编译器将为它们生成删除的版本。

不限定作用域的枚举类型（unscoped enumeration） 该枚举类型的枚举成员在外层作用域中可以访问。

volatile 是一种类型限定符，告诉编译器变量可能在程序的直接控制之外发生改变。它起到一种标示的作用，令编译器不对代码进行优化操作。

附录 A

标准库

内容

A.1 标准库名字和头文件	766
A.2 算法概览	770
A.3 随机数	781

本附录介绍了标准库中算法和随机数部分的一些额外细节。这里还提供了一个我们使用过的所有标准库名字的列表，列表中给出了每个名字所在的头文件。

在第 10 章中我们使用过一些较常用的算法，并且描述了算法之下的架构。在本附录中，我们将列出所有标准库算法，按它们执行的操作的种类来组织。

在 17.4 节（第 660 页）中我们描述了随机数库的架构，并使用了几个分布类型。库中定义了若干随机数引擎和 20 种不同的分布。在本附录中，我们将列出所有引擎和分布类型。

866 A.1 标准库名字和头文件

本书中大多数代码没有给出编译程序所需的实际#*include* 指令。为了方便读者，表A.1列出了本书程序用到的标准库名字以及它们所在的头文件。

表 A.1: 标准库名字和头文件

名字	头文件
abort	<cstdlib>
accumulate	<numeric>
allocator	<memory>
array	<array>
auto_ptr	<memory>
back_inserter	<iterator>
bad_alloc	<new>
bad_array_new_length	<new>
bad_cast	<typeinfo>
begin	<iterator>
bernoulli_distribution	<random>
bind	<functional>
bitset	<bitset>
boolalpha	<iostream>
cerr	<iostream>
cin	<iostream>
cmatch	<regex>
copy	<algorithm>
count	<algorithm>
count_if	<algorithm>
cout	<iostream>
cref	<functional>
csub_match	<regex>
dec	<iostream>
default_float_engine	<iostream>
default_random_engine	<random>
deque	<deque>
domain_error	<stdexcept>
end	<iterator>
endl	<iostream>
ends	<iostream>
equal_range	<algorithm>
exception	<exception>
fill	<algorithm>
fill_n	<algorithm>
find	<algorithm>

续表

名字	头文件
find_end	<algorithm>
find_first_of	<algorithm>
find_if	<algorithm>
fixed	<iostream>
flush	<iostream>
for_each	<algorithm>
forward	<utility>
forward_list	<forward_list>
free	<cstdlib>
front_inserter	<iterator>
fstream	<fstream>
function	<functional>
get	<tuple>
getline	<string>
greater	<functional>
hash	<functional>
hex	<iostream>
hexfloat	<iostream>
ifstream	<fstream>
initializer_list	<initializer_list>
inserter	<iterator>
internal	<iostream>
ios_base	<ios_base>
isalpha	<cctype>
islower	<cctype>
isprint	<cctype>
ispunct	<cctype>
isspace	<cctype>
istream	<iostream>
istream_iterator	<iterator>
istringstream	<sstream>
isupper	<cctype>
left	<iostream>
less	<functional>
less_equal	<functional>
list	<list>
logic_error	<stdexcept>
lower_bound	<algorithm>
lround	<cmath>
make_move_iterator	<iterator>
make_pair	<utility>

867

续表

868

名字	头文件
make_shared	<memory>
make_tuple	<tuple>
malloc	<cstdlib>
map	<map>
max	<algorithm>
max_element	<algorithm>
mem_fn	<functional>
min	<algorithm>
move	<utility>
multimap	<map>
multiset	<set>
negate	<functional>
noboolalpha	<iostream>
normal_distribution	<random>
noshowbase	<iostream>
noshowpoint	<iostream>
noskipws	<iostream>
not1	<functional>
nothrow	<new>
nothrow_t	<new>
nounitbuf	<iostream>
nouppercase	<iostream>
nth_element	<algorithm>
oct	<iostream>
ofstream	<fstream>
ostream	<iostream>
ostream_iterator	<iterator>
ostringstream	<sstream>
out_of_range	<stdexcept>
pair	<utility>
partial_sort	<algorithm>
placeholders	<functional>
placeholders::_1	<functional>
plus	<functional>
priority_queue	<queue>
ptrdiff_t	<cstddef>
queue	<queue>
rand	<random>
random_device	<random>
range_error	<stdexcept>
ref	<functional>

续表

名字	头文件
regex	<regex>
regex_constants	<regex>
regex_error	<regex>
regex_match	<regex>
regex_replace	<regex>
regex_search	<regex>
remove_pointer	<type_traits>
remove_reference	<type_traits>
replace	<algorithm>
replace_copy	<algorithm>
reverse_iterator	<iterator>
right	<iostream>
runtime_error	<stdexcept>
scientific	<iostream>
set	<set>
set_difference	<algorithm>
set_intersection	<algorithm>
set_union	<algorithm>
setfill	<iomanip>
setprecision	<iomanip>
setw	<iomanip>
shared_ptr	<memory>
showbase	<iostream>
showpoint	<iostream>
size_t	<cstddef>
skipws	<iostream>
smatch	<regex>
sort	<algorithm>
sqrt	<cmath>
sregex_iterator	<regex>
ssub_match	<regex>
stable_sort	<algorithm>
stack	<stack>
stoi	<string>
strcmp	<cstring>
strcpy	<cstring>
string	<string>
stringstream	<sstream>
strlen	<cstring>
strncpy	<cstring>
strtod	<string>

<869

续表

870

名字	头文件
swap	<utility>
terminate	<exception>
time	<ctime>
tolower	<cctype>
toupper	<cctype>
transform	<algorithm>
tuple	<tuple>
tuple_element	<tuple>
tuple_size	<tuple>
type_info	<typeinfo>
unexpected	<exception>
uniform_int_distribution	<random>
uniform_real_distribution	<random>
uninitialized_copy	<memory>
uninitialized_fill	<memory>
unique	<algorithm>
unique_copy	<algorithm>
unique_ptr	<memory>
unitbuf	<iostream>
unordered_map	<unordered_map>
unordered_multimap	<unordered_map>
unordered_multiset	<unordered_set>
unordered_set	<unordered_set>
upper_bound	<algorithm>
uppercase	<iostream>
vector	<vector>
weak_ptr	<memory>

A.2 算法概览

标准库定义了超过 100 个算法。要想高效使用这些算法需要了解它们的结构而不是单纯记忆每个算法的细节。因此，我们在第 10 章中关注标准库算法架构的描述和理解。在本节中，我们将简要描述每个算法，在下面的描述中，

- beg 和 end 是表示元素范围的迭代器（参见 9.2.1 节，第 296 页）。几乎所有算法都对一个由 beg 和 end 表示的序列进行操作。
- beg2 是表示第二个输入序列开始位置的迭代器。end2 表示第二个序列的末尾位置（如果有的话）。如果没有 end2，则假定 beg2 表示的序列与 beg 和 end 表示的序列一样大。beg 和 beg2 的类型不必匹配，但是，必须保证对两个序列中的元素都可以执行特定操作或调用给定的可调用对象。
- dest 是表示目的序列的迭代器。对于给定输入序列，算法需要生成多少元素，目的序列必须保证能保存同样多的元素。

- `unaryPred` 和 `binaryPred` 是一元和二元谓词（参见 10.3.1 节，第 344 页），分别接受一个和两个参数，都是来自输入序列的元素，两个谓词都返回可用作条件的类型。
- `comp` 是一个二元谓词，满足关联容器中对关键字序的要求（参见 11.2.2 节，第 378 页）。
- `unaryOp` 和 `binaryOp` 是可调用对象（参见 10.3.2 节，第 346 页），可分别使用来自输入序列的一个和两个实参来调用。

A.2.1 查找对象的算法

这些算法在一个输入序列中搜索一个指定值或一个值的序列。

每个算法都提供两个重载的版本，第一个版本使用底层类型的相等运算符（`==`）来比较元素；第二个版本使用用户给定的 `unaryPred` 和 `binaryPred` 比较元素。

简单查找算法

这些算法查找指定值，要求输入迭代器（`input iterator`）。

```
find(beg, end, val)
find_if(beg, end, unaryPred)
find_if_not(beg, end, unaryPred)
count(beg, end, val)
count_if(beg, end, unaryPred)
```

`find` 返回一个迭代器，指向输入序列中第一个等于 `val` 的元素。

`find_if` 返回一个迭代器，指向第一个满足 `unaryPred` 的元素。

`find_if_not` 返回一个迭代器，指向第一个令 `unaryPred` 为 `false` 的元素。上述三个算法在未找到元素时都返回 `end`。

`count` 返回一个计数器，指出 `val` 出现了多少次；`count_if` 统计有多少个元素满足 `unaryPred`。

```
all_of(beg, end, unaryPred)
any_of(beg, end, unaryPred)
none_of(beg, end, unaryPred)
```

这些算法都返回一个 `bool` 值，分别指出 `unaryPred` 是否对所有元素都成功、对任意一个元素成功以及对所有元素都不成功。如果序列为空，`any_of` 返回 `false`，而 `all_of` 和 `none_of` 返回 `true`。

查找重复值的算法

下面这些算法要求前向迭代器（`forward iterator`），在输入序列中查找重复元素。

```
adjacent_find(beg, end)
adjacent_find(beg, end, binaryPred)
```

返回指向第一对相邻重复元素的迭代器。如果序列中无相邻重复元素，则返回 `end`。

```
search_n(beg, end, count, val)
search_n(beg, end, count, val, binaryPred)
```

返回一个迭代器，从此位置开始有 `count` 个相等元素。如果序列中不存在这样的子序列，

则返回 end。

872 > 查找子序列的算法

在下面的算法中，除了 `find_first_of` 之外，都要求两个前向迭代器。`find_first_of` 用输入迭代器表示第一个序列，用前向迭代器表示第二个序列。这些算法搜索子序列而不是单个元素。

```
search(beg1, end1, beg2, end2)
search(beg1, end1, beg2, end2, binaryPred)
```

返回第二个输入范围（子序列）在第一个输入范围中第一次出现的位置。如果未找到子序列，则返回 `end1`。

```
find_first_of(beg1, end1, beg2, end2)
find_first_of(beg1, end1, beg2, end2, binaryPred)
```

返回一个迭代器，指向第二个输入范围中任意元素在第一个范围中首次出现的位置。如果未找到匹配元素，则返回 `end1`。

```
find_end(beg1, end1, beg2, end2)
find_end(beg1, end1, beg2, end2, binaryPred)
```

类似 `search`，但返回的是最后一次出现的位置。如果第二个输入范围为空，或者在第一个输入范围中未找到它，则返回 `end1`。

A.2.2 其他只读算法

这些算法要求前两个实参都是输入迭代器。

`equal` 和 `mismatch` 算法还接受一个额外的输入迭代器，表示第二个范围的开始位置。这两个算法都提供两个重载的版本。第一个版本使用底层类型的相等运算符（`==`）比较元素，第二个版本则用用户指定的 `unaryPred` 或 `binaryPred` 比较元素。

```
for_each(beg, end, unaryOp)
```

对输入序列中的每个元素应用可调用对象（参见 10.3.2 节，第 346 页）`unaryOp.unaryOp` 的返回值（如果有的话）被忽略。如果迭代器允许通过解引用运算符向序列中的元素写入值，则 `unaryOp` 可能修改元素。

```
mismatch(beg1, end1, beg2)
mismatch(beg1, end1, beg2, binaryPred)
```

比较两个序列中的元素。返回一个迭代器的 `pair`（参见 11.2.3 节，第 379 页），表示两个序列中第一个不匹配的元素。如果所有元素都匹配，则返回的 `pair` 中第一个迭代器为 `end1`，第二个迭代器指向 `beg2` 中偏移量等于第一个序列长度的位置。

```
equal(beg1, end1, beg2)
equal(beg1, end1, beg2, binaryPred)
```

确定两个序列是否相等。如果输入序列中每个元素都与从 `beg2` 开始的序列中对应元素相等，则返回 `true`。

873 > A.2.3 二分搜索算法

这些算法都要求前向迭代器，但这些算法都经过了优化，如果我们提供随机访问迭代

器 (random-access iterator) 的话, 它们的性能会好得多。从技术上讲, 无论我们提供什么类型的迭代器, 这些算法都会执行对数次的比较操作。但是, 当使用前向迭代器时, 这些算法必须花费线性次数的迭代器操作来移动到序列中要比较的元素。

这些算法要求序列中的元素已经是有序的。它们的行为类似关联容器的同名成员 (参见 11.3.5 节, 第 389 页)。`equal_range`、`lower_bound` 和 `upper_bound` 算法返回迭代器, 指向给定元素在序列中的正确插入位置——插入后还能保持有序。如果给定元素比序列中的所有元素都大, 则会返回尾后迭代器。

每个算法都提供两个版本: 第一个版本用元素类型的小于运算符 (`<`) 来检测元素; 第二个版本则使用给定的比较操作。在下列算法中, “`x 小于 y`” 表示 `x < y` 或 `comp(x, y)` 成功。

```
lower_bound(beg, end, val)
lower_bound(beg, end, val, comp)
```

返回一个迭代器, 表示第一个小于等于 `val` 的元素, 如果不存在这样的元素, 则返回 `end`。

```
upper_bound(beg, end, val)
upper_bound(beg, end, val, comp)
```

返回一个迭代器, 表示第一个大于 `val` 的元素, 如果不存在这样的元素, 则返回 `end`。

```
equal_range(beg, end, val)
equal_range(beg, end, val, comp)
```

返回一个 `pair` (参见 11.2.3 节, 第 379 页), 其 `first` 成员是 `lower_bound` 返回的迭代器, `second` 成员是 `upper_bound` 返回的迭代器。

```
binary_search(beg, end, val)
binary_search(beg, end, val, comp)
```

返回一个 `bool` 值, 指出序列中是否包含等于 `val` 的元素。对于两个值 `x` 和 `y`, 当 `x` 不小于 `y` 且 `y` 也不小于 `x` 时, 认为它们相等。

A.2.4 写容器元素的算法

很多算法向给定序列中的元素写入新值。这些算法可以从不同角度加以区分: 通过表示输入序列的迭代器类型来区分; 或者通过是写入输入序列中元素还是写入给定目的位置来区分。

只写不读元素的算法

< 874

这些算法要求一个输出迭代器 (output iterator), 表示目的位置。`_n` 结尾的版本接受第二个实参, 表示写入的元素数目, 并将给定数目的元素写入到目的位置中。

```
fill(beg, end, val)
fill_n(dest, cnt, val)
generate(beg, end, Gen)
generate_n(dest, cnt, Gen)
```

给输入序列中每个元素赋予一个新值。`fill` 将值 `val` 赋予元素; `generate` 执行生成器对象 `Gen()` 生成新值。生成器是一个可调用对象 (参见 10.3.2 节, 第 346 页), 每次调用会生成一个不同的返回值。`fill` 和 `generate` 都返回 `void`。`_n` 版本返回一个迭代器, 指向写入到输出序列的最后一个元素之后的位置。

使用输入迭代器的写算法

这些算法读取一个输入序列，将值写入到一个输出序列中。它们要求一个名为 dest 的输出迭代器，而表示输入范围的迭代器必须是输入迭代器。

```
copy(beg, end, dest)
copy_if(beg, end, dest, unaryPred)
copy_n(beg, n, dest)
```

从输入范围将元素拷贝到 dest 指定的目的序列。copy 拷贝所有元素，copy_if 拷贝那些满足 unaryPred 的元素，copy_n 拷贝前 n 个元素。输入序列必须有至少 n 个元素。

```
move(beg, end, dest)
```

对输入序列中的每个元素调用 std:: move (参见 13.6.1 节，第 472 页)，将其移动到迭代器 dest 开始的序列中。

```
transform(beg, end, dest, unaryOp)
transform(beg, end, beg2, dest, binaryOp)
```

调用给定操作，并将结果写到 dest 中。第一个版本对输入范围内每个元素应用一元操作。第二个版本对两个输入序列中的元素应用二元操作。

```
replace_copy(beg, end, dest, old_val, new_val)
replace_copy_if(beg, end, dest, unaryPred, new_val)
```

将每个元素拷贝到 dest，将指定的元素替换为 new_val。第一个版本替换那些 ==old_val 的元素。第二个版本替换那些满足 unaryPred 的元素。

```
merge(beg1, end1, beg2, end2, dest)
merge(beg1, end1, beg2, end2, dest, comp)
```

两个输入序列必须都是有序的。将合并后的序列写入到 dest 中。第一个版本用<运算符比较元素；第二个版本则使用给定比较操作。

875 使用前向迭代器的写算法

这些算法要求前向迭代器，由于它们是向输入序列写入元素，迭代器必须具有写入元素的权限。

```
iter_swap(iterator1, iterator2)
swap_ranges(beg1, end1, beg2)
```

交换 iter1 和 iter2 所表示的元素，或将输入范围内所有元素与 beg2 开始的第二个序列中所有元素进行交换。两个范围不能有重叠。iter_swap 返回 void，swap_ranges 返回递增后的 beg2，指向最后一个交换元素之后的位置。

```
replace(beg, end, old_val, new_val)
replace_if(beg, end, unaryPred, new_val)
```

用 new_val 替换每个匹配元素。第一个版本使用==比较元素与 old_val，第二个版本替换那些满足 unaryPred 的元素。

使用双向迭代器的写算法

这些算法需要在序列中有反向移动的能力，因此它们要求双向迭代器 (bidirectional iterator)。

```
copy_backward(beg, end, dest)
```

move_backward(beg, end, dest)

从输入范围中拷贝或移动元素到指定目的位置。与其他算法不同，`dest` 是输出序列的尾后迭代器（即，目的序列恰在 `dest` 之前结束）。输入范围中的尾元素被拷贝或移动到目的序列的尾元素，然后是倒数第二个元素被拷贝/移动，依此类推。元素在目的序列中的顺序与在输入序列中相同。如果范围为空，则返回值为 `dest`；否则，返回值表示从 `*beg` 中拷贝或移动的元素。

**inplace_merge(beg, mid, end)
inplace_merge(beg, mid, end, comp)**

将同一个序列中的两个有序子序列合并为单一的有序序列。`beg` 到 `mid` 间的子序列和 `mid` 到 `end` 间的子序列被合并，并被写入到原序列中。第一个版本使用<比较元素，第二个版本使用给定的比较操作，返回 `void`。

A.2.5 划分与排序算法

对于序列中的元素进行排序，排序和划分算法提供了多种策略。

每个排序和划分算法都提供稳定和不稳定版本（参见 10.3.1 节，第 345 页）。稳定算法保证保持相等元素的相对顺序。由于稳定算法会做更多工作，可能比不稳定版本慢得多并消耗更多内存。

划分算法

< 876

一个划分算法将输入范围中的元素划分为两组。第一组包含那些满足给定谓词的元素，第二组则包含不满足谓词的元素。例如，对于一个序列中的元素，我们可以根据元素是否是奇数或者单词是否以大写字母开头等来划分它们。这些算法都要求双向迭代器。

is_partitioned(beg, end, unaryPred)

如果所有满足谓词 `unaryPred` 的元素都在不满足 `unaryPred` 的元素之前，则返回 `true`。若序列为空，也返回 `true`。

partition_copy(beg, end, dest1, dest2, unaryPred)

将满足 `unaryPred` 的元素拷贝到 `dest1`，并将不满足 `unaryPred` 的元素拷贝到 `dest2`。返回一个迭代器 `pair` (11.2.3 节，第 379 页)，其 `first` 成员表示拷贝到 `dest1` 的元素的末尾，`second` 表示拷贝到 `dest2` 的元素的末尾。输入序列与两个目的序列都不能重叠。

partition_point(beg, end, unaryPred)

输入序列必须是已经用 `unaryPred` 划分过的。返回满足 `unaryPred` 的范围的尾后迭代器。如果返回的迭代器不是 `end`，则它指向的元素及其后的元素必须都不满足 `unaryPred`。

**stable_partition(beg, end, unaryPred)
partition(beg, end, unaryPred)**

使用 `unaryPred` 划分输入序列。满足 `unaryPred` 的元素放置在序列开始，不满足的元素放在序列尾部。返回一个迭代器，指向最后一个满足 `unaryPred` 的元素之后的位置，如果所有元素都不满足 `unaryPred`，则返回 `beg`。

排序算法

这些算法要求随机访问迭代器。每个排序算法都提供两个重载的版本。一个版本用元

素的`<`运算符来比较元素，另一个版本接受一个额外参数来指定排序关系（11.2.2 节，第 378 页）。`partial_sort_copy` 返回一个指向目的位置的迭代器，其他排序算法都返回 `void`。

`partial_sort` 和 `nth_element` 算法都只进行部分排序工作，它们常用于不需要排序整个序列の場合。由于这些算法工作量更少，它们通常比排序整个输入序列的算法更快。

```
sort(beg, end)
stable_sort(beg, end)
sort(beg, end, comp)
stable_sort(beg, end, comp)
```

排序整个范围。

877 ➤ `is_sorted(beg, end)`
`is_sorted(beg, end, comp)`
`is_sorted_until(beg, end)`
`is_sorted_until(beg, end, comp)`

`is_sorted` 返回一个 `bool` 值，指出整个输入序列是否有序。`is_sorted_until` 在输入序列中查找最长初始有序子序列，并返回子序列的尾后迭代器。

```
partial_sort(beg, mid, end)
partial_sort(beg, mid, end, comp)
```

排序 `mid-beg` 个元素。即，如果 `mid-beg` 等于 42，则此函数将值最小的 42 个元素有序放在序列前 42 个位置。当 `partial_sort` 完成后，从 `beg` 开始直至 `mid` 之前的范围中的元素就都已排好序了。已排序范围中的元素都不会比 `mid` 后的元素更大。未排序区域中元素的顺序是未指定的。

```
partial_sort_copy(beg, end, destBeg, destEnd)
partial_sort_copy(beg, end, destBeg, destEnd, comp)
```

排序输入范围中的元素，并将足够多的已排序元素放到 `destBeg` 和 `destEnd` 所指示的序列中。如果目的范围的大小大于等于输入范围，则排序整个输入序列并存入从 `destBeg` 开始的范围。如果目的范围大小小于输入范围，则只拷贝输入序列中与目的范围一样多的元素。

算法返回一个迭代器，指向目的范围中已排序部分的尾后迭代器。如果目的序列的大小小于或等于输入范围，则返回 `destEnd`。

```
nth_element(beg, nth, end)
nth_element(beg, nth, end, comp)
```

参数 `nth` 必须是一个迭代器，指向输入序列中的一个元素。执行 `nth_element` 后，此迭代器指向的元素恰好是整个序列排好序后此位置上的值。序列中的元素会围绕 `nth` 进行划分：`nth` 之前的元素都小于等于它，而之后的元素都大于等于它。

A.2.6 通用重排操作

这些算法重排输入序列中元素的顺序。前两个算法 `remove` 和 `unique`，会重排序列，使得排在序列第一部分的元素满足某种标准。它们返回一个迭代器，标记子序列的末尾。其他算法，如 `reverse`、`rotate` 和 `random_shuffle` 都重排整个序列。

这些算法的基本版本都进行“原址”操作，即，在输入序列自身内部重排元素。三个

重排算法提供“拷贝”版本。这些_copy 版本完成相同的重排工作，但将重排后的元素写入到一个指定目的序列中，而不是改变输入序列。这些算法要求输出迭代器来表示目的序列。

使用前向迭代器的重排算法

< 878

这些算法重排输入序列。它们要求迭代器至少是前向迭代器。

```
remove(beg, end, val)
remove_if(beg, end, unaryPred)
remove_copy(beg, end, dest, val)
remove_copy_if(beg, end, dest, unaryPred)
```

从序列中“删除”元素，采用的办法是用保留的元素覆盖要删除的元素。被删除的是那些`==val` 或满足 `unaryPred` 的元素。算法返回一个迭代器，指向最后一个删除元素的尾后位置。

```
unique(beg, end)
unique(beg, end, binaryPred)
unique_copy(beg, end, dest)
unique_copy_if(beg, end, dest, binaryPred)
```

重排序列，对相邻的重复元素，通过覆盖它们来进行“删除”。返回一个迭代器，指向不重复元素的尾后位置。第一个版本用`==`确定两个元素是否相同，第二个版本使用谓词检测相邻元素。

```
rotate(beg, mid, end)
rotate_copy(beg, mid, end, dest)
```

围绕 `mid` 指向的元素进行元素转动。元素 `mid` 成为首元素，随后是 `mid+1` 到 `end` 之前的元素，再接着是 `beg` 到 `mid` 之前的元素。返回一个迭代器，指向原来在 `beg` 位置的元素。

使用双向迭代器的重排算法

由于这些算法要反向处理输入序列，它们要求双向迭代器。

```
reverse(beg, end)
reverse_copy(beg, end, dest)
```

翻转序列中的元素。`reverse` 返回 `void`，`reverse_copy` 返回一个迭代器，指向拷贝到目的序列的元素的尾后位置。

使用随机访问迭代器的重排算法

由于这些算法要随机重排元素，它们要求随机访问迭代器。

```
random_shuffle(beg, end)
random_shuffle(beg, end, rand)
shuffle(beg, end, Uniform_rand)
```

混洗输入序列中的元素。第二个版本接受一个可调用对象参数，该对象必须接受一个正整数值，并生成 0 到此值的包含区间内的一个服从均匀分布的随机整数。`shuffle` 的第三个参数必须满足均匀分布随机数生成器的要求（参见 17.4 节，第 659 页）。所有版本都返回 `void`。

< 879

A.2.7 排列算法

排列算法生成序列的字典序排列。对于一个给定序列，这些算法通过重排它的一个排列来生成字典序中下一个或前一个排列。算法返回一个 `bool` 值，指出是否还有下一个或前一个排列。

为了理解什么是下一个或前一个排列，考虑下面这个三字符的序列：abc。它有六种可能的排列：abc、acb、bac、bca、cab 及 cba。这些排列是按字典序递增序列出的。即，abc 是第一个排列，这是因为它的第一个元素小于或等于任何其他排列的首元素，并且它的第二个元素小于任何其他首元素相同的排列。类似的，acb 排在下一位，原因是它以 a 开头，小于任何剩余排列的首元素。同理，以 b 开头的排列也都排在以 c 开头的排列之前。

对于任意给定的排列，基于单个元素的一个特定的序，我们可以获得它的前一个和下一个排列。给定排列 bca，我们知道其前一个排列为 bac，下一个排列为 cab。序列 abc 没有前一个排列，而 cba 没有下一个排列。

这些算法假定序列中的元素都是唯一的，即，没有两个元素的值是一样的。

为了生成排列，必须既向前又向后处理序列，因此算法要求双向迭代器。

```
is_permutation(beg1, end1, beg2)
is_permutation(beg1, end1, beg2, binaryPred)
```

如果第二个序列的某个排列和第一个序列具有相同数目的元素，且元素都相等，则返回 `true`。第一个版本用`==`比较元素，第二个版本使用给定的 `binaryPred`。

```
next_permutation(beg, end)
next_permutation(beg, end, comp)
```

如果序列已经是最后一个排列，则 `next_permutation` 将序列重排为最小的排列，并返回 `false`。否则，它将输入序列转换为字典序中下一个排列，并返回 `true`。第一个版本使用元素的`<`运算符比较元素，第二个版本使用给定的比较操作。

```
prev_permutation(beg, end)
prev_permutation(beg, end, comp)
```

类似 `next_permutation`，但将序列转换为前一个排列。如果序列已经是最小的排列，则将其重排为最大的排列，并返回 `false`。

880 A.2.8 有序序列的集合算法

集合算法实现了有序序列上的一般集合操作。这些算法与标准库 `set` 容器不同，不要与 `set` 上的操作相混淆。这些算法提供了普通顺序容器（`vector`、`list` 等）或其他序列（如输入流）上的类集合行为。

这些算法顺序处理元素，因此要求输入迭代器。他们还接受一个表示目的序列的输出迭代器，唯一的例外是 `includes`。这些算法返回递增后的 `dest` 迭代器，表示写入 `dest` 的最后一个元素之后的位置。

每种算法都有重载版本，第一个使用元素类型的`<`运算符，第二个使用给定的比较操作。

```
includes(beg, end, beg2, end2)
includes(beg, end, beg2, end2, comp)
```

如果第二个序列中每个元素都包含在输入序列中，则返回 `true`。否则返回 `false`。

```
set_union(beg, end, beg2, end2, dest)
set_union(beg, end, beg2, end2, dest, comp)
```

对两个序列中的所有元素，创建它们的有序序列。两个序列都包含的元素在输出序列中只出现一次。输出序列保存在 `dest` 中。

```
set_intersection(beg, end, beg2, end2, dest)
set_intersection(beg, end, beg2, end2, dest, comp)
```

对两个序列都包含的元素创建一个有序序列。结果序列保存在 `dest` 中。

```
set_difference(beg, end, beg2, end2, dest)
set_difference(beg, end, beg2, end2, dest, comp)
```

对出现在第一个序列中，但不在第二个序列中的元素，创建一个有序序列。

```
set_symmetric_difference(beg, end, beg2, end2, dest)
set_symmetric_difference(beg, end, beg2, end2, dest, comp)
```

对只出现在一个序列中的元素，创建一个有序序列。

A.2.9 最小值和最大值

这些算法使用元素类型的`<`运算符或给定的比较操作。第一组算法对值而非序列进行操作。第二组算法接受一个序列，它们要求输入迭代器。

```
min(val1, val2)
min(val1, val2, comp)
min(initializer_list)
min(initializer_list, comp)
max(val1, val2)
max(val1, val2, comp)
max(initializer_list)
max(initializer_list, comp)
```

881

返回 `val1` 和 `val2` 中的最小值/最大值，或 `initializer_list` 中的最小值/最大值。两个实参的类型必须完全一致。参数和返回类型都是 `const` 的引用，意味着对象不会被拷贝。

```
minmax(val1, val2)
minmax(val1, val2, comp)
minmax(initializer_list)
minmax(initializer_list, comp)
```

返回一个 `pair`(参见 11.2.3 节, 第 379 页), 其 `first` 成员为提供的值中的较小者, `second` 成员为较大者。`initializer_list` 版本返回一个 `pair`, 其 `first` 成员为 `list` 中的最小值, `second` 为最大值。

```
min_element(beg, end)
min_element(beg, end, comp)
max_element(beg, end)
max_element(beg, end, comp)
minmax_element(beg, end)
minmax_element(beg, end, comp)
```

`min_element` 和 `max_element` 分别返回指向输入序列中最小和最大元素的迭代器。`minmax_element` 返回一个 `pair`, 其 `first` 成员为最小元素, `second` 成员为最大元素。

字典序比较

此算法比较两个序列, 根据第一对不相等的元素的相对大小来返回结果。算法使用元素类型的<运算符或给定的比较操作。两个序列都要求用输入迭代器给出。

```
lexicographical_compare(beg1, end1, beg2, end2)
lexicographical_compare(beg1, end1, beg2, end2, comp)
```

如果第一个序列在字典序中小于第二个序列, 则返回 `true`。否则, 返回 `false`。如果一个序列比另一个短, 且所有元素都与较长序列的对应元素相等, 则较短序列在字典序中更小。如果序列长度相等, 且对应元素都相等, 则在字典序中任何一个都不大于另外一个。

A.2.10 数值算法

数值算法定义在头文件 `numeric` 中。这些算法要求输入迭代器; 如果算法输出数据, 则使用输出迭代器表示目的位置。

882 > `accumulate(beg, end, init)`
`accumulate(beg, end, init, binaryOp)`

返回输入序列中所有值的和。和的初值从 `init` 指定的值开始。返回类型与 `init` 的类型相同。第一个版本使用元素类型的+运算符, 第二个版本使用指定的二元操作。

```
inner_product(beg1, end1, beg2, init)
inner_product(beg1, end1, beg2, init, binOp1, binOp2)
```

返回两个序列的内积, 即, 对应元素的积的和。两个序列一起处理, 来自两个序列的元素相乘, 乘积被累加起来。和的初值由 `init` 指定, `init` 的类型确定了返回类型。

第一个版本使用元素类型的乘法 (*) 和加法 (+) 运算符。第二个版本使用给定的二元操作, 使用第一个操作代替加法, 第二个操作代替乘法。

```
partial_sum(beg, end, dest)
partial_sum(beg, end, dest, binaryOp)
```

将新序列写入 `dest`, 每个新元素的值都等于输入范围中当前位置和之前位置上所有元素之和。第一个版本使用元素类型的+运算符; 第二个版本使用指定的二元操作。算法返回递增后的 `dest` 迭代器, 指向最后一个写入元素之后的位置。

```
adjacent_difference(beg, end, dest)
adjacent_difference(beg, end, dest, binaryOp)
```

将新序列写入 `dest`, 每个新元素(除了首元素之外)的值都等于输入范围中当前位置和前一个位置元素之差。第一个版本使用元素类型的-运算符, 第二个版本使用指定的二元操作。

```
iota(beg, end, val)
```

将 `val` 赋予首元素并递增 `val`。将递增后的值赋予下一个元素, 继续递增 `val`, 然后将递增后的值赋予序列中的下一个元素。继续递增 `val` 并将其新值赋予输入序列中的后续元素。

A.3 随机数

标准库定义了一组随机数引擎类和适配器，使用不同数学方法生成伪随机数。标准库还定义了一组分布模板，根据不同的概率分布生成随机数。引擎和分布类型的名字都与它们的数学性质相对应。

这些类如何生成随机数的细节已经大大超出了本书的范围。在本节中，我们将列出这些引擎和分布类型，但读者需要查询其他资料来学习如何使用这些类型。

A.3.1 随机数分布

883

除了总是生成 `bool` 类型的 `bernoulli_distribution` 外，其他分布类型都是模板。每个模板都接受单个类型参数，它指出了分布生成的结果类型。

分布类与我们已经用过的其他类模板不同，它们限制了我们可以为模板类型指定哪些类型。一些分布模板只能用来生成浮点数，而其他模板只能用来生成整数。

在下面的描述中，我们通过将类型说明为 `template_name<RealT>` 来指出分布生成浮点数。对这些模板，我们可以用 `float`、`double` 或 `long double` 代替 `RealT`。类似的，`IntT` 表示要求一个内置整型类型，但不包括 `bool` 类型或任何 `char` 类型。可以用来代替 `IntT` 的类型是 `short`、`int`、`long`、`long long`、`unsigned short`、`unsigned int`、`unsigned long` 或 `unsigned long long`。

分布模板定义了一个默认模板类型参数（参见 17.4.2 节，第 664 页）。整型分布的默认参数是 `int`，生成浮点数的模板的默认参数是 `double`。

每个分布的构造函数都有这种分布特定的参数。某些参数指出了分布的范围。这些范围与迭代器范围不同，都是包含的。

均匀分布

```
uniform_int_distribution<IntT> u(m, n);  
uniform_real_distribution<RealT> u(x, y);
```

生成指定类型的，在给定包含范围内的值。`m`（或 `x`）是可以返回的最小值；`n`（或 `y`）是最大值。`m` 默认为 0；`n` 默认为类型 `IntT` 对象可以表示的最大值。`x` 默认为 0.0，`y` 默认为 1.0。

伯努利分布

```
bernoulli_distribution b(p);
```

以给定概率 `p` 生成 `true`；`p` 的默认值为 0.5。

```
binomial_distribution<IntT> b(t, p);
```

分布是按采样大小为整型值 `t`，概率为 `p` 生成的；`t` 的默认值为 1，`p` 的默认值为 0.5。

```
geometric_distribution<IntT> g(p);
```

每次试验成功的概率为 `p`；`p` 的默认值为 0.5。

```
negative_binomial_distribution<IntT> nb(k, p);
```

`k`（整型值）次试验成功的概率为 `p`；`k` 的默认值为 1，`p` 的默认值为 0.5。

泊松分布

`poisson_distribution<IntT> p(x);`

均值为 double 值 x 的分布。

884 `exponential_distribution<RealT> e(lam);`

指数分布，参数 lambda 通过浮点值 lam 给出；lam 的默认值为 1.0。

`gamma_distribution<RealT> g(a, b);`

alpha（形状参数）为 a, beta（尺度参数）为 b；两者的默认值均为 1.0。

`weibull_distribution<RealT> w(a, b);`

形状参数为 a, 尺度参数为 b 的分布；两者的默认值均为 1.0。

`extreme_value_distribution<RealT> e(a, b);`

a 的默认值为 0.0, b 的默认值为 1.0。

正态分布

`normal_distribution<RealT> n(m, s);`

均值为 m, 标准差为 s; m 的默认值为 0.0, s 的默认值为 1.0。

`lognormal_distribution<RealT> ln(m, s);`

均值为 m, 标准差为 s; m 的默认值为 0.0, s 的默认值为 1.0。

`chi_squared_distribution<RealT> c(x);`

自由度为 x; 默认值为 1.0。

`cauchy_distribution<RealT> c(a, b);`

位置参数 a 和尺度参数 b 的默认值分别为 0.0 和 1.0。

`fisher_f_distribution<RealT> f(m, n);`

自由度为 m 和 n; 默认值均为 1。

`student_t_distribution<RealT> s(n);`

自由度为 n; n 的默认值均为 1。

抽样分布

`discrete_distribution<IntT> d(i, j);`

`discrete_distribution<IntT> d(il);`

i 和 j 是一个权重序列的输入迭代器, il 是一个权重的花括号列表。权重必须能转换为 double。

`piecewise_constant_distribution<RealT> pc(b, e, w);`

b、e 和 w 是输入迭代器。

`piecewise_linear_distribution<RealT> pl(b, e, w);`

b、e 和 w 是输入迭代器。

A.3.2 随机数引擎

标准库定义了三个类，实现了不同的算法来生成随机数。标准库还定义了三个适配器，可以修改给定引擎生成的序列。引擎和引擎适配器类都是模板。与分布的参数不同，这些引擎的参数更为复杂，且需深入了解特定引擎使用的数学知识。我们在这里列出所有引擎，以便读者对它们有所了解，但介绍如何生成这些类型超出了本书的范围。885

标准库还定义了几个从引擎和适配器类型构造的类型。`default_random_engine` 类型是一个参数化的引擎类型的类型别名，参数化所用的变量的目的是在通常情况下获得好的性能。标准库还定义了几个类，它们都是一个引擎或适配器的完全特例化版本。标准库定义的引擎和特例化版本如下：

`default_random_engine`

某个其他引擎类型的类型别名，目的是用于大多数情况。

`linear_congruential_engine`

`minstd_rand0` 的乘数为 16807，模为 2147483647，增量为 0。

`minstd_rand` 的乘数为 48271，模为 2147483647，增量为 0。

`mersenne_twister_engine`

`mt19937` 为 32 位无符号梅森旋转生成器。

`mt19937_64` 为 64 位无符号梅森旋转生成器。

`subtract_with_carry_engine`

`ranlux24_base` 为 32 位无符号借位减法生成器。

`ranlux48_base` 为 64 位无符号借位减法生成器。

`discard_block_engine`

引擎适配器，将其底层引擎的结果丢弃。用要使用的底层引擎、块大小和旧块大小来参数化。

`ranlux24` 使用 `ranlux24_base` 引擎，块大小为 223，旧块大小为 23。

`ranlux48` 使用 `ranlux48_base` 引擎，块大小为 389，旧块大小为 11。

`independent_bits_engine`

引擎适配器，生成指定位数的随机数。用要使用的底层引擎、结果的位数以及保存生成的二进制位的无符号整型类型来参数化。指定的位数必须小于指定的无符号类型所能保存的位数。

`shuffle_order_engine`

引擎适配器，返回的就是底层引擎生成的数，但返回的顺序不同。用要使用的底层引擎和要混洗的元素数目来参数化。

`knuth_b` 使用 `minstd_rand0` 和表大小 256。

索引

粗体页码指的是第一次定义该术语的页码，斜体页码指的是各章“术语表”定义该术语的页码。

C++11 的新特性

= default, 237, 449
= delete, 449
allocator, construct forwards to anyconstructor
(allocator, construct 转到任意构造函数), 428
array container (array容器), 292
auto, 61
 for type abbreviation (为类型缩写), 79, 115
 not with dynamic array (不能用于动态数组), 424
 with dynamic object (可用于动态对象), 408
begin function (begin函数), 106
bind function (bind函数), 354
bitset enhancements (bitset增强功能), 643
constexpr
 constructor (构造函数), 267
 function (函数), 214
 variable (变量), 59
container (容器)
 cbegin and cend (cbegin和cend), 98, 299
 emplace members (emplace成员), 308
 insert return type (insert返回类型), 308
 nonmember swap (非成员swap), 303
 of container (容器的), 87, 294
 shrink_to_fit, 318
decltype, 62
 function return type (函数返回类型), 223
delegating constructor (委托构造函数), 261
deleted copy-control (删除的拷贝控制), 553
division rounding (除法取整), 125
end function (endi函数), 106
enumeration (枚举)
 controlling representation (控制表示形式), 738
 forward declaration (前置声明), 738
 scoped (限定作用域的), 736
explicit conversion operator (显式的类型转换运算符), 516
explicit instantiation (显式实例化), 597
final class (final类), 533
format control for floating-point (浮点数格式化控制),
 670
forward function (forward函数), 614
forward_list container (forward_list容器),
 292
function interface to callable objects (function可
 调用对象接口), 512
in-class initializer (类内初始值), 65, 246
inherited constructor (继承的构造函数), 557, 712
initializer_list, 197
inline namespace (内联命名空间), 699
lambda expression (lambda表达式), 346
list initialization (列表初始化)
 = (assignment) (赋值), 129
 container (容器), 299, 376
 dynamic array (动态数组), 424
 dynamic object (动态对象), 407
 pair, 384
 return value (返回值), 203, 380
 variable (变量), 38
 vector, 87
long long, 30
mem_fn function (mem_fn函数), 746
move function (move函数), 472
move avoids copies (移动避免拷贝), 469
move constructor (移动构造函数), 473
move iterator (移动迭代器), 480
move-enabled this pointer (可移动this指针), 483
noexcept
 exception specification (异常说明), 473, 690
 operator (运算符), 691
nullptr, 48
random-number library (随机数库), 660
range for statement (范围for语句), 82, 168
 not with dynamic array (不能用于动态数组), 424
regular expression-library (正则表达式库), 645
rvalue reference (右值引用), 471
 cast from lvalue (左值类型转换), 612
 reference collapsing (引用折叠), 609
sizeof data member (sizeof数据成员), 139
sizeof... operator (sizeof运算符), 619
smart pointer (智能指针), 400
 shared_ptr, 400
 unique_ptr, 417
 weak_ptr, 420
string

- numeric conversions (数值转换), 327
 parameter with IO types (输入输出类型的形参), 284
template (模板)
 function template default template argument (函数模板默认模板实参), 594
 type alias (类型别名), 590
 type parameter as friend (类型参数作为友元), 590
 variadic (可变参数), 618
 varadic and forwarding (可变参数与转发), 622
 trailingreturn type (尾置返回类型), 206
 in **function template** (在函数模板中), 605
 in **lambda expression** (在lambda表达式中), 354
tuple, 636
type alias declaration (类型别名声明), 60
union member of class type (类类型的联合成员), 750
unordered containers (无序容器), 394
virtual function (虚函数)
 final, 538
 override, 530, 538
- ## Symbols
- ...** (ellipsis parameter) (省略符形参), 199
`/* */` (block comment) (块注释), 8, 23
`//` (single-line comment) (单行注释), 8, 23
`= default`, 237, 274
 copy-control members (拷贝控制成员), 448
 default constructor (默认构造函数), 237
`= delete`, 449
 copy control (拷贝控制), 449–450
 default constructor (默认构造函数), 449
 function matching (函数匹配), 450
 move operations (移动操作), 475
`_ _DATE_ _`, 216
`_ _FILE_ _`, 216
`_ _LINE_ _`, 216
`_ _TIME_ _`, 216
`_ _cplusplus`, 760
`\0` (null character) (空字符), 36
`\Xnnn` (hexadecimal escape sequence) (十六进制转义序列), 36
`\nnn` (octal escape sequence) (八进制转义序列), 36
`{ }` (curly brace) (花括号), 2, 23
`#include`, 6, 25
 standard header (标准库头文件), 6
 user-defined header (用户定义的头文件), 16
`#define`, 68, 71
`#endif`, 68, 71
`#ifndef`, 68, 71
#ifndef, 68, 71
`-classname` (~类名), 参见 **destructor**
`;` (semicolon) (分号), 3
 class definition (类定义), 65
 null statement (空语句), 154
`++` (increment) (递增运算符), 11, 25, 131–133, 150
 iterator (迭代器), 95, 118
 overloaded operator (重载运算符), 501–504
 pointer (指针), 105
 precedence and associativity (优先级和结合律), 132
 reverse iterator (反向迭代器), 363
 StrBlobPtr, 502
`--` (decrement) (递减运算符), 11, 25, 131–133, 151
 iterator (迭代器), 95
 overloaded operator (重载运算符), 501–504
 pointer (指针), 105
 precedence and associativity (优先级和结合律), 132
 reverse iterator (反向迭代器), 363, 364
 StrBlobPtr, 502
`* (dereference)` (解引用), 48, 71, 398
 iterator (迭代器), 95
 map iterators (map迭代器), 382
 overloaded operator (重载运算符), 504
 pointer (指针), 48
 precedence and associativity (优先级和结合律), 132
 smart pointer (智能指针), 400
 StrBlobPtr, 504
`& (address-of)` (取地址符), 47, 71
 overloaded operator (重载运算符), 491
`->` (arrow operator) (箭头运算符), 98, 118, 133
 overloaded operator (重载运算符), 504
 StrBlobPtr, 502
`. (dot)` (点运算符), 20, 25, 133
`->*` (pointer to member arrow) (成员指针箭头运算符), 740
`.*` (pointer to member dot) (成员指针点运算符), 740
`[](subscript)` (下标), 84
 array (数组), 101, 118
 array, 310
 bitset, 644
 deque, 310
 does not add elements (不添加元素), 93
 map and unordered_map (map和unordered_map), 387, 398
 adds element (添加元素), 387
 multidimensional array (多维数组), 113
 out-of-range index (越界索引), 84
 overloaded operator (重载运算符), 500
 pointer (指针), 108

string, 84, 118, 310
StrVec, 501
subscript range (下标范围), 85
vector, 92, 118, 310
(-) (call operator) (调用运算符), 21, 25, 182, 226
absInt, 506
const member function (const成员函数), 508
execution flow (执行流程), 183
overloaded operator (重载运算符), 506
PrintString, 507
ShorterString, 508
SizeComp, 508
:: (scope operator) (作用域运算符), 7, 25, 74
base-class member (基类成员), 539
class type member (类类型成员), 79, 253
container, type members (容器, 类型成员), 298
global namespace (全局命名空间), 698, 723
member function, definition(成员函数, 定义), 232
overrides name lookup (覆盖名字查找), 256
= (assignment) (赋值), 10, 25, 129–131
 see also copy assignment (参见“拷贝赋值”)
 see also move assignment (参见“移动赋值”)
associativity (结合律), 129
base from derived (由派生类对象构建的基类对象), 535
container (容器), 80, 92, 301
conversion (类型转换), 129, 141
derived class (派生类), 555
in condition (在条件中), 130
initializer_list, 499
list initialization (列表初始化), 129
low precedence (低优先级), 130
multiple inheritance (多重继承), 712
overloaded operator (重载运算符), 443, 499
pointer (指针), 49
to signed (指向signed的), 33
to unsigned (指向unsigned的), 33
vs. == (equality) (对比相等运算符), 130
vs. initialization (对比初始化), 39
+= (compound assignment) (复合赋值), 11, 25, 131
 bitwise operators (位运算符), 137
 iterator (迭代器), 99
 overloaded operator (重载运算符), 492, 497
 Sales_data, 500
 exception version (异常版本), 694
 string, 80
+ (addition) (加法), 5, 125
 iterator (迭代器), 99
 pointer (指针), 106
 Sales_data, 497
 exception version (异常版本), 694
 Sales_item, 19
SmallInt, 522
string, 80
- (subtraction) (减法), 125
 iterator (迭代器), 99
 pointer (指针), 106
* (multiplication) (乘法), 125
/ (division) (除法), 125
 rounding (取整), 125
% (modulus) (取模), 125
grading program (成绩程序), 157
== (equality) (相等), 16, 25
arithmetic conversion (算术类型转换), 128
container (容器), 80, 92, 304, 305

chained-input (链式输入), 7
 istream, 7
 istream_iterator, 359
 overloaded operator (重载运算符), 495–496
 precedence and associativity (优先级和结合律), 137
 Sales_data, 495
 Sales_item, 18
 string, 76, 118
 << (output operator) (输出运算符), 6, 25
 bitset, 644
 chained output (链式输出), 6
 ostream, 6
 ostream_iterator, 361
 overloaded operator (重载运算符), 494–495
 precedence and associativity (优先级和结合律), 137
 Query, 568
 Sales_data, 494
 Sales_item, 18
 string, 77, 118
 >> (right-shift) (右移), 136, 151
 << (left-shift) (左移), 136, 151
 && (logical AND) (逻辑与), 85, 118, 126, 150
 order of evaluation (求值顺序), 123
 overloaded operator (重载运算符), 491
 short-circuit evaluation (短路求值), 126
 || (logical OR) (逻辑或), 126
 order of evaluation (求值顺序), 123
 overloaded operator (重载运算符), 491
 short-circuit evaluation (短路求值), 126
& (bitwise AND) (位与), 137, 150
 Query, 565
! (logical NOT) (逻辑非), 79, 118, 127, 151
|| (logical OR) (逻辑或), 118, 150
| (bitwise OR) (位或), 137, 150
 Query, 565
^ (bitwise XOR) (位异或), 137, 150
~ (bitwise NOT) (位求反), 137, 150
 Query, 565, 569
, (comma operator) (逗号运算符), 140, 150
 order of evaluation (求值顺序), 123
 overloaded operator (重载运算符), 491
?: (conditional operator) (条件运算符), 120, 150
 order of evaluation (求值顺序), 123
 precedence and associativity (优先级和结合律), 135
+ (unary plus) (一元加法), 125
- (unary minus) (一元减法), 125
L'c' (wchar_t literal) (wchar_t字面值常量), 37
ddd.dddL or ddd.ddd1 (long double literal) (long double字面值常量), 37
numEnum or numenum (double literal) (double字面值常量), 37
numF or numf (float literal) (float字面值常量), 37
numL or numl (long literal) (long字面值常量), 37
numLL or numll (long long literal) (long long字面值常量), 37
numU or numu (unsigned literal) (unsigned字面值常量), 37
class member:constant expression (类成员: 常量表达式), 参见bitfield

A

absInt, 506
() (call operator) (调用运算符), 506
abstract base class (抽象基类), 541, 575
 BinaryQuery, 570
 Disc_quote, 541
 Query_base, 564
abstract data type (抽象数据类型), 228, 273
access control (访问控制), 542–546
 class derivation list (类派生列表), 529
 default inheritance access (默认继承访问), 546
 default member access (默认成员访问), 240
 derived class (派生类), 544
 derived-to-base conversion (派生类向基类的类型转换), 544
 design (设计), 544
 inherited members (继承的成员), 543
 local class (局部类), 755
 nested class (嵌套类), 747
 private, 240
 protected, 529, 542
 public, 240
 using declaration (using声明), 545
access specifier (访问说明符), 240, 273
accessible (可访问的), 542, 575
 derived-to-base conversion (派生类向基类的类型转换), 544
Account, 269
accumulate, 338, 780
 bookstore program (书店程序), 362
Action, 742
adaptor (适配器), 332
 back_inserter, 358
 container (容器), 329, 329–330
 front_inserter, 358
 inserter, 358
 make_move_iterator, 480
add, Sales_data, 234
add_item, Basket, 561
add_to_Folder, Message, 462

- address (地址), 31, 69
`adjacent_difference`, 780
`adjacent_find`, 771
advice (建议)
 - always initialize a pointer (总是初始化指针), 48
 - avoid casts (避免类型转换), 146
 - avoid undefined behavior (避免未定义的行为), 33
 - choosing a built-in type (选择内置类型), 32
 - define small utility functions (定义小功能集函数), 248
 - define variables near first use (在邻近第一次使用时再定义变量), 44
 - don't create unnecessary `regex` objects (不要创建不必要的`regex`对象), 649
 - forwarding parameter pattern (转发形参模式), 624
 - keep lambda captures simple (保持lambda的变量捕获简单化), 351
 - managing iterators (管理迭代器), 296, 315
 - prefix vs. postfix operators (前置运算符与后置运算符), 132
 - rule of five (五个拷贝控制成员的规则), 478
 - use move sparingly (谨慎使用move), 481
 - use constructor initializer lists (使用构造函数初始值列表), 259
 - when to use overloading (何时使用重载), 208
 - writing compound expressions (编写复合表达式), 124

aggregate class (聚合类), 266, 273
 - initialization (初始化), 266

algorithm header (`algorithm`头文件), 336

algorithms (算法), 336, 371
 - 参见附录A

architecture (体系结构)
 - `_copy` versions (`_copy`版本), 342, 369
 - `_if` versions (`_if`版本), 368
 - naming convention (命名规范), 368–369
 - operate on iterators not containers (操作迭代器而非容器), 337
 - overloading pattern (重载模式), 368
 - parameter pattern (形参模式), 367–368
 - read-only (只读), 338–339
 - reorder elements (重排序元素), 342–343, 369
 - write elements (写元素), 339–342

associative container and (关联容器与), 382

bind as argument (bind作为实参), 354

can't change container size (不能改变容器大小), 343

element type requirements (元素类型需求), 337

function object arguments (函数对象实参), 507

`istream_iterator`, 360

iterator category (迭代器类别), 365–367

iterator range (迭代器范围), 336

lambda as argument (`lambda`作为实参), 348, 353

library function object (标准库函数对象), 509

`ostream_iterator`, 360

sort comparison, requires strict weak ordering (排序所需比较操作, 要求严格弱序), 378

supplying comparison operation (支持比较操作), 344, 368
 - function (函数), 344
 - `lambda`, 346, 347

two input ranges (两个输入范围), 368

type independence (类型独立), 337

use element's == (equality) (使用元素的相等运算符), 343, 368

use element's < (less-than) (使用元素的小于运算符), 343, 368

`accumulate`, 338
 - bookstore program (书店程序), 362

`copy`, 341

`count`, 337

`equal_range`, 639

`equal`, 340

`fill_n`, 340

`fill`, 340

`find_if`, 346, 354, 368

`find`, 336

`for_each`, 348

`replace_copy`, 342

`replace`, 342

`set_intersection`, 573

`sort`, 343

`stable_sort`, 345

`transform`, 353

`unique`, 343

alias declaration (别名声明)
 - namespace (命名空间), 701, 723
 - template type (模板类型), 590
 - type (类型), 60

`all_of`, 771

`alloc_n_copy`, `StrVec`, 467

`allocate`, `allocator`, 427

`allocator`, 427, 427–429, 436, 464–470
 - `allocate`, 427, 467
 - compared to `operator new` (对比`operator new`), 729

`construct`, 428
 - forwards to constructor (转到构造函数), 467

`deallocate`, 429, 467
 - compared to `operator delete` (对比`operator delete`), 729

`destroy`, 428, 467

alternative operator name (可选择的运算符名字), 42

`alternative_sum`, `program` (`alternative_sum`)

- 程序), 604
ambiguous (二义性)
 conversion (类型转换), 516–522
 - multiple inheritance (多重继承), 713
 - function call (函数调用), 209, 219, 225
 - multiple inheritance (多重继承), 715
 - overloaded operator (重载运算符), 521**AndQuery**, 564
 - class definition (类定义), 570
 - eval** function (**eval**函数), 572**anonymous union** (匿名联合), 750, 762
any, **bitset**, 643
any_of, 771
app (file mode) (文件模式), 286
append, **string**, 323
argc, 197
argument (实参), 21, 23, 182, 225
 - array (数组), 192–197
 - buffer overflow (缓冲溢出), 193
 - to pointer conversion (指针转换的), 193
 - C-style string (C风格字符串), 194
 - conversion, function matching (类型转换, 函数匹配), 209
 - default (默认), 211
 - forwarding (转发), 622
 - initializes parameter (初始化形参), 183
 - iterator (迭代器), 194
 - low-level **const** (底层**const**), 191
 - main function (main函数), 196
 - multidimensional array (多维数组), 195
 - nonreference parameter (非引用形参), 188
 - pass by reference (引用传递), 189, 226
 - pass by value (值传递), 188, 226
 - uses copy constructor (使用拷贝构造函数), 441
 - uses move constructor (使用移动构造函数), 476, 478
 - passing (传递), 187–190
 - pointer (指针), 193
 - reference parameter (引用形参), 189, 192
 - reference to **const** (常量引用), 189
 - top-level **const** (顶层**const**), 190**argument list** (实参列表), 182
argument-dependent lookup (实参相关的查找), 706
 - move和forward, 707**argv**, 197
arithmetic (算术)
 conversion (类型转换), 32, 141, 149
 - in equality and relational operators (在相等性运算符和关系运算符中), 128
 integral promotion (整型提升), 142, 149
 - signed to unsigned (signed转为unsigned), 32
 to **bool** (转换为**bool**类型), 144
operators (运算符), 124
 - compound assignment (e.g., +=) (复合赋值, 如 +=), 131
 - function object (函数对象), 509
 - overloaded (重载), 496**type** (类型), 30, 69
 - machine-dependent (机器相关的), 30**arithmetic** (addition and subtraction) (算术, 加法和减法)
 - iterators (迭代器), 99, 117
 - pointers (指针), 106, 118**array** (数组), 101–116
 - [](subscript) (下标), 104, 118
 - argument and parameter (实参和形参), 192–197
 - argument conversion (实参类型转换), 193
 - auto returns pointer (auto返回指针), 105
 - begin function (begin函数), 106
 - compound type (复合类型), 101
 - conversion to pointer (转换为指针), 105, 143
 - function arguments (函数实参), 192
 - template argument deduction (模板实参推断), 601
 - decltype** returns array type (**decltype**返回数组类型), 105
 - definition (定义), 101
 - dimension, constant expression (维度, 常量表达式), 101
 - dynamically allocated (动态分配), 423, 423–429
 - allocator, 427
 - can't use begin and end (不能调用begin和end), 424
 - can't use range for statement (无法使用范围for语句), 424
 - delete [], 425
 - empty array (空数组), 424
 - new [], 424
 - shared_ptr, 426
 - unique_ptr, 425
 - elements and destructor (元素与析构函数), 445
 - end function (end函数), 106
 - initialization (初始化), 102
 - initializer of vector (vector的初始值), 111
 - multidimensional (多维的), 111–116
 - no copy or assign (不允许拷贝和赋值), 102
 - of char initialization (字符数组初始化), 102
 - parameter (形参)
 - buffer overflow (缓冲溢出), 193
 - converted to pointer (转换为指针), 193
 - function template (函数模板), 579
 - pointer to (指向……), 196
 - reference to (引用……), 195

- return type (返回类型), 184
trailing (尾置), 206
type alias (类型别名), 206
decltype, 206
sizeof, 139
subscript range (下标范围), 104
subscript type (下标类型), 103
understanding complicated declarations (理解复杂声明), 102
- array**
see also container (参见“容器”)
see also sequential container (参见“顺序容器”)
[] (subscript) (下标), 310
= (assignment) (赋值), 302
assign, 302
copy initialization (拷贝初始化), 301
default initialization (默认初始化), 301
definition (定义), 301
header (头文件), 294
initialization (初始化), 299–301
list initialization (列表初始化), 301
overview (概览), 293
random-access iterator (随机访问迭代器), 367
swap, 303
- assert** preprocessor macro (assert预处理宏), 215, 225
- assign**
array, 302
invalidates iterator (令迭代器失效), 302
sequential container (顺序容器), 302
string, 322
- assignment, vs. initialization (赋值对比初始化), 39, 258
assignment operators (赋值运算符), 129–131
associative array (关联数组), *see map* (参见map)
associative container (关联容器), 374, 397
and library algorithms (和标准库算法), 383
initialization (初始化), 377
key_type requirements (key_type需求), 378, 396
members (成员)
begin, 382
count, 388, 389
emplace, 384
end, 382
equal_range, 390
erase, 386
find, 388, 389
insert, 384
key_type, 381, 397
mapped_type, 381, 397
value_type, 381, 398
- override default comparison (覆盖默认比较), 379
override default hash (覆盖默认哈希), 396
overview (概览), 376
- associativity** (结合律), 120, 121–123, 149
= (assignment) (赋值), 129
?: (conditional operator) (条件运算符), 134
dot and dereference (点运算符和解引用运算符), 134
increment and dereference (递增和递减运算符), 132
IO operator (输入输出运算符), 138
overloaded operator (重载运算符), 491
- at**
- deque, 310
map, 387
string, 310
unordered_map, 387
vector, 310
- ate** (file mode) (文件模式), 286
- auto**, 61, 69
cbegin, 97, 339
cend, 97, 339
for type abbreviation (为类型缩写), 79, 115
of array (数组的), 105
of reference (引用的), 61
pointer to function (函数指针), 222
with new (与new), 407
- auto_ptr** deprecated (auto_ptr已弃用), 419
- automatic object** (自动对象), 185, 225
see also local variable (参见“局部变量”)
see also parameter (参见“形参”)
and destructor (和析构函数), 445
- avg_price, Sales_data**, 232
- B**
- back**
queue, 330
sequential container (顺序容器), 309
StrBlob, 406
- back_inserter**, 341, 358, 371
requires push_back (需要push_back), 341, 358
- bad**, 280
bad_alloc, 176, 408
bad_cast, 176, 731
bad_typeid, 733
badbit, 279
- base, reverse iterator** (反向迭代器), 364
- base class** (基类), 526, 575
see also virtual function (参见“虚函数”)
abstract (抽象), 541, 575
- base-to-derived conversion, not automatic** (派生类)

向基类的转换, 非自动), 534
can be a derived class (可以作为派生类), 533
definition (定义), 527
derived-to-base conversion (派生类向基类的转换), 530
accessibility (可访问性), 544
key concepts (关键概念), 537
multiple inheritance (多重继承), 712
final, 533
friendship not inherited (友元关系不能继承), 545
initialized or assigned from derived (用派生类初始化或赋值), 535
member hidden by derived (派生类隐藏的成员), 549
member new and delete (作为成员的new和delete), 727
multiple, see multiple inheritance (多重, 参见“多重继承”)
must be complete type (必须是完全类型), 533
protected member (受保护的成员), 542
scope (作用域), 547
 inheritance (继承), 547–551
 multiple inheritance (多重继承), 715
 virtual function (虚函数), 550
static members (静态成员), 532
user of (……的用户), 544
virtual, see virtual base class (虚, 参见“虚基类”)
virtual destructor (虚析构函数), 552

Basket, 559
 add_item, 561
 total, 560

Bear, 710
 virtual base class (虚基类), 718

before_begin, forward_list, 313

begin
 associative container (关联容器), 382
 container (容器), 95, 117, 298, 332
 function (函数), 106, 117
 not with dynamic array (不能用于动态数组), 424
 multidimensional array (多维数组), 115
 StrBlob, 422
 StrVec, 465

bernoulli_distribution, 665

best match (最佳匹配), 209, 225
 see also function matching (参见“函数匹配”)

bidirectional iterator (双向迭代器), 366, 371

biggies program (biggies程序), 348

binary (file mode) (文件模式), 286

binary operators (二元运算符), 120, 149
 overloaded operator (重载运算符), 490

binary predicate (二元谓词), 344, 371

binary_function deprecated (**binary_function** 已弃用), 513

binary_search, 773

BinaryQuery, 564
 abstract base class (抽象基类), 570

bind, 354, 371
 check_size, 355
 generates callable object (生成可调用对象), 354
 from pointer to member (从成员指针), 746
 placeholders, 355
 reference parameter (引用形参), 356

bind1st deprecated (**bind1st**已弃用), 357

bind2nd deprecated (**bind2nd**已弃用), 357

binops desk calculator (**binops**桌面计算器), 512

bit-field (位域), 756, 762
 access to (访问), 756
 constant expression (常量表达式), 756

bitset, 641, 641–645, 680
 [](subscript) (下标), 644
 <<(output operator) (输出运算符), 644
 any, 643
 count, 644
 flip, 644
 grading program (成绩程序), 644
 header (头文件), 640
 initialization (初始化), 641–642
 from **string** (从**string**), 642
 from **unsigned** (从**unsigned**), 641

none, 643
 reset, 644
 set, 644
 test, 644
 to_ulong, 644

bitwise, bitset, operators(位, **bitset**, 运算符), 643

bitwise operators (位运算符), 135–138
 += (compound assignment) (复合赋值), 138
 compound assignment (e.g., **+=**) (复合赋值, 如**+=**), 131
 grading program (成绩程序), 137
 operand requirements (运算对象需求), 136

Blob
 class template (类模板), 583
 constructor (构造函数), 586
 initializer_list, 587
 iterator parameters (迭代器形参), 596
 instantiation (实例化), 584
 member functions (成员函数), 585–587

block (块), 2, 11, 23, 155, 178
 function (函数), 184
 scope (作用域), 44, 70, 155
 try, 173, 174, 179, 724

block /* */, comment (块注释), 8, 23

- book from author program (书籍作者程序), 389–391
bookstore program (书店程序)
 Sales_data, 229
 using algorithms (使用算法), 362
 Sales_item, 21
bool, 30
 conversion (类型转换), 32
 literal (字面值常量), 37
 in condition (在条件中), 128
boolalpha, manipulator (操纵符), 667
brace, curly (花括号), 2, 23
braced list, *see* list initialization (花括号列表, 见列表初始化)
break statement (break语句), 170, 178
 in *switch* (在switch中), 160–162
bucket management, unordered container (桶管理, 无序容器), 394
buffer (缓冲区), 6, 23
 flushing (刷新), 281
buffer overflow (缓冲溢出), 94, 104, 117
 array parameter (数组形参), 193
 C-style string (C风格字符串), 109
buildMap program (*buildMap*程序), 393
built-in type (内置类型), 2, 23, 30–32
 default initialization (默认初始化), 40
Bulk_quote
 class definition (类定义), 530
 constructor (构造函数), 531, 541
 derived from *Disc_quote* (从*Disc_quote*派生), 541
 design (设计), 526
 synthesized copy control (合成拷贝控制), 553
byte (字节), 31, 69
- C**
- .C file (.C文件), 3
.cc file (.cc文件), 3
.cppfile (.cpp文件), 3
.cp file (.cp文件), 3
C library header (C标准库头文件), 82
C-style cast (C风格类型转换), 146
C-style string (C风格字符串), 102, 109, 109–110, 117
 buffer overflow (缓冲溢出), 110
 initialization (初始化), 109
 parameter (形参), 194
 string, 111
c_str, 111
call by reference (传引用调用), 187, 189, 225
call by value (传值调用), 187, 225
 uses copy constructor (使用拷贝构造函数), 441
 uses move constructor (使用移动构造函数), 476
call signature (调用形式), 511, 523
callable object (可调用对象), 346, 371, 506–507
 absInt, 506
 bind, 354
 call signature (调用形式), 511
 function and function pointers (函数和函数指针), 346
function objects (函数对象), 507
pointer to member (成员指针)
 and bind (和bind), 746
 and function (和function), 745
 and mem_fn (和mem_fn), 746
 not callable (不可调用), 745
PrintString, 506
ShorterString, 508
SizeComp, 508
with function (与function), 511–514
with algorithms (与算法), 348
candidate function (候选函数), 217, 225
 see also function matching (参见“函数匹配”)
function template (函数模板), 615
namespace (命名空间), 708
overloaded operator (重载运算符), 521
capacity
 string, 318
 StrVec, 465
 vector, 318
capture list, *see* lambda expression (捕获列表, 参见“lambda表达式”)
case label (case标签), 161, 161–163, 178
 default, 162
 constant expression (常量表达式), 161
case sensitive, string (大小写敏感, string), 325
cassert header (cassert头文件), 215
cast, *see also* named cast (强制类型转换, 参见“命名的强制类型转换”), 149
 checked, *see* dynamic_cast (检查, 参见dynamic_cast)
 old-style (旧式的), 146
 to rvalue reference (右值引用的), 612
catch, 173, 174, 178, 687, 722
 catch(…), 688, 722
 exception declaration (异常声明), 174, 179, 687, 722
 exception object (异常对象), 687
 matching (匹配), 687
 ordering of (……的顺序), 687
 runtime_error, 174
catch all (*catch*(…)) (捕获全部), 688, 722
caution (提示)
 ambiguous conversion operator (二义性转换运算符), 515

conversions to `unsigned` (转换为`unsigned`) , 35
 dynamic memory pitfalls (动态内存使用误区) , 410
 exception safety (异常安全性) , 175
 IO buffers (输入输出缓冲) , 282
 overflow (溢出) , 125
 overloaded operator misuse (误用重载运算符) , 492
 overloaded operators and conversion operators (重载运算符与类型转换运算符) , 519
 smart pointer, pitfalls (智能指针, 误区) , 417
 uninitialized variables (未初始化的变量) , 41
 using directives cause pollution (using指示造成程序污染) , 704
cbegin
 auto, 97, 339
 decltype, 97, 339
 container (容器) , 98, 298, 299, 332
cctype
 functions (函数) , 82–84
 header (头文件) , 82
cend
 auto, 97, 339
 decltype, 97, 339
 container (容器) , 98, 298, 299, 332
cerr, 5, 23
 chained input (链式输入) , 7
 chained output (链式输出) , 6
char, 30
 signed, 32
 unsigned, 32
 array initialization (数组初始化) , 102
 literal (字面值常量) , 36
 representation (表示) , 32
char16_t, 30
char32_t, 30
 character (字符)
 newline (\n) (换行符) , 36
 nonprintable (不可打印的) , 36, 70
 null (\0) (空) , 36
 tab (\t) (制表符) , 36
 character string literal, *see* string literal (字符串型字符串字面值常量, 参见“字符串字面值常量”)
check
 StrBlob, 406
 StrBlobPtr, 421
check_size, 355
 bind, 354
checked cast, *see* dynamic_cast (经检查的强制类型转换, 见dynamic_cast)
 children's story program (孩子的故事程序) , 343–349
chk_n_alloc, StrVec, 465
cin, 5, 23
 tied to cout (绑定到cout) , 282
c1, 4
class (类) , 17, 23, 64, 273
see also constructor (参见“构造函数”)
see also destructor (参见“析构函数”)
see also member function (参见“成员函数”)
see also static member (参见“静态成员”)
 access specifier (访问说明符) , 240
 default (默认) , 240
 private, 240, 274
 public, 240, 274
aggregate (聚合) , 266, 273
assignment operator (赋值运算符)
see copy assignment (参见“拷贝赋值”)
see move assignment (参见“移动赋值”)
base, *see* base class (基类) , 575
data member (数据成员) , 65, 69
 const vs. mutable (const对比mutable) , 245
 const, initialization (const, 初始化) , 259
 in-class initializer (类内初始值) , 246
 initialization (初始化) , 236, 245
 must be complete type (必须是完全类型) , 250
 mutable, 245, 274
 order of destruction (析构顺序) , 445
 order of initialization (初始化顺序) , 259
 pointer, not deleted (指针, 非删除) , 446
 reference, initialization (引用, 初始化) , 259
 sizeof, 139
declaration (声明) , 249, 273
default inheritance specifier (默认继承说明符) , 546
definition (定义) , 64, 230–239
 ends with semicolon (以分号结束) , 65
derived, *see* derived class (派生, 参见“派生类”) , 575
exception, 173, 179
final specifier (final说明符) , 533
forward declaration (前向声明) , 250, 274
friend (友元) , 241, 251
 class (类) , 251
 function (函数) , 241
 member function (成员函数) , 251
 overloaded function (重载函数) , 252
 scope (作用域) , 242, 252
 template class or function (模板类或函数) , 588
implementation (实现) , 228
interface (接口) , 228
literal (字面值常量) , 267
local, *see* local class (局部, 参见“局部类”)

- member (成员), 65, 69
member access (成员访问), 253
member new and delete (作为成员的new和delete), 728
member : *constant expression*, see bitfield (成员:常量表达式, 参见“位域”)
multiple base classes, see multiple inheritance (多重基类, 参见“多重继承”)
name lookup (名字查找), 254
nested, see nested class (嵌套, 参见“嵌套类”)
pointer to member, see pointer to member (成员指针)
preventing copies (阻止拷贝), 449
scope (作用域), 65, 253, 253–257, 273
synthesized, copy control (合成, 拷贝控制), 239, 440, 444, 446, 475
template member, see member template (模板成员, 参见“成员模板”)
type member (类型成员), 243
 :: (scope operator) (作用域运算符), 253
user of (……的用户), 228
value-like (类值), 453
without move constructor (不含移动构造函数), 477
class
 compared to typename (对比类型名), 580
 default access specifier (默认访问说明符), 240
 default inheritance specifier (默认继承说明符), 546
 template parameter (模板形参), 579
class derivation list (类派生列表), 530
 access control (访问控制), 543
 default access specifier (默认访问说明符), 546
 direct base class (直接基类), 533
 indirect base class (间接基类), 533
 multiple inheritance (多重继承), 710
 virtual base class (虚基类), 718
class template (类模板), 87, 117, 583, 584, 583–591, 630
 see also template parameter (参见“模板形参”)
 see also instantiation (参见“实例化”)
 Blob, 584
 declaration (声明), 592
 default template argument (默认模板实参), 594
 definition (定义), 583
 error detection (错误检测), 582
 explicit instantiation (显式实例化), 597, 597–598
 explicit template argument (显式模板实参), 584
 friend (友元), 588
 all instantiations (全部实例化), 589
 declaration dependencies (声明依赖性), 589
 same instantiation (相同实例化), 588
 specific instantiation (特定实例化), 589
instantiation (实例化), 584
member function (成员函数)
 defined outside class body (定义在类的外部), 585
instantiation (实例化), 587
member template, see member template (成员模板)
specialization (特例化), 625, 626–629, 630
 hash<key_type>, 626, 698
 member (成员), 629
 namespace (命名空间), 698
 partial (部分), 628, 630
static member (静态成员), 590
accessed through an instantiation (通过实例访问), 591
definition (定义), 591
template argument (模板实参), 585
template parameter, used in definition (模板形参, 在定义中使用), 585
type parameter as friend (类型参数作为友元), 590
type-dependent code (类型相关的代码), 583
class type (类类型), 17, 23
conversion (类型转换), 144, 273, 523
ambiguities (二义性), 521
conversion operator (类型转换运算符), 514
converting constructor (类型转换构造函数), 263
impact on function matching (对函数匹配的影响), 517
overloaded function (重载函数), 519
with standard conversion (与标准类型转换), 515
default initialization (默认初始化), 40
initialization (初始化), 65, 76, 235
union member of (union成员), 750
variable vs. function declaration (变量对比函数声明), 263
clear
 sequential container (顺序容器), 312
 stream (流), 280
clog, 5, 23
close, file stream (文件流), 285
cmatch, 649
cmath header (cmath头文件), 664, 669
collapsing rule, reference (折叠规则, 引用), 609
combine, Sales_data, 232
comma (,) operator (逗号运算符), 140
comment (注释), 8, 23
 block /* */ (块), 8, 23
 single-line // (单行), 8, 23
compare
 default template argument (默认模板实参), 594

- function template (函数模板), 578
 - default template argument (默认模板实参), 594
 - explicit template argument (显式模板实参), 604
 - specialization (特例化), 624
 - string literal version (字符串字面值版本), 580
 - template argument deduction (模板实参推断), 602
 - string, 326
- compareIsbn
 - and associative container (和关联容器), 379
 - Sales_data, 345
- compilation (编译)
 - common errors (一般错误), 14
 - compiler options (编译选项), 186
 - conditional (有条件的), 214
 - declaration vs. definition (声明与定义), 40
 - mixing C and C++ (混合C和C++), 760
 - needed when class changes (当类改变时需要), 242
 - templates (模板), 581
 - error detection (错误检测), 582
 - explicit instantiation (显式实例化), 597–598
- compiler (编译器)
 - extension (扩展), 102, 117
 - GNU, 4
 - Microsoft (微软), 4
 - options for separate compilation (单独编译选项), 186
- composition vs. inheritance (组合与继承), 564
- compound assignment (e.g., +=) (复合赋值, 如+=)
 - arithmetic operators (算术运算符), 131
 - bitwise operators (位运算符), 131
- compound expression, *see* expression (复合表达式, 参见“表达式”)
- compound statement (复合语句), 155, 178
- compound type (复合类型), 45, 45–52, 69
 - array (数组), 101
 - declaration style (声明形式), 51
 - understanding complicated declarations (理解复杂声明), 102
- concatenation (连接)
 - string, 80
 - string literal (字符串字面值常量), 36
- condition (条件), 10, 23
 - = (assignment) in (赋值), 130
 - conversion (类型转换), 141
 - do while statement (do while语句), 169
 - for statement (for语句), 11, 166
 - if statement (if语句), 16, 157
 - in IO expression (在输入输出表达式中), 138
 - logical operators (逻辑运算符), 123
- smart pointer as (智能指针作为条件), 400
- stream type as (流类型作为条件), 13, 144, 279
- while statement (while语句), 10, 165
- condition state, IO classes (条件状态, 输入输出类), 279, 290
- conditional compilation (条件编译), 215
- conditional operator (?) (条件运算符), 134
- connection, 416
- console window (控制台窗口), 4
- const, 53, 69
 - and typedef (和typedef), 60
 - conversion (类型转换), 144
 - template argument deduction (模板实参推断), 601
- dynamically allocated (动态分配)
 - destruction (析构), 409
 - initialization (初始化), 408
 - initialization (初始化), 53
 - class type object (类类型对象), 236
- low-level const (底层const), 57
 - argument and parameter (实参与形参), 191
 - conversion from (转换自), 145
 - conversion to (转换为), 144
 - overloaded function (重载函数), 208
 - template argument deduction (模板实参推断), 613
- member function (成员函数), 231, 273
 - () (call operator) (调用运算符), 508
 - not constructors (非构造函数), 235
 - overloaded function (重载函数), 247
 - reference return (引用返回), 247
- parameter (形参), 190
 - function matching (函数匹配), 220
 - overloaded function (重载函数), 208
- pointer (指针), 56, 69
 - pointer to (指向), 56, 70
 - conversion from nonconst (转换自非常量), 144
 - initialization from nonconst (初始化自非常量), 56
 - overloaded parameter (重载形参), 208
- reference, *see* reference to const (引用, 参见“常量引用”)
- top-level const (顶层const), 57
 - and auto (和auto), 61
 - argument and parameter (实参与形参), 190
 - decltype, 63
 - parameter (形参), 208
 - template argument deduction (模板实参推断), 601
- variable (变量), 53
 - declared in header files (在头文件中声明), 67

- extern, 54
local to file (文件局部), 54
- const_cast**, 145, 145
- const_iterator**, container (**const_iterator**, 容器), 97, 297
- const_reference**, container (**const_reference**, 容器), 298
- const_reverse_iterator**, container (**const_reverse_iterator**, 容器), 298, 298
- constant expression (常量表达式), 58, 69
 array dimension (数组维度), 101
 bit-field (位域), 756
 case label (case标签), 161
 enumerator (枚举成员), 737
 integral (整型), 58
 nontype template parameter (无类型模板形参), 580
 sizeof, 139
 static data member (静态数据成员), 270
- constexpr**, 59, 69
 constructor (构造函数), 267
 declared in header files (在头文件中声明), 67
 function (函数), 214, 225
 nonconstant return value (非常量返回值), 214
 function template (函数模板), 581
 pointer (指针), 59
 variable (变量), 59
- construct**
 allocator, 428
 forwards to constructor (转到构造函数), 468
- constructor** (构造函数), 235, 236, 235–238, 273
 see also default constructor (参见“默认构造函数”)
 see also copy constructor (参见“拷贝构造函数”)
 see also move constructor (参见“移动构造函数”)
 calls to virtual function (调用虚函数), 556
 constexpr, 267
 converting (转换), 263, 273
 function matching (函数匹配), 518
 Sales_data, 264
 with standard conversion (与标准类型转换), 515
 default argument (默认实参), 260
 delegating (委托), 261, 274
 derived class (派生类), 531
 initializes direct base class (初始化直接基类), 541
 initializes virtual base (初始化虚基类), 720
 explicit, 265, 274
 function try block (函数try块), 689, 723
 inherited (继承), 557
 initializer list (初始值列表), 237, 258–261, 273
- class member initialization (类成员初始化), 245
compared to assignment (对比赋值), 258
derived class (派生类), 531
function try block (函数try块), 689, 723
sometimes required (有时必不可少), 258
virtual base class (虚基类), 720
- initializer_list** parameter
(**initializer_list**形参), 587
- not **const** (构造函数不能声明为**const**), 235
- order of initialization (初始化顺序), 259
 derived class object (派生类对象), 531, 553
 multiple inheritance (多重继承), 712
 virtual base classes (虚基类), 720
- overloaded (重载), 235
- StrBlob**, 405
- StrBlobPtr**, 421
- TextQuery**, 433
- Blob**, 586
 initializer_list, 587
 iterator parameters (迭代器形参), 596
- Bulk_quote**, 531, 541
- Disc_quote**, 540
- Sales_data**, 236–238
- container** (容器), 86, 117, 292, 332
 see also sequential container (参见“顺序容器”)
 see also associative container (参见“关联容器”)
 adaptor (适配器), 329, 329–330
 equality and relational operators (相等性运算符和关系运算符), 329
 initialization (初始化), 329
 requirements on container (对容器的要求), 329
 and inheritance (和继承), 558
 as element type (作为元素类型), 87, 294
 associative (关联的), 374, 397
 copy initialization (拷贝初始化), 299
 element type constraints (元素类型约束), 294, 305
 elements and destructor (元素与析构函数), 445
 elements are copies (元素是副本), 306
 initialization from iterator range (由迭代器范围初始化), 300
 list initialization (列表初始化), 300
 members (成员)
 see also iterator (参见“迭代器”)
 = (assignment) (赋值), 302
 == (equality) (相等), 304
 != (inequality) (不相等), 304
 begin, 95, 298, 332
 cbegin, 98, 298, 299, 332
 cend, 98, 298, 299, 332
 const_iterator, 97, 297
 const_reference, 298

const_reverse_iterator, 296, 298
crbegin, 298
crend, 296
difference_type, 117, 297
empty, 78, 91, 117, 304
end, 95, 117, 298, 332
equality and relational operators(相等性运算符和关系运算符), 79, 91, 304
iterator, 97, 297
rbegin, 298, 363
reference, 298
relational operators(关系运算符), 304
rend, 298, 363
reverse_iterator, 298, 364
size, 79, 91, 118, 304
size_type, 79, 91, 118, 297
swap, 303
move operations(移动运算符), 469
 moved-from object is valid but unspecified(源对象在移动后是有效的但值未指定), 475
nonmember swap(非成员swap), 303
of container(容器的), 87, 294
overview(概览), 294
sequential(顺序的), 292, 333
type members, ::(scope operator)(类型成员,作用域), 298
continue statement(continue语句), 171, 178
control, flow of(控制流), 10, 154, 178
conversion(类型转换), 69, 141, 149
 = (assignment)(赋值), 129, 141
 ambiguous(二义性), 517–522
 argument(实参), 183
 arithmetic(算术), 33, 142, 149
 array to pointer(数组转换为指针), 105
 argument(实参), 192
 exception object(异常对象), 686
 multidimensional array(多维数组), 114
 template argument deduction(模板实参推断), 601
base-to-derived, not automatic(从基类向派生类的转换, 非自动), 534
bool, 32
class type(类类型), 144, 263, 273, 523
 ambiguities(二义性), 521
 conversion operator(类型转换运算符), 514
 function matching(函数匹配), 517, 519
 with standard conversion(与标准类型转换), 515
copy algorithms(*_copy*算法), 342, 369
copy, 341, 774
copy and swap assignment(拷贝与交换赋值), 459
 move assignment(移动赋值), 477
 self-assignment(自赋值), 459
copy assignment(拷贝赋值), 444, 486
 = default, 449
 = delete, 449
base from derived(由派生类构造的基类), 535
copy and swap(拷贝与交换), 459, 486

derived class (派生类), 555
HasPtr
 reference counted (引用计数), 456
 valuelike (类值), 453
memberwise (逐个成员), 443
Message, 463
preventing copies (阻止拷贝), 449
private, 451
reference count (引用计数), 455
rule of three/five (三/五法则), 448
 virtual destructor exception (虚析构函数的例外), 552
self-assignment (自赋值), 453
StrVec, 467
synthesized (合成的), 444, 487
 deleted function (删除的函数), 450, 553
 derived class (派生类), 553
 multiple inheritance (多重继承), 712
union with class type member (含有类类型成员的联合), 753
 valuelike class (类值类), 453
copy constructor (拷贝构造函数), 440, 440–442, 486
 = default, 449
 = delete, 449
 base from derived (由派生类构造的基类), 535
 derived class (派生类), 555
 HasPtr
 reference counted (引用计数), 456
 valuelike (类值), 453
 memberwise (逐个成员), 441
 Message, 462
 parameter (形参), 440
 preventing copies (阻止拷贝), 449
 private, 451
 reference count (引用计数), 455
 rule of three/five (三/五法则), 448
 virtual destructor exception (虚析构函数异常), 552
StrVec, 467
synthesized (合成的), 441, 487
 deleted function (删除的函数), 450, 553
 derived class (派生类), 553
 multiple inheritance (多重继承), 712
union with class type member (含有类类型成员的union), 753
used for copy-initialization (用于拷贝初始化), 441
copy control (拷贝控制), 239, 440, 486
 = delete, 449–450
inheritance (继承), 553–558
memberwise (逐个成员), 239, 487
 copy assignment (拷贝赋值), 444
copy constructor (拷贝构造函数), 441
move assignment (移动赋值), 476
move constructor (移动构造函数), 476
multiple inheritance (多重继承), 712
synthesized (合成的), 239
 as deleted function (作为删除的函数), 450
 as deleted in derived class (在派生类中作为删除的成员), 553
move operations as deleted function (移动操作作为删除的函数), 476
unions, 751
virtual base class, synthesized (虚基类, 合成的), 721
copy initialization (拷贝初始化), 76, 117, 441, 441–442, 486
array, 301
container (容器), 299
container elements (容器元素), 306
explicit constructor (显式构造函数), 442
invalid for arrays (对数组无效), 102
move vs. copy (移动与拷贝), 476
parameter and return value (形参和返回值), 442
uses copy constructor (使用拷贝构造函数), 441
uses move constructor (使用移动构造函数), 478
copy_backward, 774
copy_if, 774
copy_n, 774
copyUnion, Token, 753
count, reference (计数, 引用), 487
count
 algorithm (算法), 337, 771
 associative container (关联容器), 388, 389
 bitset, 644
count_calls, program (count_calls程序), 185
count_if, 771
cout, 5, 23
 tied to cin (绑定到cin), 282
cplusplus_primer, namespace (命名空间), 697
crbegin, container (容器), 298
cref, binds reference parameter (绑定引用形参), 356, 371
cregex_iterator, 649, 680
crend, container (容器), 298
cstddef header (头文件), 103, 107
cstdlib header (头文件), 48, 203, 689, 728
cstring
 functions (函数), 109–110
 header (头文件), 109
csub_match, 649, 680
ctime header (头文件), 663
curly brace (花括号), 2, 23

D

dangling `else` (空悬 `else`) , 158, 178
dangling pointer (空悬指针) , 202, 411, 436
 undefined behavior (未定义的行为) , 413
data abstraction (数据抽象) , 228, 273
data hiding (数据隐藏) , 242
data member, *see* class data member (数据成员, 参见“类数据成员”)
data structure (数据结构) , 17, 23
`deallocate, allocator`, 429, 467
`debug_re` program (`debug_re`程序)
 additional nontemplate versions (额外的非模板版本) , 617
 general template version (通用模板版本) , 615
 nontemplate version (非模板版本) , 616
 pointer template version (指针模板版本) , 615
`DebugDelete`, member template (成员模板) , 596
`dec`, manipulator (操纵符) , 667
decimal, literal (十进制, 字面值常量) , 35
declaration (声明) , 41, 69
 class (类) , 250, 273
 class template (类模板) , 592
 class type, variable (类类型, 变量) , 262
 compound type (复合类型) , 51
 dependencies (依赖性)
 member function as friend (成员函数作为友元) , 252
 overloaded templates (重载模板) , 617
 template friends (模板友元) , 589
 template instantiation (模板实例化) , 582
 template specializations (模板特例化) , 625
 variadic templates (可变参数模板) , 620
 derived class (派生类) , 532
 explicit instantiation (显式实例化) , 597
 friend (友元) , 241
 function template (函数模板) , 592
 instantiation (实例化) , 630
 member template (成员模板) , 596
 template (模板) , 592
 template specialization (模板特例化) , 625
 type alias (类型别名) , 60
 using , 74, 118
 access control (访问控制) , 545
 overloaded inherited functions (重载继承的函数) , 551
 variable (变量) , 41
 `const` , 54
declarator (声明符) , 45, 69
`decltype`, 62, 69
 array return type (数组返回类型) , 206
`cbegin`, 98, 338

`cend`, 98, 338
depends on form (依赖形式) , 63
for type abbreviation (为类型缩写) , 79, 94, 115
of array (数组的) , 105
of function (函数的) , 223
pointer to function (函数指针) , 222
top-level `const` (顶层 `const`) , 63
yields lvalue (产生左值) , 63, 121
decrement operators (递减运算符) , 131–133
default argument (默认实参) , 211, 225
 adding default arguments (添加默认实参) , 212
 and header file (和头文件) , 213
 constructor (构造函数) , 260
 default constructor (默认构造函数) , 260
 function call (函数调用) , 211
 function matching (函数匹配) , 217
 initializer (初始值) , 212
 static member (静态成员) , 271
 virtual function (虚函数) , 538
default case label (default case 标签) , 162, 178
default constructor (默认构造函数) , 236, 274
 `= default`, 237
 `= delete`, 449
 default argument (默认实参) , 260
 `Sales_data`, 235
 `StrVec`, 465
 synthesized (合成的) , 236, 274
 deleted function (删除的函数) , 450, 553
 derived class (派生类) , 553
`Token`, 751
used implicitly (隐式使用)
default initialization (默认初始化) , 262
value initialization (值初始化) , 262
default initialization (默认初始化) , 39
 `array`, 300
 built-in type (内置类型) , 39
 class type (类类型) , 40
 `string`, 40, 76
 uses default constructor (使用默认构造函数) , 262
 `vector`, 87
default template argument (默认模板实参) , 594
 class template (类模板) , 594
 `compare`, 594
 function template (函数模板) , 594
 `template<>`, 594
`default_random_engine`, 659, 680
`defaultfloat` manipulator (`defaultfloat` 操纵符) , 670
definition (定义) , 70
 `array` (数组) , 101
 associative container (关联容器) , 377
 base class (基类) , 527

class (类), 64, 230–239
class template (类模板), 583
 member function (成员函数), 585
 static member (静态成员), 590
class template partial specialization (类模板部分特例化), 628
derived class (派生类), 530
dynamically allocated object (动态分配的对象), 407
explicit instantiation (显式实例化), 597
function, 512
in if condition (在if条件中), 157
in while condition (在while条件中), 165
instantiation (实例化), 630
member function (成员函数), 230–233
multidimensional array (多维数组), 112
namespace (命名空间), 695
 can be discontiguous (允许不连续), 696
 member (成员), 698
overloaded operator (重载运算符), 443, 490
pair, 379
pointer (指针), 47
pointer to function (函数指针), 221
pointer to member (成员指针), 740
reference (引用), 46
sequential container (顺序容器), 299
shared_ptr, 400
static member (静态成员), 270
string, 76
template specialization (模板特例化), 624–629
unique_ptr, 417, 419
variable (变量), 38, 41
 const, 54
variable after case label (case标签后的变量), 163
vector, 87
weak_ptr, 420
delegating constructor (委托构造函数), 261, 274
delete, 409, 409–411, 436
 const object (const对象), 409
execution flow (执行流程), 726
memory leak (内存泄露), 410
null pointer (空指针), 410
pointer (指针), 409
 runs destructor (运行析构函数), 445
delete [], dynamically allocated array (动态分配的数组), 424
deleted function (删除的函数), 449, 486
deleter (删除器), 416, 436
 shared_ptr, 416, 426, 436
 unique_ptr, 419, 436
deprecated (已弃用的), 357
auto_ptr, 419
binary_function, 513
bind1st, 357
bind2nd, 357
generalized exception specification (泛化异常说明), 691
ptr_fun, 357
unary_function, 513
deque, 332
 see also container, container member (参见“容器”和“容器成员”)
sequential container (参见“顺序容器”)
[] (subscript) (下标), 310
at, 310
header (头文件), 294
initialization (初始化), 299–301
list initialization (列表初始化), 300
overview (概述), 292
push_back, invalidates iterator (令迭代器无效), 315
push_front, invalidates iterator (令迭代器无效), 315
random-access iterator (随机访问迭代器), 367
value initialization (值初始化), 300
dereference, StrBlobPtr, 422
derived class (派生类), 526, 575
 see also virtual function (参见“虚函数”)
 :: (scope operator) to access baseclass member (基类成员的作用域运算符), 539
 = (assignment) (赋值), 555
access control (访问控制), 544
as base class (作为基类), 533
assigned or copied to base object (赋值或拷贝给基类), 535
base-to-derived conversion, not automatic (基类向派生类的转换, 非自动), 534
constructor (构造函数), 531
 initializer list (初始值列表), 531
 initializes direct base class (初始化直接基类), 541
 initializes virtual base (初始化虚基类), 720
copy assignment (拷贝赋值), 555
copy constructor (拷贝构造函数), 555
declaration (声明), 472
default derivation specifier (默认派生说明符), 546
definition (定义), 530
derivation list (派生列表), 530, 575
 access control (访问控制), 543
derived object (派生类对象)
 contains base part (包含基类部分), 530
multiple inheritance (多重继承), 710
derived-to-base conversion (派生类到基类的类型)

- 转换), 530
 accessibility (可访问性), 544
 key concepts (关键概念), 537
 multiple inheritance (多重继承), 712
 destructor (析构函数), 556
 direct base class (直接基类), 533, 575
 final, 472
 friendship not inherited (友元关系不可继承), 545
 indirect base class (间接基类), 533, 576
 is user of base class (是基类的成员), 544
 member new and delete (作为成员的new和delete), 727
 move assignment (移动赋值), 555
 move constructor (移动构造函数), 555
 multiple inheritance (多重继承), 710
 name lookup (名字查找), 547
 order of destruction (析构顺序), 556
 multiple inheritance (多重继承), 712
 order of initialization (初始化顺序), 531, 553
 multiple inheritance (多重继承), 712
 virtual base classes (虚基类), 720
 scope (作用域), 547
 hidden base members (隐藏基类成员), 549
 inheritance (继承), 547–551
 multiple inheritance (多重继承), 715
 name lookup (名字查找), 547
 virtual function (虚函数), 550
 static members (静态成员), 532
 synthesized (合成的)
 copy control members (拷贝控制成员), 553
 deleted copy control members (删除的拷贝控制成员), 553
 using declaration (using声明)
 access control (访问控制), 544
 overloaded inherited functions (重载继承的函数), 551
 virtual function (虚函数), 530
 derived-to-base conversion (派生类到基类的类型转换), 530, 575
 accessible (可访问的), 544
 key concepts (关键概念), 537
 multiple inheritance (多重继承), 712
 not base-to-derived (不存在基类向派生类的转换), 534
 shared_ptr, 559
 design (设计)
 access control (访问控制), 544
 Bulk_quote, 526
 conversion operator (类型转换运算符), 515
 Disc_quote, 540
 equality and relational operators (相等性运算符和关系运算符), 498
 generic programs (泛型程序), 581
 inheritance (继承), 564
 Message class (Message类), 460
 namespace (命名空间), 696
 overloaded operator (重载运算符), 491–493
 Query classes (Query类), 564–566
 Quote, 526
 reference count (引用计数), 455
 StrVec, 464
 destination sequence (目的序列), 340, 368
 destroy, allocator, 428, 467
 destructor (析构函数), 402, 436, 444, 444–446, 486
 = default, 449
 called during exception handling (在异常处理过程中调用), 685
 calls to virtual function (调用虚函数), 556
 container elements (容器元素), 445
 derived class (派生类), 556
 doesn't delete pointer members (不删除指针成员), 446
 explicit call to (显式调用), 729
 HasPtr
 reference counted (引用计数), 456
 valuelike (类值), 453
 local variables (局部变量), 445
 Message, 462
 not deleted function (不应该被删除的函数), 450
 not private (不是私有的), 451
 order of destruction (析构顺序), 445
 derived class (派生类), 556
 multiple inheritance (多重继承), 712
 virtual base classes (虚基类), 721
 reference count (引用计数), 455
 rule of three/five (三/五法则), 447
 virtual destructor, exception (虚析构函数, 异常), 552
 run by delete (由delete运行), 445
 shared_ptr, 402
 should not throw exception (不应该抛出异常), 685
 StrVec, 467
 synthesized (合成的), 446, 487
 deleted function (删除的函数), 450, 553
 derived class (派生类), 553
 multiple inheritance (多重继承), 712
 Token, 751
 valuelike class (类值类), 453
 virtual function (虚函数), 552
 virtual in base class (基类中的虚函数), 552
 development environment, integrated (集成开发环境), 3
 difference_type, 100
 vector, 100

- container (容器), 117, 297
string, 100
- direct base class (直接基类), 533
- direct initialization (直接初始化), 76, 117
emplace members use (成员使用), 308
- Disc_quote
- abstract base class (抽象基类), 541
 - class definition (类定义), 540
 - constructor (构造函数), 541
 - design (设计), 540
- discriminant (判别式), 751, 762
Token, 751
- distribution types (分布类型)
- bernoulli_distribution, 665
 - default template argument (默认模板实参), 664
 - normal_distribution, 664
 - random-number library (随机数标准库), 659
 - uniform_int_distribution, 661
 - uniform_real_distribution, 664
- divides<T>, 510
- division rounding (除法取整), 125
- do while statement (do while语句), 169, 178
- domain_error, 176
- double, 30
- literal (numEnum or numenum) (字面值常量), 35
 - output format (输出格式), 668
 - output notation (输出记数法), 670
- dynamic binding (动态绑定), 527, 575
- requirements for (要求), 535
 - static vs. dynamic type (静态类型与动态类型), 537
- dynamic type (动态类型), 534, 575
- dynamic_cast, 144, 730, 730, 762
- bad_cast, 731
 - to pointer (指针的), 730
 - to reference (引用的), 731
- dynamically allocated (动态分配), 400, 436
- array (数组), 423, 423–429
 - allocator, 427
 - can't use begin and end (无法调用begin或end), 424
 - can't use range for statement (无法使用范围for语句), 424
 - delete[], 425
 - empty array (空数组), 424
 - new[], 424
 - returns pointer to an element (返回指向元素的指针), 424
 - shared_ptr, 426
 - unique_ptr, 425
- delete runs destructor (delete运行析构函数), 445
- lifetime (生存期), 400
- new runs constructor (new运行构造函数), 407
- object (对象), 407–411
- const object (常量对象), 408
 - delete, 409
 - factory program (factory程序), 409
 - initialization (初始化), 407
 - make_shared, 401
 - new, 407
 - shared objects (共享对象), 404, 431
 - shared_ptr, 412
 - unique_ptr, 417
- ## E
- echo command (echo命令), 4
- ECMAScript, 646, 654
- regular expression library (正则表达式库), 646
- edit-compile-debug (编辑-编译-调试), 15, 23
- errors at link time (链接时错误), 582
- element type constraints, container (元素类型约束, 容器), 294, 304
- elimDups program (elimDups程序), 342–348
- ellipsis, parameter (省略符形参), 199
- else, see if statement (参见“if语句”)
- emplace
- associative container (关联容器), 384
 - priority_queue, 330
 - queue, 330
 - sequential container (顺序容器), 308
 - stack, 330
- emplace_back
- sequential container (顺序容器), 308
 - StrVec, 623
- emplace_front, sequential container (顺序容器), 308
- empty
- container (容器), 78, 92, 117, 304
 - priority_queue, 330
 - queue, 330
 - stack, 330
- encapsulation (封装), 228, 274
- benefits of (益处), 242
- end
- associative container (关联容器), 382
 - container (容器), 95, 117, 298, 332
 - function (函数), 106, 117
 - multidimensional array (多维数组), 115
 - StrBlob, 422
 - StrVec, 465
- end-of-file (文件结束), 13, 23, 674
- character (字符), 13

- Endangered, 710
endl, 6
 manipulator (操纵符), 282
ends, manipulator (操纵符), 282
engine, random-number library (标准库随机数引擎), 660, 680
 default_random_engine, 660
 max, min, 661
 retain state (保持状态), 662
 seed, 662, 681
enum, unscoped enumeration (不限定作用域的枚举类型), 737
enum class, scoped enumeration (限定作用域的枚举类型), 736
enumeration (枚举类型), 736, 762
 as union discriminant (作为union判别式), 752
 function matching (函数匹配), 739
 scoped (限定作用域的), 736, 763
 unscoped (不限定作用域的), 736, 763
 conversion to integer (转换为整数), 738
 unnamed (未命名的), 736
enumerator (枚举成员), 736, 762
 constant expression (常量表达式), 737
 conversion to integer (转换为整数), 738
eof, 280
eofbit, 280
equal, 339, 772
equal virtual function (虚函数), 734
equal_range
 algorithm (算法), 639, 773
 associative container (关联容器), 390
equal_to<T>, 509
equality operators (相等运算符), 126
 arithmetic conversion (算术类型转换), 128
 container adaptor (容器适配器), 329
 container member (容器成员), 304
 iterator (迭代器), 95
 overloaded operator (重载运算符), 498
 pointer (指针), 107
 Sales_data, 497
 string, 79
 vector, 91
erase
 associative container (关联容器), 386
 changes container size (改变容器大小), 344
 invalidates iterator (令迭代器无效), 312
 sequential container (顺序容器), 311
 string, 323
error, standard (错误, 标准), 5
error_type, 649
error_msg program (error_msg程序), 198
ERRORLEVEL, 4
escape sequence (转义序列), 36, 70
 hexadecimal (\Xnnn) (十六进制), 36
 octal (\nnn) (八进制), 36
eval function (eval函数)
 AndQuery, 572
 NotQuery, 573
 OrQuery, 572
exception
 class (类), 173, 178
 class hierarchy (类层次), 693
 deriving from (派生自), 692
 Sales_data, 694
 header (头文件), 176
 initialization (初始化), 176
 what, 175, 693
exception handling (异常处理), 173–177, 684, 722
 see also throw (参见throw)
 see also catch (参见catch)
 exception declaration (异常声明), 174, 687, 722
 and inheritance (和继承), 687
 must be complete type (必须是完整类型), 687
exception in destructor (析构函数的异常), 685
exception object (异常对象), 686, 723
 finding a catch (找到一个catch), 687
 function try block (函数try块), 689, 724
 handler, *see catch* (处理代码, 见catch)
 local variables destroyed (局部变量销毁), 685
 noexcept specification (noexcept说明), 473, 690, 723
 nonthrowing function (不抛出异常的函数), 690, 723
 safe resource allocation (安全资源分配), 415
 stack unwinding (栈展开), 685, 723
 terminate function (terminate函数), 175, 179
 try block (try块), 174, 684
 uncaught exception (未捕获的异常), 685
 unhandled exception (未处理的异常), 175
exception object (异常对象), 686, 722
 catch, 687
 conversion to pointer (转换为指针), 686
 initializes catch parameter (初始化catch形参), 687
 pointer to local object (局部对象的指针), 686
 rethrow (重新抛出), 688
exception safety (异常安全), 175, 179
 smart pointers (智能指针), 415
exception specification (异常说明)
 argument (实参), 691
 generalized, deprecated (更一般化的版本已弃用), 691
 noexcept, 690

- nonthrowing (不抛出异常), 690
pointer to function (函数指针), 690, 692
throw(), 691
violation (违反), 690
virtual function (虚函数), 692
executable file (可执行文件), 4, 225
execution flow (执行流程)
 () (call operator) (调用运算符), 183
 delete, 726
 for statement (for语句), 166
 new, 726
 switch statement (switch语句), 161
 throw, 174, 684
EXIT_FAILURE, 203
EXIT_SUCCESS, 204
expansion (扩展)
 forward, 623
 parameter pack (参数包), 620, 620–622, 630
 function parameter pack (函数参数包), 621
 template parameter pack (模板参数包), 621
 pattern (模式), 621
explicit
 constructor (构造函数), 265, 273
 copy initialization (拷贝初始化), 442
 conversion operator (类型转换运算符), 516, 523
 conversion to **bool** (转换为bool), 516
explicit call to (显式调用)
 destructor (析构函数), 730
 overloaded operator (重载运算符), 491
 postfix operators (后置运算符), 504
explicit instantiation (显式实例化), 597, 630
explicit template argument (显式模板实参), 584, 630
 class template (类模板), 584
 forward, 614
 function template (函数模板), 603
 function pointer (函数指针), 607
 template argument deduction (模板实参推断), 604
exporting C++ to C (导出C++到C), 760
expression (表达式), 6, 23, 120, 149
 callable, *see callable object* (可调用的, 参见“可调用对象”)
 constant (常量), 58, 69
 lambda, *see lambda expression* (lambda, 参见“lambda表达式”)
 operand conversion (运算对象类型转换), 142
 order of evaluation (求值顺序), 123
 parenthesized (括号之中的), 122
 precedence and associativity (优先级和结合律), 121–123
 regular, *see regular expression* (正则, 参见“正则表达式”)
expression statement (表达式语句), 154, 179
extension, compiler (编译器扩展), 102, 117
extern
 and **const** variables (和const变量), 54
 explicit instantiation (显式实例化), 597
 variable declaration (变量声明), 41
extern 'C', *see linkage directive* (参见“链接指示”)
- ## F
- fact program** (fact程序), 182
factorial program (factorial程序), 204
factory program (factory程序)
 new, 410
 shared_ptr, 402
fail, 280
failbit, 279
failure (失败), **new**, 408
file, source (源文件), 3
file extension, program (文件扩展, 程序), 647
 version 2 (版本2), 654
file marker, stream (文件标记, 流), 676
file mode (文件模式), 286, 290
file redirection (文件重定向), 19
file static (文件中的静态声明), 701, 723
file stream, *see fstream* (文件流, 参见fstream)
fill, 340, 773
fill_n, 340, 773
final specifier (final说明符), 533
 class (类), 533
 virtual function (虚函数), 538
find
 algorithm (算法), 336, 771
 associative container (关联容器), 388, 389
 string, 324
find last word program (找出最后一个词的程序), 364
find_char program (find_char程序), 189
find_first_of, 772
find_first_not_of, **string**, 325
find_first_of, 772
 string, 325
find_if, 346, 354, 369, 771
find_if_not, 771
find_if_not_of, 771
find_last_not_of, **string**, 326
find_last_of, **string**, 326
findBook, program (findBook程序), 639
fixed manipulator (fixed操纵符), 670
flip
 bitset, 644
 program (程序), 614
flip1, program (flip1程序), 612
flip2, program (flip2程序), 613

- float**, 30
 literal (*numF* or *numf*) (字面值常量), 37
floating-point (浮点), **30**
 conversion (类型转换), 32
 literal (字面值常量), 35
 output format (输出格式), 668
 output notation (输出记号), 670
flow of control (控制流), 10, **154**, 179
flush, manipulator (*flush*操纵符), 282
Folder, 参见**Message**
for statement (*for*语句), **11**, **24**, **166**, 166–168, 179
 condition (条件), 10
 execution flow (执行流程), 166
 for header (*for*语句头), 166
 range (范围), **82**, **168**, 168–169, 179
 can't add elements (无法添加元素), 90, 168
 multidimensional array (多维数组), 114
for_each, 348, 772
format state, stream (流格式化状态), 667
formatted IO (格式化的输入输出), **673**, 680
forward, **614**
 argument-dependent lookup (实参相关的查找), 707
 explicit template argument (显式模板实参), 614
 pack expansion (包扩展), 623
 passes argument type unchanged (传递过程中保持参数类型不变), 614, 623
 usage pattern (用法模式), 624
forward declaration, class (类前向声明), **250**, 274
forward iterator (前向迭代器), **366**, 371
forward_list
 see also container (参见“容器”)
 see also sequential container (参见“顺序容器”)
 before_begin, 313
 forward iterator (前向迭代器), 366
 header (头文件), 294
 initialization (初始化), 299–301
 list initialization (列表初始化), 300
 merge, 369
 overview (概览), 294
 remove, 369
 remove_if, 369
 reverse, 369
 splice_after, 370
 unique, 369
 value initialization (值初始化), 300
forwarding (转发), 612–614
 passes argument type unchanged (传递过程中保持参数类型不变), 614
preserving type information (保持类型信息), 613
rvalue reference parameters (右值引用形参), 613,
- 623
- typical implementation** (典型实现), 624
variadic template (可变参数模板), 622
free, StrVec, 467
free library function (*free*标准库函数), **728**, 762
free store (自由空间), **400**, 436
friend (友元), **241**, 274
 class (类), 251
 class template type parameter (类模板类型形参), 590
 declaration (声明), **241**
 declaration dependencies (声明依赖性)
 member function as friend (成员函数作为友元), 252
 template friends (模板友元), 589
 function (函数), 241
 inheritance (继承), 544
 member function (成员函数), 251, 252
 overloaded function (重载函数), 252
 scope (作用域), 242, 252
 namespace (命名空间), 707
 template as (模板), 588
front
 queue, 330
 sequential container (顺序容器), 309
 StrBlob, 406
front_inserter, **358**, 371
 compared to inserter (对比inserter), 358
 requires push_front (需要push_front), 358
fstream, 283–286
 close, 285
 file marker (文件标记), 676
 file mode (文件模式), **286**
 header (头文件), 278, 283
 initialization (初始化), 284
 off_type, 677
 open, 284
 pos_type, 677
 random access (随机访问), 676
 random IO program (随机输入输出程序), 677
 seek和tell, 675–679
function (函数), **2**, **24**, **182**, 225
 see also return type (参见“返回类型”)
 see also return value (参见“返回值”)
 block (块), 184
 body (体), **2**, **24**, **182**, 225
 callable object (可调用对象), 346
 candidate (候选), 225
 candidate function (候选函数), **217**
 constexpr, **214**, 225
 nonconstant return value (非常量返回值), 214
 declaration (声明), 186
 declaration and header file (声明与头文件), 186

- `decltype` returns function type (`decltype`返回函数类型), 223
`default argument` (默认实参), 211, 225
 adding default arguments (添加默认实参), 212
 and header file (和头文件), 213
 initializer (初始值), 212
`deleted` (删除的), 449, 486
 function matching (函数匹配), 450
`exception specification` (异常说明)
 `noexcept`, 690
 `throw()`, 691
`friend` (友元), 241
`function to pointer conversion` (函数向指针的类型转换), 221
`inline`, 213, 225
 and header (和头文件), 215
`linkage directive` (链接指示), 760
`member`, *see member function* (成员, 参见“成员函数”)
name (名字), 2, 24
`nonthrowing` (不抛出异常), 690, 723
`overloaded` (重载)
 compared to redeclaration (对比重新声明), 207
 friend declaration (友元声明), 252
 scope (作用域), 210
`parameter`, *see parameter* (形参)
`parameter list` (形参列表), 2, 24, 182, 183
`prototype` (原型), 186, 225
`recursive` (递归), 204
 variadic template (可变参数模板), 620
 scope (作用域), 184
 viable (可行的), 226
 viable function (可行函数), 217
 virtual, *see virtual function* (虚, 见虚函数)
`function`, 512, 511–514, 523
 and pointer to member (和成员指针), 745
 definition (定义), 511
 desk calculator (桌面计算器), 511
`function call` (函数调用)
 ambiguous (二义性), 209, 219, 225
 default argument (默认实参), 211
 execution flow (执行流程), 183
 overhead (开销), 213
 through pointer to function (通过函数指针), 221
 through pointer to member (通过成员指针), 742
 to overloaded operator (重载运算符), 491
 to overloaded postfix operator (重载后置运算符), 504
`function matching` (函数匹配), 208, 225
 = `delete`, 450
 argument, conversion (实参, 类型转换), 209
 candidate function (候选函数), 217
 overloaded operator (重载运算符), 521
`const` arguments (常量实参), 220
`conversion, class type` (类类型转换), 516–520
`conversion operator` (类型转换运算符), 518, 519
`conversion rank` (转换顺序), 219
 class type conversions (类类型转换), 519
`default argument` (默认实参), 217
`enumeration` (枚举), 739
`function template` (函数模板), 614–618
 specialization (特例化), 625
`integral promotions` (整型提升), 219
`member function` (成员函数), 244
`multiple parameters` (多个形参), 218
`namespace` (命名空间), 708
`overloaded operator` (重载运算符), 520–522
`prefers more specialized function` (更特例化的函数优先), 615
`rvalue reference` (右值引用), 477
`variadic template` (可变参数模板), 620
`viable function` (可行函数), 217
`function object` (函数对象), 506, 523
 argument to algorithms (算法实参), 507
 arithmetic operators (算术运算符), 509
 is callable object (是可调用对象), 506
`function parameter`, *see parameter* (函数形参, 参见“形参”)
`function parameter pack` (函数参数包), 618
 expansion (扩展), 621
 pattern (模式), 622
`function pointer` (函数指针), 221–223
 callable object (可调用对象), 346
 definition (定义), 221
 exception specification (异常说明), 690, 692
 function template instantiation (函数模板实例化), 607
 overloaded function (重载函数), 222
 parameter (形参), 222
 return type (返回类型), 184, 223
 using `decltype` (使用`decltype`), 223
 template argument deduction (模板实参推断), 607
 type alias declaration (类型别名声明), 222
 typedef (定义类型别名), 222
`function table` (函数表), 511, 511, 523, 743
`function template` (函数模板), 578, 630
 see also template parameter (参见“模板形参”)
 see also template argument deduction (参见“模板实参推断”)
 see also instantiation (参见“实例化”)
 argument conversion (实参类型转换), 602
 array function parameters (数组函数形参), 580
 candidate function (候选函数), 615
 compare, 578

string literal version (字符串字面值版本), 580
constexpr, 581
 declaration (声明), 592
 default template argument (默认模板实参), 594
 error detection (错误检测), 582
 explicit instantiation (显式实例化), 597, 597–598
 explicit template argument (显式模板实参), 603
 compare, 604
 function matching (函数匹配), 614–618
inline function (内联函数), 581
 non-type parameter (非类型形参), 580
 overloaded function (重载函数), 614–618
 parameter pack (参数包), 630
 specialization (特例化), 625, 630
 compare, 624
 function matching (函数匹配), 625
 is an instantiation (是一个实例), 625
 namespace (命名空间), 698
 scope (作用域), 626
 vs. overloading (对比重载), 625
 trailing return type (尾置返回类型), 605
 type-dependent code (类型相关的代码), 583
function try block (函数try块), 689, 724
functional header (functional头文件), 354, 356, 357, 509, 512, 746

G

g++, 4
gcount, istream, 675
generate, 773
generate_n, 773
 generic algorithms, *see* algorithms (泛型算法, 参见“算法”)
 generic programming (泛型编程), 97
 type-independent code (类型相关的代码), 581
get
 istream, 673
 multi-byte version (多字节版本), **istream**, 674
 returns int (返回int), **istream**, 674, 676
get<n>, 637, 680
getline, 78, 117
 istream, 674
 istringstream, 288
 TextQuery constructor (TextQuery构造函数), 433
 global function (全局函数)
 operator delete, 762
 operator new, 762
 global namespace (全局命名空间), 698, 723
 :: (scope operator) (作用域运算符), 699, 724
 global scope (全局作用域), 44, 70

global variable, lifetime (全局变量, 生命周期), 184
 GNU compiler (GNU编译器), 4
good, 280
goto statement (goto语句), 172, 179
 grade clusters program (成绩聚合程序), 92
greater<T>, 510
greater_equal<T>, 510

H

.h file header (.h头文件), 17
handler, *see* catch (处理代码, 参见catch)
 has-a relationship (有一个……关系), 564
hash<key_type>, 396, 397
 override (重载), 396
 specialization (特例化), 627, 698
 compatiblewith == (equality) (与相等运算符兼容), 627
hash function (哈希函数), 394, 397
HasPtr
 reference counted (引用计数), 455–456
 copy assignment (拷贝赋值), 456
 destructor (析构函数), 456
 value-like (类值), 453
 copy assignment (拷贝赋值), 453
 move assignment (移动赋值), 477
 move constructor (移动构造函数), 477
 swap, 457
header (头文件), 6, 24
 iostream, 24
 C library (C标准库), 82
 Const and **constexpr** (const和constexpr), 67
 default argument (默认实参), 213
 function declaration (函数声明), 186
 .h file (.h文件), 17
 #include, 5, 18
 inline function (内联函数), 214
 inline member function definition (内联成员函数定义), 244
 namespace members (命名空间成员), 696
 standard (标准), 5
 table of library names (标准库名字表), 766
 template definition (模板定义), 581
 template specialization (模板特例化), 625
 user-defined (用户定义的), 18, 67–68, 186, 214
 using declaration (using声明), 74
 Sales_data.h, 67
 Sales_item.h, 17
 algorithm, 336
 array, 293
 bitset, 641

- cassert, 215
ctype, 82
cmath, 664, 669
cstddef, 103, 107
cstdlib, 49, 203, 689, 728
cstring, 109
ctime, 663
deque, 294
exception, 176
forward_list, 294
fstream, 278, 283
functional, 354, 356, 357, 509, 512, 746
initializer_list, 197
iomanip, 669
iostream, 5, 278, 676
iterator, 106, 341, 358
list, 294
map, 374
memory, 400, 401, 427, 429
new, 176, 409, 424, 726
numeric, 336, 780
queue, 330
random, 660
regex, 645
set, 374
sstream, 278, 287
stack, 330
stdexcept, 174, 176
string, 66, 67, 76
tuple, 636
type_info, 176
type_traits, 605
typeinfo, 731, 732, 735
unordered_map, 374
unordered_set, 374
utility, 379, 469, 472, 614
vector, 86, 294
header guard (头文件保护符), 68, 70
preprocessor (预处理器), 68
heap (堆), 400, 436
hex, manipulator (操纵符), 667
hexadecimal (十六进制)
 escape sequence (\Xnnn) (转义序列), 36
 literal (0Xnum or 0xnum) (字面值常量), 35
hexfloat manipulator (操纵符), 670
high-order bits (高位), 641, 680
- ## I
- i before e, program (查找i在e之前的文本的程序), 646
 version 2 (版本2), 650
IDE (集成开发环境), 3
- identifier (标识符), 42, 70
 reserved (保留的), 42
_if algorithms (_if算法), 368
if statement (if语句), 15, 24, 156, 156–159, 179
 compared to switch (对比switch), 159
 condition (条件), 16, 156
 dangling else (空悬else), 158
 else branch (else分支), 16, 156, 178
ifstream, 279, 283–286, 290
 see also istream (参见istream)
close, 285
file marker (文件标记), 676
file mode (文件模式), 286
initialization (初始化), 284
off_type, 677
open, 285
pos_type, 677
random access (随机访问), 676
random IO program (随机输入输出程序), 676
seek与tell, 676–680
ignore, istream, 675
implementation (实现), 228, 228, 274
in (file mode) (文件模式), 286
in scope (在作用域中), 44, 70
in-class initializer (类内初始值), 65, 65, 70, 237, 237,
 246
#include
 standard header (标准头文件), 6, 17
 user-defined header (用户定义的头文件), 17
includes, 778
incomplete type (不完全类型), 250, 274
 can't be base class (不能是基类), 533
 not in exception declaration (不能在异常声明中),
 687
 restrictions on use (用法约束), 250
incr, StrBlobPtr, 422
increment operators (递增运算符), 131–133
indentation (缩进), 16, 158
index (索引), 84, 117
 see also [](subscript) (参见“下标”)
indirect base class (间接基类), 533, 576
inferred return type, lambda expression (推断返回类型,
 lambda表达式), 353
inheritance (继承), 576
 and container (和容器), 558
 conversions (类型转换), 536
 copy control (拷贝控制), 552–557
 friend (友元), 544
 hierarchy (层次), 526, 532
 interface class (接口类), 565
 IO classes (输入输出类), 279, 290
 name collisions (名字冲突), 548

private, 543, 576
protected, 543, 576
public, 543, 576
 vs. composition (对比组合), 564
 inherited, constructor (继承的构造函数), 557
 initialization (初始化)

- aggregate class (聚合类), 266
- array (数组), 102
- associative container (关联容器), 376, 377
- bitset, 641–643
- C-style string (C风格字符串), 109
- class type objects (类类型对象), 65, 235
- const
 - static** data member (静态数据成员), 270
 - class type object (类类型对象), 235
 - data member (数据成员), 259
 - object (对象), 53
- copy (拷贝), 76, 117, 441, 441–442, 486
- default (默认), 40, 262
- direct (直接), 76, 117
- dynamically allocated object (动态分配对象), 407
- exception, 176
- istream_iterator, 361
- list, *see* list initialization (列表, 参见“列表初始化”)
- lvalue reference (左值引用), 471
- multidimensional array (多维数组), 112
- new [], 424
- ostream_iterator, 361
- pair, 379
- parameter (形参), 183, 187
- pointer (指针), 47–48
 - to const (指向常量的), 56
- queue, 329
- reference (引用), 46
 - data member (数据成员), 259
 - to const (对常量的), 54
- return value (返回值), 200
- rvalue reference (右值引用), 471
- sequential container (顺序容器), 299–301
- shared_ptr, 412
- stack, 329
- string, 75–76, 320–322
- string streams (字符串流), 287
- tuple, 636
- unique_ptr, 417
- value (值), 88, 118, 262
- variable (变量), 38, 39, 71
- vector, 86–89
- vs. assignment (对比赋值), 39, 258
- weak_ptr, 420

- initializer_list, 197, 197–199, 225
- = (assignment) (赋值), 499
- constructor (构造函数), 587
- header (头文件), 197
- inline function (内联函数), 213, 225
- and header (和头文件), 215
- function template (函数模板), 581
- member function (成员函数), 230, 244
 - and header (和头文件), 244
- inline namespace (内联命名空间), 699, 723
- inner scope (内层作用域), 44, 70
- inner_product, 780
- inplace_merge, 775
- input, standard (输入, 标准), 5
- input iterator (输入迭代器), 366, 371
- insert
- associative container (关联容器), 384
- multiple key container (多关键字容器), 385
- sequential container (顺序容器), 306
- string, 323
- insert iterator (插入迭代器), 341, 357, 358, 371
- back_inserter, 358
- front_inserter, 358
- inserter, 358
- inserter, 358, 371
- compared to front_inserter (对比 front_inserter), 358
- instantiation (实例化), 87, 117, 579, 582, 630
- Blob, 584
- class template (类模板), 584
 - member function (成员函数), 587
- declaration (声明), 630
- definition (定义), 630
- error detection (错误检测), 582
- explicit (显式), 597–598
- function template from function pointer (来自函数指针的函数模板), 607
- member template (成员模板), 597
- static member (static成员), 591
- int, 30
- literal (字面值常量), 35
- integral (整型)
- constant expression (常量表达式), 58
- promotion (提升), 120, 142, 150
 - function matching (函数匹配), 220
 - type (类型), 30, 71
- integrated development environment (集成开发环境), 3
- interface (接口), 228, 274
- internal, manipulator (操纵符), 671
- interval, left-inclusive (左闭合区间), 333
- invalid pointer (无效指针), 47
- invalid_argument, 176
- invalidated iterator (无效迭代器)

and container operations (和容器操作), 315
undefined behavior (未定义的行为), 315
invalidates iterator (令迭代器无效)
 assign, 302
 erase, 312
 resize, 314
IO (输入输出)
 formatted (格式化的), **673, 680**
 unformatted (未格式化的), **673, 681**
IO classes (输入输出类)
 condition state (条件状态), **279, 290**
 inheritance (继承), 290
IO stream, *see stream* (输入输出流, 参见“流”)
Iomanip header (*iomanip*头文件), 669
iostate, 279
 machine-dependent (机器相关的), 279
iostream, 5
 file marker (文件标记), 677
 header (头文件), 6, 24, 278, 674
 off_type, 677
 pos_type, 677
 random access (随机访问), 676
 random IO program (随机输入输出程序), 676
 seek and tell (seek和tell), 676–679
 virtual base class (虚基类), 717
iota, 780
is-a relationship (是一种……关系), 564
is_partitioned, 775
is_permutation, 778
is_sorted, 776
is_sorted_until, 776
isalnum, 82
isalpha, 82
isbn
 Sales_data, 230
 Sales_item, 20
ISBN (书籍编号), 2
isbn_mismatch, 694
iscntrl, 82
isdigit, 82
isgraph, 82
islower, 82
isprint, 82
ispunct, 82
isShorter program (*isShorter*程序), 189
isspace, 82
istream, 5, 24, 279
 see also manipulator (参见“操纵符”)
 >>(input operator) (输入运算符), 7
 precedence and associativity (优先级和结合律),
 138
 as condition (作为条件), 13
 chained input (链式输入), 7
 condition state (条件状态), 279
 conversion (类型转换), 144
 explicit conversion to bool (显式转换为
 bool), 516
 file marker (文件标记), 676
 flushing input buffer (刷新输入缓冲区), 281
 format state (格式化状态), 667
 gcount, 675
 get, 673
 multi-byte version (多字节版本), 674
 returns int (返回int), 674, 676
 getline, 78, 288, 674
 ignore, 675
 no copy or assign (不能拷贝或赋值), 279
 off_type, 677
 peek, 673
 pos_type, 677
 put, 673
 putback, 673
 random access (随机访问), 676
 random IO program (随机输入输出程序), 676
 read, 675
 seek and tell (seek和tell), 675–679
 unformatted IO (未格式化的输入输出), 673
 multi-byte (多字节), 675
 single-byte (单字节), 673
 unget, 673
istream_iterator, **359, 371**
 >>(input operator) (输入运算符), 359
 algorithms (算法), 360
 initialization (初始化), 361
 off-the-end iterator (尾后迭代器), 359
 operations (操作), 360
 type requirements (类型需求), 362
istringstream, 279, **287, 287–289**
 see also istream (参见*istream*)
 word per line processing (一行一词), 392
 file marker (文件标记), 676
 getline, 288
 initialization (初始化), 287
 off_type, 677
 phone number program (电话号码程序), 288
 pos_type, 677
 random access (随机访问), 676
 random IO program (随机输入输出程序), 676
 seek and tell (seek和tell), 675–679
 TextQuery constructor (*TextQuery*构造函数),
 433
 isupper, 82
 isxdigit, 82
 iter_swap, 774

iterator (迭代器), 95, 95–100, 117
 ++ (increment) (递增), 96, 118
 -- (decrement) (递减), 96
 * (dereference) (解引用), 96
 += (compound assignment) (复合赋值), 99
 + (addition) (加法), 99
 - (subtraction) (减法), 99
 == (equality) (相等), 95, 96
 != (inequality) (不相等), 95, 96
 algorithm type independence (算法类型独立), 337
 arithmetic (算术), 99, 117
 compared to reverse iterator (对比反向迭代器), 364
 destination (目标), 368
 insert (插入), 358, 371
 move (移动), 358, 372, 480
 uninitialized_copy, 480
 off-the-beginning (首前位置)
 before_begin, 313
 forward_list, 313
 off-the-end (尾后位置), 95, 118, 333
 istream_iterator, 359
 parameter (形参), 194
 regex (正则表达式), 650
 relational operators (关系运算符), 99
 reverse (反向), 358, 363–364, 372
 stream (流), 357, 359–362, 372
 used as destination (作为目的地址), 341
iterator
 compared to reverse_iterator (对比
 reverse_iterator), 364
 container (容器), 97, 297
 header (头文件), 106, 341, 357
 set iterators are const (集合迭代器是常量), 382
 iterator category (迭代器类别), 365, 365–367, 372
 bidirectional iterator (双向迭代器), 367, 371
 forward iterator (前向迭代器), 366, 371
 input iterator (输入迭代器), 366, 371
 output iterator (输出迭代器), 366, 372
 random-access iterator (随机访问迭代器), 367,
 372
 iterator range (迭代器范围), 296, 296–297, 333
 algorithms (算法), 336
 as initializer of container (作为容器的初始值), 300
 container `erase` member (容器`erase`成员), 312
 container `insert` member (容器`insert`成员), 307
 left-inclusive (左闭合), 296
 off-the-end (尾后位置), 296

K

key concept (关键概念)
 algorithms (算法)
 and containers (和容器), 337
 iterator arguments (迭代器实参), 340
 class user (类用户), 229
 classes define behavior (类定义行为), 18
 defining an assignment operator (定义赋值运算
 符), 453
 dynamic binding in C++ (C++的动态绑定), 537
 elements are copies (元素是拷贝), 306
 encapsulation (封装), 242
 headers for template code (模板代码的头文件), 582
 indentation (缩进), 16
 inheritance and conversions (继承与类型转换), 537
 is A and has A relationships (是一个……关系和包
 含一个……关系), 564
 name lookup and inheritance (名字查找与继承), 549
 protected members (受保护的成员), 544
 refactoring (重构), 542
 respecting base class interface (关于基类接口), 532
 specialization declarations (特例化声明), 626
 type checking (类型检查), 42
 types define behavior (类型定义行为), 3
 use concise expressions (使用简洁的表达式), 132
key_type
 associative container (关联容器), 381, 397
 requirements (需求)
 ordered container (有序容器), 378
 unordered container (无序容器), 395

keyword table (关键字表), 43
 Koenig lookup (Koenig查找), 706

L

L'c' (wchar_t literal) (wchar_t字面常量), 35
 label (标签)
 case, 161, 178
 statement (语句), 172
 labeled statement (带标签语句), 172, 179
 lambda expression (lambda表达式), 346, 371
 arguments (实参), 347
 biggies program (biggies程序), 348
 reference capture (引用捕获), 350
 capture list (捕获列表), 346, 371
 capture by reference (引用捕获), 350
 capture by value (值捕获), 347, 350

- implicit capture (隐式捕获), 351
- inferred return type (推断返回类型), 347, 353
- mutable, 352
- parameters (形参), 347
- passed to `find_if` (传给`find_if`), 348
- passed to `stable_sort` (传给`stable_sort`), 347
- synthesized class type (合成的类类型), 508–509
- trailing return type (尾置返回类型), 353
- left, manipulator (操纵符), 670
- left-inclusive interval (左闭合区间), 296, 333
- length_error, 176
- less<T>, 510
- less_equal<T>, 510
- lettergrade, program (字母型成绩程序), 157
- lexicographical_compare, 780
- library function objects (标准库函数对象), 509
 - as arguments to algorithms (作为算法的实参), 510
- library names to header table (头文件表的库名字), 766
- library type (标准库类型), 5, 23, 74
- lifetime (生命周期), 184, 226
 - compared to scope (对比作用域), 184
 - dynamically allocated objects (动态分配对象), 400, 409
 - global variable (全局变量), 184
 - local variable (局部变量), 184
 - parameter (形参), 185
- linkage directive (链接指示), 758, 762
 - C++ to C (由C++向C), 760
 - compound (复合), 759
 - overloaded function (重载函数), 761
 - parameter or return type (形参或返回类型), 760
 - pointer to function (函数指针), 759
 - return type (返回类型), 760
 - single (单独的), 759
- linker (链接), 187, 225
 - template errors at link time (链接时模板错误), 582
- list, 333
 - see also* container (参见“容器”)
 - see also* sequential container (参见“顺序容器”)
- bidirectional iterator (双向迭代器), 366
- header (头文件), 294
- initialization (初始化), 299–301
- list initialization (列表初始化), 300
- merge, 369
- overview (概览), 294
- remove, 369
- remove_if, 369
- reverse, 369
- splice, 370
- unique, 369
- value initialization (值初始化), 300
- list initialization (列表初始化), 39, 70
 - = (assignment) (赋值), 129
- array, 301
- associative container (关联容器), 377
- container (容器), 300
- dynamically allocated, object (动态分配对象), 407
- pair, 380, 384, 467
- preferred (首选的), 89
- prevents narrowing (阻止缩减), 39
- return value (返回值), 203, 380, 467
- sequential container (顺序容器), 300
- vector, 88
- literal (字面值常量), 35, 35–37, 70
 - bool, 37
 - in condition (在条件中), 127
 - char, 36
 - decimal (十进制), 35
 - double (`numEnum` or `numenum`), 36
 - float (`numF` or `numf`), 37
 - floating-point (浮点), 35
 - hexadecimal (`0Xnum` or `0xnum`) (十六进制), 35
 - int, 35
 - long (`numL` or `numl`), 35
 - long double (`ddd.dddL` or `ddd.dddL`), 37
 - long long (`numLL` or `numll`), 35
 - octal (`0num`) (八进制), 35
 - string (字符串), 6, 24, 36
 - unsigned (`numU` or `numu`), 37
 - wchar_t, 37
- literal type (字面值类型), 59
- class type (类类型), 267
- local class (局部类), 754, 762
 - access control (访问控制), 755
 - name lookup (名字查找), 755
 - nested class in (嵌套类), 755
 - restrictions (约束), 754
- local scope, *see* block scope (局部作用域, 参见“块作用域”)
- local static object (局部静态对象), 185, 226
- local variable (局部变量), 184, 226
 - destroyed during exception handling (异常处理时销毁), 415, 685
 - destructor (析构函数), 445
 - lifetime (生命周期), 184
 - pointer, return value (指针, 返回值), 201
 - reference, return value (引用, 返回值), 201
 - return statement (return语句), 200
- lock, weak_ptr, 420
- logic_error, 176
- logical operators (逻辑运算符), 123, 126

- condition (条件), 123
 function object (函数对象), 509
logical_and<T>, 510
logical_not<T>, 510
logical_or<T>, 510
long, 30
 literal (*numL* or *num1*) (字面值), 35
long double, 30
 literal (*ddd.dddL* or *ddd.ddd1*) (字面值), 37
long long, **30**
 literal (*numLL* or *num11*) (字面值), 35
lookup, name, see name lookup (名字查找)
low-level const (底层const), **57, 70**
 argument and parameter (实参和形参), 191
 conversion from (转换自), 145
 conversion to (转换为), 144
 overloaded function (重载函数), 208
 template argument deduction (模板实参推断), 612
low-order bits (低位), **641, 680**
lower_bound
 algorithm (算法), 773
 ordered container (有序容器), 389
lround, 664
lvalue (左值), **121, 149**
 cast to rvalue reference (转换为右值引用), 612
 copy initialization, uses copy constructor (拷贝初始化, 使用拷贝构造函数), 477
decltype, 121
 reference collapsing rule (引用折叠准则), 609
 result (结果)
 ->(arrow operator) (箭头运算符), 133
 ++ (increment) prefix (前置递增), 133
 -- (decrement) prefix (前置递减), 133
 * (dereference) (解引用), 120
 [](subscript) (下标), 120
 = (assignment) (赋值), 129
 , (comma operator) (逗号运算符), 140
 ?: (conditional operator) (条件运算符), 134
 cast (强制类型转换) 144
decltype, 63
 function reference return type (函数引用返回类型), 203
 variable (变量), 472
lvalue reference, see also reference (左值引用, 参见“引用”), **471, 486**
 collapsing rule (折叠规则), 609
 compared to rvalue reference (对比右值引用), 472
 function matching (函数匹配), 477
 initialization (初始化), 472
 member function (成员函数), **484**
 overloaded (重载), 484
move, 472
 template argument deduction (模板实参推断), 608
- ## M
- machine-dependent (机器相关)
 bit-field layout (位域布局), 756
char representation (表示形式), 32
end-of-file character (文件结束符), 13
enum representation (表示形式), 739
iostate, 281
linkage directive language (链接指示语言), 760
nonzero return from main (main函数返回非0值), 203
random IO (随机输入输出), 676
reinterpret_cast, 145
return from exception what (exception what返回), 177
signed out-of-range value (越界值), 33
signed types and bitwise operators (带符号的类型和位运算符), 136
size of arithmetic types (算术类型的大小), 30
terminate function (terminate函数), 175
type_info members (type_info成员), 735
vector, memory management (内存管理), 317
volatile implementation (可变的实现), 670
main, **2, 24**
 not recursive (不允许递归), 204
 parameters (形参), 196
 return type (返回类型), 2
 return value (返回值), 2–4, 203
make_move_iterator, 480
make_pair, 381
make_plural program (make_plural程序), 201
make_shared, 401
make_tuple, 637
malloc library function (标准库函数), **728, 763**
manipulator (操纵符), **6, 24, 666, 681**
 boolalpha, 667
 change format state (改变格式化状态), 667
 dec, 667
 defaultfloat, 670
 endl, 282
 ends, 283
 fixed, 670
 flush, 283
 hex, 667
 hexfloat, 670
 internal, 671
 left, 671
 noboolalpha, 667
 noshowbase, 668
 noshowpoint, 671

- noskipws, 672
nouppercase, 668
oct, 667
right, 671
scientific, 670
setfill, 671
setprecision, 669
setw, 671
showbase, 668
showpoint, 671
skipws, 672
unitbuf, 283
uppercase, 668
map, 374, 397
 see also ordered container (参见“有序容器”)
 * (dereference) (解引用), 382
 [](subscript) (下标), 387, 398
 adds element (添加元素), 388
 at, 387
 definition (定义), 376
 header (头文件), 374
 insert, 383
 key_type requirements (key-type要求), 378
 list initialization (列表初始化), 377
 lower_bound, 390
 map, initialization (初始化), 377
 TextQuery class (TextQuery类), 431
 upper_bound, 390
 word_count program(word_count程序), 375
mapped_type, associative container (关联容器), 381, 397
match (匹配)
 best (最佳), 225
 no (不), 226
match_flag_type, regex_constants, 658
max, 779
max_element, 779
mem_fn, 746, 763
 generates callable (生成可调用对象), 746
member, *see* class data member (成员, 参见“类数据成员”)
member access operators (成员访问运算符), 133
member function (成员函数), 20, 24, 274
 as friend (作为友元), 252
 base member hidden by derived (派生类隐藏的基本成员), 549
 class template (类模板)
 defined outside class body (定义在类外部的), 585
 instantiation (实例化), 587
 const, 231, 273
 () (call operator) (调用运算符), 508
 reference return (引用返回), 247
 declared but not defined (声明但未定义), 451
 defined outside class (定义在类外部的), 232
 definition (定义), 229–233
 :: (scope operator) (作用域运算符), 232
 name lookup (名字查找), 254
 parameter list (形参列表), 253
 return type (返回类型), 254
 explicitly inline (显式内联), 244
 function matching (函数匹配), 245
 implicit this parameter (隐式this形参), 231
 implicitly inline (隐式内联), 231
 inline and header (内联和头文件), 244
 move-enabled (可移动的), 482
 name lookup (名字查找), 257
 overloaded (重载), 245
 on const (在const上), 247
 on lvalue or rvalue reference (左值或右值引用), 484
 overloaded operator (重载运算符), 443, 490
 reference qualified (引用限定符), 483, 487
 returning *this (返回*this), 233, 246
 rvalue reference parameters (右值引用形参), 427
 scope (作用域), 253
 template, *see* member template (模板, 参见“成员模板”)
member template (成员模板), 595, 630
 Blob, iterator constructor (迭代器构造函数), 596
 DebugDelete, 596
 declaration (声明), 596
 defined outside class body (定义在类外部的), 597
 instantiation (初始化), 597
 template parameters (模板参数), 596, 597
memberwise (逐个成员)
 copy assignment (拷贝赋值), 443
 copy constructor (拷贝构造函数), 440
 copy control (拷贝控制), 239, 486
 destruction is implicit (析构是隐式的), 446
 move assignment (移动赋值), 476
 move constructor (移动构造函数), 476
memory (内存)
 see also dynamically allocated (参见“动态分配”)
 exhaustion (耗尽), 408
 leak (泄漏), 410
memory header (memory头文件), 400, 401, 427, 429
merge, 774
 list and forward_list (list和forward_list), 369
Message, 460–464
 add_to_Folder, 462
 class definition (类定义), 461
 copy assignment (拷贝赋值), 463

- copy constructor** (拷贝构造函数), 462
design (设计), 460
destructor (析构函数), 462
move assignment (移动赋值), 479
move constructor (移动构造函数), 479
move_Folders, 479
remove_from_Folders, 463
method, *see* member function (方法, 参见“成员函数”)
Microsoft compiler (微软编译器), 4
min, 779
min_element, 779
minmax, 779
minus<T>, 510
mismatch, 772
mode, file (文件模式), 290
modulus<T>, 510
move, **469, 472**, 774
 argument-dependent lookup (实参相关的查找), 706
 binds rvalue reference to lvalue (将右值引用绑定到左值), 471
 explained (解释), 610–612
 inherently dangerous (固有的危险), 481
Message, move operations (移动操作), 478
 moved from object has unspecified value (不能对移后源对象的值做任何假设), 472
 reference collapsing rule (引用折叠准则), 611
StrVec reallocate, 469
remove_reference, 611
move assignment (移动赋值), **474, 487**
 copy and swap (拷贝和交换), 478
 derived class (派生类), 555
HasPtr, valuelike (类值), 478
 memberwise (逐个成员), 476
Message, 479
 moved-from object destructible (源对象在移动后应该是可析构的), 475
noexcept, 473
rule of three/five, virtual destructor exception (三/五法则, 虚析构函数异常), 552
 self-assignment (自赋值), 475
StrVec, 474
 synthesized (合成的)
 deleted function (删除的函数), 476, 553
 derived class (派生类), 552
 multiple inheritance (多重继承), 712
 sometimes omitted (有时被忽略), 476
move constructor (移动构造函数), **469, 473, 473–474**, 487
 and copy initialization (和拷贝初始化), 478
 derived class (派生类), 555
HasPtr, valuelike (类值), 478
 memberwise (逐个成员), 476
Message, 479
 moved-from object destructible (源对象在移动后应该是可析构的), 475
noexcept, 473
rule of three/five, virtual destructor exception (三/五法则, 虚析构函数异常), 552
String, 468
StrVec, 473
 synthesized (合成的)
 deleted function (删除的函数), 553
 derived class (派生类), 552
 multiple inheritance (多重继承), 712
 sometimes omitted (有时被忽略), 476
move iterator (移动迭代器), 358, 372, **480, 487**
make_move_iterator, 480
StrVec, reallocate, 480
uninitialized_copy, 480
move operations (移动操作), 471–485
 function matching (函数匹配), 477
move, 472
noexcept, 473
 rvalue references (右值引用), 471
 valid but unspecified (有效但未指明的), 475
move_backward, 775
move_Folders, Message, 479
multidimensional array (多维数组), 112–116
 [](subscript) (下标), 113
 argument and parameter (实参和形参), 196
begin, 115
 conversion to pointer (转换为指针), 114
 definition (定义), 112
end, 115
 initialization (初始化), 113
 pointer (指针), 114
 range for statement and (范围for语句和), 114
multimap, 397
see also ordered container (参见“有序容器”)
 * (dereference) (解引用), 382
 definition (定义), 376
 has no subscript operator (不含下标运算符), 387
insert, 383, 386
key_type requirements (需求), 378
 list initialization (列表初始化), 377
lower_bound, 390
map, initialization (初始化), 377
upper_bound, 390
multiple inheritance (多重继承), 710, 723
see also virtual base class (参见“虚基类”)
 = (assignment) (赋值), 712
 ambiguous conversion (二义性转换), 713
 ambiguous names (二义性名字), 715

avoiding ambiguities (避免二义性), 716
 class derivation list (类派生列表), 711
 conversion (类型转换), 712
 copy control (拷贝控制), 712
 name lookup (名字查找), 715
 object composition (对象组合), 711
 order of initialization (初始化顺序), 711
 scope (作用域), 715
 virtual function (虚函数), 714

multiplies<T>, 510

multiset, 397

see also ordered container (参见“有序容器”)

 insert, 386

 iterator, 382

 key_type requirements (key-type要求), 378

 list initialization (列表初始化), 377

 lower_bound, 390

 override comparison (覆盖比较)

 Basket class (Basket类), 559

 using compareIsbn (使用compareIsbn), 379

 upper_bound, 390

 used in Basket (在Basket中使用), 560

mutable

 data member (数据成员), 245

 lambda expression (lambda表达式), 352

N

\n (newline character) (换行符), 36

name lookup (名字查找), 254, 274

 :: (scope operator), overrides (作用域运算符, 覆盖), 256

 argument-dependent lookup (实参相关的查找), 706

 before type checking (类型检查之前), 549

 multiple inheritance (多重继承), 716

block scope (块作用域), 43

class (类), 254

class member (类成员)

 declaration (声明), 254

 definition (定义), 255, 257

 function (函数), 254

depends on static type (由静态类型决定), 547, 549

 multiple inheritance (多重继承), 713

derived class (派生类), 547

 name collisions (名字冲突), 548

local class (局部类), 754

multiple inheritance (多重继承), 715

ambiguous names (二义性名字), 715

namespace (命名空间), 705

 nested class (嵌套类), 748

 overloaded virtual functions (重载虚函数), 551

 templates (模板), 582

 type checking (类型检查), 210

 virtual base class (虚基类), 718

named cast (命名的强制类型转换), 144

 const_cast, 144, 144

 dynamic_cast, 144, 730

 reinterpret_cast, 145, 145

 static_cast, 145, 145

namespace (命名空间), 7, 24, 695, 723

 alias (别名), 702, 723

 argument-dependent lookup (实参相关的查找), 706

 candidate function (候选函数), 708

 cplusplus_primer, 697

 definition (定义), 695

 design (设计), 696

 discontiguous definition (不连续的定义), 696

 friend declaration scope (友元声明作用域), 707

 function matching (函数匹配), 708

 global (全局的), 698, 723

 inline (内联), 699, 723

 member (成员), 696

 member definition (成员定义), 698

 outside namespace (命名空间之外), 698

 name lookup (名字查找), 705

 nested (嵌套的), 698

 overloaded function (重载函数), 708

 placeholders, 355

 scope (作用域), 695–700

 std, 7

 template specialization (模板特例化), 626, 698

 unnamed (未命名的), 700, 724

 local to file (文件局部), 700

 replace file static (替换文件中的静态声明), 620

 namespace pollution (命名空间污染), 695, 723

 narrowing conversion (缩减的类型转换), 39

NDEBUG, 216

negate<T>, 510

nested class (嵌套类), 746, 763

 access control (访问控制), 747

 class defined outside enclosing class (定义在外层类之外的类), 748

constructor, QueryResult (构造函数), 748

in local class (在局部类中), 756

member defined outside class body (定义在类外的成员), 748

name lookup (名字查找), 748

QueryResult, 747

relationship to enclosing class (与外层类的关系),

- 747, 749
 scope (作用域), 747
 static member (静态成员), 748
 nested namespace (嵌套的命名空间), 698
 nested type, *see* nested class (嵌套类型, 参见“嵌套类”)
 new, 407, 407–408, 436
 execution flow (执行流程), 726
 failure (失败), 408
 header (头文件), 176, 409, 424, 727
 initialization (初始化), 407
 placement (定位), 409, 436, 729, 763
 union with class type member (含有类类型成员的union), 753
 shared_ptr, 412
 unique_ptr, 417
 with auto (和auto), 408
 new[], 423, 423–424
 initialization (初始化), 423
 returns pointer to an element (返回指向元素的指针), 423
 value initialization (值初始化), 424
 newline (\n), character (换行符), 36
 next_permutation, 778
 no match (没有匹配的函数), 209, 226
 see also function matching (参见“函数匹配”)
 noboolalpha, manipulator (操纵符), 667
 NoDefault, 262
 noexcept
 exception specification (异常说明), 690, 723
 argument (实参), 690–692
 violation (违反), 690
 move operations (移动操作), 473
 operator (运算符), 691, 723
 nonconst reference, *see* reference (非常量引用, 参见“引用”)
 none_bitset, 643
 none_of, 771
 nonportable (不可移植), 33, 763
 nonprintable character (不可打印的字符), 36, 70
 nonthrowing function (不抛出异常的函数), 690, 723
 nontype parameter (非类型参数), 580, 630
 compare, 580
 must be constant expression (必须是常量表达式), 581
 type requirements (类型需求), 580
 normal_distribution, 664
 noshowbase, manipulator (操纵符), 668
 noshowpoint, manipulator (操纵符), 671
 noskipws, manipulator (操纵符), 672
 not_equal_to<T>, 510
 NotQuery, 564
 class definition (类定义), 568
 eval function (eval函数), 573
 nouppercase, manipulator (操纵符), 668
 nth_element, 776
 NULL, 48
 null (\0), character (空字符), 36
 null pointer (空指针), 48, 70
 delete of, 409
 null statement (空语句), 154, 179
 null-terminated character string, *see* C-style string (未终止的字符串, 参见“C风格字符串”)
 nullptr, 48, 70
 numeric header (numeric头文件), 336, 779
 numeric conversion, to and from string (数值类型转换为string以及string转换为数值类型), 327
 numeric literal (数字字面值常量)
 float (numF or numf), 37
 long (numL or numl), 37
 long double (ddd.dddL or ddd.ddd1), 37
 long long (numLL or numl1), 37
 unsigned (numU or numu), 37
- ## O
- object (对象), 39, 70
 automatic (自动的), 185, 225
 dynamically allocated (动态分配), 407–412
 const object (常量对象), 409
 delete, 409
 factory program (factory程序), 409
 initialization (初始化), 408
 lifetime (生命周期), 400
 new, 407
 lifetime (生命周期), 184, 226
 local static (局部静态), 185, 226
 order of destruction (析构顺序)
 class type object (类类型对象), 445
 derived class object (派生类对象), 556
 multiple inheritance (多重继承), 712
 virtual base classes (虚基类), 720
 order of initialization (初始化顺序)
 class type object (类类型对象), 259
 derived class object (派生类对象), 531, 552
 multiple inheritance (多重继承), 712
 virtual base classes (虚基类), 720
 object code (对象代码), 226
 object file (对象文件), 187, 226
 object-oriented programming (面向对象程序设计), 576
 oct, manipulator (操纵符), 667
 octal, literal (onum) (八进制字面值常量), 35
 octal escape sequence (\nnn) (八进制转义序列), 36
 off-the-beginning iterator (首前迭代器), 313, 333
 before_begin, 313

forward_list, 313
off-the-end (尾后)
 iterator (迭代器), 95, 118, 333
 iterator range (迭代器范围), 296
 pointer (指针), 105
ofstream, 279, 283–287, 290
 see also ostream (参见ostream)
 close, 284
 file marker (文件标记), 676
 file mode (文件模式), 286
 initialization (初始化), 284
 off_type, 677
 open, 284
 pos_type, 677
 random access (随机访问), 676
 random IO program (随机输入输出程序), 677
 seek and tell (seek和tell), 675–679
old-style, cast (旧式强制类型转换), 146
open, file stream (文件流), 284
operand (运算对象), 120, 150
 conversion (类型转换), 141
operator (运算符), 120, 150
operator alternative name (运算符可选名字), 42
operator delete
 class member (类成员), 728
 global function (全局函数), 726, 763
operator delete[]
 class member (类成员), 728
 compared to deallocate (对比deallocate),
 728
 global function (全局函数), 726
operator new
 class member (类成员), 728
 global function (全局函数), 726, 763
operator new[]
 class member (类成员), 727
 compared to allocate (对比allocate), 728
 global function (全局函数), 726
operator overloading, *see* overloaded operator (运算符重载, 参见“重载运算符”)
operators (运算符)
 arithmetic (算术), 124
 assignment (赋值), 11, 129–131
 binary (二元), 120, 149
 bitwise (位), 135–139
 bitset, 643
 comma (,) (逗号), 140
 compound assignment (复合赋值), 11
 conditional (? :) (条件), 134
 decrement (递减), 131–133
 equality (相等), 16, 125
 increment (递增), 11, 131–133
 input (输入), 7
 iterator (迭代器)
 addition and subtraction (加法和减法), 99
 arrow (箭头), 98
 dereference (解引用), 96
 equality (相等), 95, 97
 increment and decrement (递增和递减), 96
 relational (关系), 99
 logical (逻辑), 125
 member access (成员访问), 133
 noexcept, 691
 output (输出), 6
 overloaded, arithmetic (重载的, 算术), 497
 pointer (指针)
 addition and subtraction (加法和减法), 106
 equality (相等), 96
 increment and decrement (递增和递减), 105
 relational (关系), 107, 110
 subscript (下标), 108
 relational (关系), 11, 125, 128
Sales_data
 += (compound assignment) (复合赋值), 443
 + (addition) (加法), 497
 == (equality) (相等), 498
 != (inequality) (不相等), 498
 >> (input operator) (输入运算符), 495
 << (output operator) (输出运算符), 494
Sales_item, 17
scope (作用域), 74
sizeof, 139
sizeof..., 619
string
 addition (加法), 80
 equality and relational (相等性和关系), 80
 IO (输入输出), 77
 subscript (下标), 84–86
subscript (下标), 103
typeid, 732, 763
unary (一元), 120, 150
vector
 equality and relational (相等性和关系), 92
 subscript (下标), 92–94
options to main (main选项), 196
order of destruction (析构顺序)
 class type object (类类型对象), 445
 derived class object (派生类对象), 556
 multiple inheritance (多重继承), 712
 virtual base classes (虚基类), 721
order of evaluation (求值顺序), 120, 150
 && (logical AND) (逻辑与), 123
 || (logical OR) (逻辑或), 123
 , (comma operator) (逗号运算符), 123

- ? : (conditional operator) (条件运算符), 123
 expression (表达式), 123
 pitfalls (误区), 132
order of initialization (初始化顺序)
 class type object (类类型对象), 259
 derived class object (派生类对象), 531
 multiple base classes (多重基类), 723
 multiple inheritance (多重继承), 712
 virtual base classes (虚基类), 720
ordered container (有序容器)
see also container (参见“容器”)
see also associative container (参见“关联容器”)
 key_type requirements (需求), 378
 lower_bound, 390
 override default comparison (覆盖默认比较操作), 378
 upper_bound, 390
ordering, strict weak (严格弱序), 378, 398
OrQuery, 564
 class definition (类定义), 570
 eval function (eval函数), 571
ostream, 5, 24, 279
see also manipulator (参见“操作符”)
 << (output operator) (输出运算符), 6
 precedence and associativity (优先级和结合律), 138
 chained output (链式输出), 6
 condition state (条件状态), 280
 explicit conversion to bool (显式转换为 bool), 516
 file marker (文件标记), 676
 floating-point notation (浮点数符号), 670
 flushing output buffer (刷新输出缓冲), 282
 format state (格式化状态), 666
 no copy or assign (不允许拷贝或赋值), 279
 not flushed if program crashes (如果程序崩溃则不刷新), 283
 off_type, 677
 output format, floating-point (输出格式, 浮点), 668
 pos_type, 677
 precision member (precision成员), 669
 random access (随机访问), 676
 random IO program (随机输入输出程序), 677
 seek and tell (seek和tell), 676–679
 tie member (tie成员), 283
 virtual base class (虚基类), 717
 write, 676
ostream_iterator, 359, 372
 << (output operator) (输出运算符), 361
 algorithms (算法), 360
 initialization (初始化), 361
 operations (操作), 361
 type requirements (类型需求), 362
ostringstream, 279, 287, 287–289
see also ostream (参见ostream)
 file marker (文件标记), 676
 initialization (初始化), 287
 off_type, 677
 phone number program (电话号码程序), 289
 pos_type, 677
 random access (随机访问), 676
 random IO program (随机输入输出程序), 677
 seek and tell, 676–679
 str, 289
out (file mode) (文件模式), 286
 out-of-range value (越界值), signed, 33
out_of_range, 176
 at function (at函数), 310
out_of_stock, 694
 outer scope (外层作用域), 44, 70
 output, standard (标准输出), 5
 output iterator (输出迭代器), 366, 372
 overflow (溢出), 125
overflow_error, 176
 overhead, function call (函数调用开销), 213
 overload resolution, *see* function matching (重载解析, 参见“函数匹配”)
 overloaded function (重载函数), 206, 206–210, 226
see also function matching (参见“函数匹配”)
 as friend (作为友元), 252
 compared to redeclaration (对比重新声明), 207
 compared to template specialization (对比模板特例化), 625
 const parameters (常量形参), 208
 constructor (构造函数), 235
 function template (函数模板), 614–619
 linkage directive (链接指示), 761
 member function (成员函数), 245
 const, 247
 move-enabled (可移动的), 482
 reference qualified (引用限定), 484
 virtual (虚), 551
 move-enabled (可移动的), 482
 namespace (命名空间), 708
 pointer to (指向), 221
 scope (作用域), 210
 derived hides base (派生类隐藏基类), 549
using declaration (using声明), 708
 in derived class (在派生类中), 551
 using directive (using指示), 709
 overloaded operator (重载运算符), 120, 150, 443, 487, 490, 523
 ++ (increment) (递增), 502–504

- (decrement) (递减), 502–504
 - * (dereference) (解引用), 504
 - StrBlobPtr, 504
 - & (address-of) (取地址), 491
 - > (arrow operator) (箭头运算符), 504
 - StrBlobPtr, 504
 - [](subscript) (下标), 501
 - StrVec, 501
 - () (call operator) (调用运算符), 506
 - absInt, 506
 - PrintString, 507
 - = (assignment) (赋值), 443, 499
 - StrVec initializer_list, 499
 - +=(compound assignment) (复合赋值), 493, 497
 - Sales_data, 443
 - + (addition) (加法), Sales_data, 497
 - = (equality) (相等), 497
 - Sales_data, 497
 - != (inequality) (不相等), 498
 - Sales_data, 497
 - < (less-than), strict weak ordering (小于, 严格弱序), 498
 - >> (input operator) (输入运算符), 495–496
 - Sales_data, 495
 - << (output operator) (输出运算符), 494–495
 - Sales_data, 494
 - && (logical AND) (逻辑与), 491
 - || (logical OR) (逻辑或), 491
 - & (bitwise AND) (位与), Query, 570
 - | (bitwise OR) (位或), Query, 571
 - ~ (bitwise NOT) (位取反), Query, 569
 - , (comma operator) (逗号运算符), 491
 - ambiguous (二义性), 521
 - arithmetic operators (算术运算符), 497
 - associativity (结合律), 490
 - binary operators (二元运算符), 490
 - candidate function (候选函数), 521
 - consistency between relational and equality operators (关系运算符与相等性运算符的一致性), 498
 - definition (定义), 443, 490
 - design (设计), 491–493
 - equality operators (相等性运算符), 497
 - explicit call to (显式调用), 491
 - postfix operators (后置运算符), 503
 - function matching (函数匹配), 520–522
 - member function (成员函数), 443, 490
 - member vs. nonmember function (成员与非成员函数), 490, 492
 - precedence (优先级), 490
 - relational operators (关系运算符), 498
 - requires class-type parameter (需要类类型形参), 490
 - short-circuit evaluation lost (丢失短路求值功能), 491
 - unary operators (一元运算符), 490
 - override, virtual function (覆盖, 虚函数), 528, 576
 - override specifier (override说明符), 527, 530, 538
- ## P
- pair, 379, 398
 - default initialization (默认初始化), 380
 - definition (定义), 379
 - initialization (初始化), 379
 - list initialization (列表初始化), 380, 384, 467
 - make_pair, 381
 - map, * (dereference) (解引用运算符), 382
 - operations (操作), 380
 - public data members (公有数据成员), 380
 - return value (返回值), 467
 - Panda, 710
 - parameter (形参), 182, 187, 226
 - array (数组), 193–197
 - buffer overflow (缓冲区溢出), 193
 - to pointer conversion (转换为指针), 193
 - C-style string (C风格字符串), 194
 - const, 190
 - copy constructor (拷贝构造函数), 440
 - ellipsis (省略符), 199
 - forwarding (转发), 612
 - function pointer, linkage directive (函数指针, 链接指示), 759
 - implicit this (隐式this), 231
 - initialization (初始化), 183, 187
 - iterator (迭代器), 193
 - lifetime (生命周期), 184
 - low-level const (底层const), 191
 - main function (main函数), 196
 - multidimensional array (多维数组), 195
 - nonreference (非引用), 188
 - uses copy constructor (使用拷贝构造函数), 441
 - uses move constructor (使用移动构造函数), 476
 - pass by reference (传引用), 188, 226
 - pass by value (传值), 189, 226
 - passing (参数传递), 187–191
 - pointer (指针参数), 189, 193
 - pointer to const (const的指针), 220
 - pointer to array (数组指针), 195
 - pointer to function (函数指针), 223

- linkage directive (链接指示), 759
- reference (引用参数), 188–192
 - to const (转换为const), 191, 220
 - to array (转换为数组), 195
- reference to const (const的引用), 190
- template, *see* template parameter (模板, 参见“模板参数”)
- top-level const (顶层const), 190
- parameter list (参数列表)
 - function (函数), 2, 24, 182
 - template (模板), 578, 631
- parameter pack (参数包), 631
 - expansion (扩展), 621, 621–623, 631
 - function template (函数模板), 630
 - sizeof..., 619
 - variadic template (可变参数模板), 618
- parentheses, override precedence (括号, 覆盖优先级), 122
- partial_sort, 776
- partial_sort_copy, 776
- partial_sum, 780
- partition, 775
- partition_copy, 775
- partition_point, 775
- pass by reference (传引用), 187, 188, 226
- pass by value (传值), 189, 226
 - uses copy constructor (使用拷贝构造函数), 441
 - uses move constructor (使用移动构造函数), 476
- pattern (模式), 621, 631
 - function parameter pack (函数参数包), 622
 - regular expression, phone number (正则表达式, 电话号码), 654
 - template parameter pack (模板参数包), 621
- peek, istream, 673
- PersonInfo, 288
- phone number, regular expression (电话号码, 正则表达式)
 - program (程序), 654
 - reformat program (重排格式程序), 657
 - valid, 655
- pitfalls (误区)
 - dynamic memory (动态内存), 410
 - order of evaluation (求值顺序), 133
 - self-assignment (自赋值), 454
 - smart pointer (智能指针), 417
- using directive (using指示), 704
- placeholders, 355
- placement new (定位new), 409, 436, 729, 763
 - union, class type member(union, 类类型成员), 752
- plus<T>, 510
- pointer (指针), 47, 47–48, 70
 - ++ (increment) (递增运算符), 131
 - (decrement) (递减运算符), 131
 - * (dereference) (解引用运算符), 48
 - [](subscript) (下标运算符), 108
 - = (assignment) (赋值运算符), 49
 - + (addition) (加法运算符), 106
 - (subtraction) (减法运算符), 106
 - == (equality) (相等运算符), 50, 109
 - != (inequality) (不等运算符), 50, 106
 - and array (和数组), 105
 - arithmetic (算术运算), 106, 118
 - const, 56, 69
 - const pointer to const (const的const指针), 56
 - constexpr, 59
 - conversion (类型转换)
 - from array (从数组转换), 143
 - to bool (转换为bool), 144
 - to const (转换为const), 56, 144
 - to void* (转换为void*), 143
 - dangling (空悬指针), 411, 436
 - declaration style (声明风格), 51
 - definition (定义), 47
 - delete, 409
 - derived-to-base conversion (派生类到基类的转换), 530
 - under multiple inheritance (多重继承下的转换), 712
 - dynamic_cast, 730
 - implicit this (隐式this), 231, 274
 - initialization (初始化), 47–49
 - invalid (无效指针), 47
 - multidimensional array (多维数组), 114
 - null (空指针), 48, 70
 - off-the-end (尾后指针), 105
 - parameter (形参), 189, 192
 - relational operators (关系运算符), 109
 - return type (返回类型), 184
 - return value, local variable (返回值, 局部变量), 201
 - smart (智能指针), 400, 436
 - to const (const的指针), 56
 - and typedef (和类型别名), 60
 - to array (数组的指针)
 - parameter (形参), 196
 - return type (返回类型), 184
 - return type declaration (返回类型声明), 205
 - to const (const的指针), 69
 - overloaded parameter (重载参数), 208, 220
 - to pointer (指针的指针), 52
 - typeid operator (typeid运算符), 733
 - valid (有效指针), 48

- volatile**, 757
pointer to function (函数指针), 221–223
 auto, 223
 callable object (可调用对象), 346
 decltype, 222
 exception specification (异常说明), 690, 692
 explicit template argument (显式模板实参), 607
 function template instantiation (函数模板实例化),
 607
 linkage directive (链接指示), 759
 overloaded function (重载函数), 222
 parameter (形参), 222
 return type (返回类型), 184, 222
 using **decltype** (使用**decltype**), 223
 template argument deduction (模板实参推断), 607
 trailing return type (尾置返回类型), 223
 type alias (类型别名), 222
 typedef, 222
pointer to member (成员指针), 739, 763
 arrow (\rightarrow^*) (箭头运算符), 740
 definition (定义), 740
 dot (\cdot^*) (点运算符), 740
 function (函数), 741
 and bind (和bind), 746
 and function (和function), 746
 and mem_fn (和mem_fn), 746
 not callable object (不是可调用对象), 745
 function call (函数调用), 742
 function table (函数表), 743
 precedence (优先级), 742
polymorphism (多态), 537, 576
pop
 priority_queue, 330
 queue, 330
 stack, 330
pop_back
 sequential container (顺序容器), 311
 StrBlob, 406
pop_front, sequential container (顺序容器的)
 pop_front, 311
portable (可移植), 755
precedence (优先级), 120, 121–123, 150
 = (assignment) (赋值运算符), 130
 ?: (conditional operator) (条件运算符), 134
 assignment and relational operators (赋值和关系运
 算符), 130
 dot and dereference (点和解引用运算符), 133
 increment and dereference (递增和解引用运算符),
 132
 of IO operator (IO运算符的优先级), 138
 overloaded operator (重载运算符), 491
 parentheses overrides (括号覆盖), 122
pointer to member and call operator (成员指针和
 调用运算符), 741
precedence table (优先级表), 147
precision member, ostream (ostream的
 precision成员), 669
predicate (谓词), 344, 372
 binary (二元), 344, 371
 unary (一元), 344, 372
prefix, smatch, 652
preprocessor (预处理器), 68, 70
 #include, 6
 assert macro (**assert**宏), 215, 225
 header guard (头文件保护符), 68
 variable (变量), 49, 71
prev_permutation, 778
print, Sales_data, 234
print program (**print**程序)
 array parameter (数组参数), 193
 array reference parameter (数组引用参数), 193
 pointer and size parameters (指针和大小参数),
 195
 pointer parameter (指针参数), 194
 two pointer parameters (两个指针参数), 194
 variadic template (可变参数模板), 620
print_total
 explained (解释), 536
 program (程序), 527
PrintString, 507
 () (call operator) (调用运算符), 506
priority_queue, 330, 333
 emplace, 330
 empty, 330
 equality and relational operators (相等和关系运算
 符), 329
 initialization (初始化), 329
 pop, 330
 push, 330
 sequential container (顺序容器), 330
 size, 330
 swap, 330
 top, 330
private
 access specifier (访问说明符), 240, 273
 copy constructor and assignment (拷贝构造函数和
 赋值运算符), 451
 inheritance (继承), 543, 576
program (程序)
 addition (加法程序)
 Sales_data, 66
 Sales_item, 18, 20
 alternative_sum, 604
 biggies, 348

binops desk calculator (*binops*桌面计算器), 511
book from author version 1 (查找作者书籍程序版本1), 389
book from author version 2 (查找作者书籍程序版本2), 390
book from author version 3 (查找作者书籍程序版本3), 391
bookstore (书店程序)
 Sales_data, 229
 Sales_data using algorithms (使用算法), 362
 Sales_item, 21
buildMap, 392
children's story (儿童故事程序), 342–349
compare, 578
count_calls, 185
debug_rep
 additional nontemplate versions (另一个非模板版本), 617
 general template version (通用模板版本), 615
 nontemplate version (非模板版本), 617
 pointer template version (指针模板版本), 616
elimDups, 343–346
error_msg, 198
fact, 182
factorial, 204
factory
 new, 409
 shared_ptr, 402
file extension (文件扩展名), 647
 version 2 (版本2), 654
find last word (查找最后一个单词的程序), 364
find_char, 189
findBook, 639
flip, 614
flip1, 612
flip2, 613
grade clusters (成绩分类程序), 92
grading (成绩统计程序)
 bitset, 644
 bitwise operators (位运算), 137
i before e (查找*i*在*e*之前的文本的程序), 646
 version 2 (版本2), 651
isShorter, 189
letter grade (字母成绩统计程序), 157
make_plural, 201
message handling (消息处理), 460
phone number (电话号码程序)
 istringstream, 287
 ostringstream, 289
 reformat (重排格式), 658
regular expression version (正则表达式版本), 654
valid, 655
print
 array parameter (数组参数), 193
 array reference parameter (数组引用参数), 195
 pointer and size parameters (指针和大小参数), 195
 pointer parameter (指针参数), 194
 two pointer parameters (两个指针参数), 194
variadic template (可变参数模板), 620
print_total, 527
Query, 563
 class design (类定义), 563–566
random IO (随机IO程序), 677
reset
 pointer parameters (指针参数), 188
 reference parameters (引用参数), 189
restricted word_count (限制的word_count程序), 375
sum, 604
swap, 200
TextQuery, 431
 design (设计), 430
transform, 393
valid, 655
vector capacity (*vector*容量程序), 318
vowel counting (元音计数程序), 160
word_count
 map, 375
 unordered_map, 394
word_transform, 392
ZooAnimal, 710
promotion, see *integral promotion* (提升, 参见“整形提升”)
protected
 access specifier (访问说明符), 529, 542, 575
 inheritance (继承), 543, 576
 member (成员), 542
ptr_fun deprecated (*ptr_fun*已弃用), 357
ptrdiff_t, 107, 118
public
 access specifier (访问说明符), 240, 273
 inheritance (继承), 543, 576
pure virtual function (纯虚函数), 540, 576
 Disc_quote, 540
 Query_base, 564
push
 priority_queue, 330
 queue, 330
 stack, 330

- push_back**
 back_inserter, 341, 358
 sequential container (顺序容器), 88, 118, 306
 move-enabled (可移动的), 482
 strVec, 465
 move-enabled (可移动的), 482
push_front
 front_inserter, 358
 sequential container (顺序容器), 306
put, istream, 673
putback, istream, 673
- Q**
- Query**, 565
 << (output operator) (输出运算符), 568
 & (bitwise AND) (位与运算符), 565
 definition (定义), 570
 | (bitwise OR) (位或运算符), 565
 definition (定义), 569
 ~ (bitwise NOT) (位非运算符), 565
 definition (定义), 566
 classes (Query类), 563–566
 definition (定义), 567
 interface class (接口类), 564
 operations (操作), 562
 program (Query程序), 563
 recap (概述), 566
- Query_base**, 564
 abstract base class (抽象基类), 564
 definition (定义), 567
 member function (成员函数), 564
- QueryResult**, 431
 class definition (类定义), 434
 nested class (嵌套类), 747
 constructor (构造函数), 748
 print, 434
- queue**, 330, 333
 back, 330
 emplace, 330
 empty, 330
 equality and relational operators (相等和关系运算符), 330
 front, 330
 header (头文件), 330
 initialization (初始化), 329
 pop, 330
 push, 330
 sequential container (顺序容器), 329
 size, 330
 swap, 329
- Quote**
 class definition (类定义), 527
 design (设计), 526
- R**
- Raccoon**, virtual base class (Raccoon虚基类), 718
raise exception, see **throw** (抛出异常, 参见**throw**)
rand function, drawbacks (**rand**函数, 缺点), 660
random header (**random**头文件), 660
random IO (随机IO), 676
 machine-dependent (机器相关的), 677
 program (随机IO程序), 676
random-access iterator (随机访问迭代器), 366, 372
random-number library (随机数库), 660
 compared to **rand** function (与**rand**函数对比), 660
 distribution types (分布类型), 660, 681
 engine (引擎), 660, 681
 default_random_engine, 660
 max, min, 661
 retain state (保持状态), 662
 seed, 662, 681
 generator (生成器), 661, 681
 range (范围), 661
random_shuffle, 777
range for statement (范围for语句), 82, 118, 168, 168–170, 179
 can't add elements (不能添加元素), 91, 169
 multidimensional array (多维数组), 114
 not with dynamic array (不能和动态数组一起使用), 423
range_error, 176
rbegin, container (容器的**rbegin**操作), 298, 363
rdstate, stream (流的**rdstate**操作), 281
read
 istream, 676
 Sales_data, 234
reallocate, StrVec, 469
 move iterator version (移动迭代器版本), 480
recursion loop (递归循环), 204, 226, 539
recursive function (递归函数), 204, 226
 variadic template (可变参数模板), 620
ref, binds reference parameter (**ref**绑定引用参数), 356, 372
refactoring (重构), 542, 576
reference (引用), 45, 70
 see also lvalue reference (参见“左值引用”)
 see also rvalue reference (参见“右值引用”)
 auto deduces referred to type (**auto**推断引用的类型), 61
 collapsing rule (折叠规则), 608

- forward, 614
- lvalue arguments (左值实参), 608
- move, 611
- rvalue reference parameters (右值引用参数), 613
- const, see reference to const (参见“const的引用”)
- conversion (类型转换)
 - not from const (不从const转换), 55
 - to reference to const (转换为const的引用), 144
- data member, initialization (数据成员, 初始化), 259
- declaration style (声明风格), 51
- decltype yields reference type (decltype生成引用类型), 63
- definition (定义), 46
- derived-to-base conversion (派生类到基类的转换), 530
 - under multiple inheritance (多重继承下的转换), 712
- dynamic_cast operator (dynamic_cast运算符), 731
- initialization (初始化), 46
- member function (成员函数), 483
- parameter (形参), 188–192
 - bind, 356
 - limitations (限制), 192
 - template argument deduction (模板实参推断), 608–610
 - remove_reference, 605
- return type (返回类型), 201
 - assignment operator (赋值运算符), 443
 - is lvalue (返回类型是左值), 202
 - return value, local variable (返回值, 局部变量), 201
 - to array parameter (数组参数的引用), 195
- reference, container (容器的reference), 298
- reference count (引用计数), 402, 436, 455, 487
 - copy assignment (拷贝赋值操作), 455
 - copy constructor (拷贝构造函数), 455
 - design (设计), 455
 - destructor (析构函数), 455
 - HasPtr class (HasPtr类), 455–457
- reference to const (const的引用), 54, 70
 - argument (实参), 189
 - initialization (初始化), 55
 - parameter (形参), 189, 191
 - overloaded (重载), 208, 220
 - return type (返回类型), 203
- regex, 645, 681
 - error_type, 649
 - header (头文件), 645
- regex_error, 648, 681
- syntax_option_type, 647
- regex_constants, 658
 - match_flag_type, 658
- regex_error, 648, 681
- regex_match, 646, 681
- regex_replace, 657, 681
 - format flags (格式标志), 659
 - format string (格式字符串), 657
- regex_search, 646, 647, 681
- regular expression library (正则表达式库), 645, 681
 - case sensitive (大小写敏感), 647
 - compiled at run time (运行时编译), 648
 - ECMAScript, 647
 - file extension program (文件扩展名程序), 647
 - i before e program (查找i在e之前的文本的程序), 646
 - version 2 (版本2), 651
 - match data (匹配数据), 652
 - pattern (模式), 646
 - phone number (电话号码), valid, 655
 - phone number pattern (电话号码模式), 654
 - phone number program (电话号码程序), 654
 - phone number reformat, program (电话号码重排格式程序), 657
 - regex iterators (regex迭代器), 650
 - search functions (搜索函数), 657
 - smatch, provides context for a match (smatch, 为匹配提供了上下文), 652
 - subexpression (子表达式), 654
 - file extension program version 2 (文件扩展名程序版本2), 654
 - types (类型), 649
 - valid, program (valid程序), 655
- reinterpret_cast, 145, 145
 - machine-dependent (机器相关的), 146
- relational operators (关系运算符), 126, 127
 - arithmetic conversion (算术转换), 128
- container adaptor (容器适配器), 329
- container member (容器成员), 304
- function object (函数对象), 509
- iterator (迭代器), 99
- overloaded operator (重载运算符), 498
- pointer (指针), 107, 109
 - Sales_data, 499
 - string, 80
 - tuple, 638
 - vector, 92
- release, unique_ptr, 418
- remove, 777
 - list and forward_list (list和forward_list), 369

remove_copy, 777
remove_copy_if, 777
remove_from_Folders, Message, 463
remove_if, 777
 list and forward_list (list和forward_list), 369
remove_pointer, 606
remove_reference, 605
 move, 611
rend, container (容器), 298, 363
replace, 342, 774
 string, 322
replace_copy, 342, 774
replace_copy_if, 774
replace_if, 774
reserve
 string, 318
 vector, 318
reserved identifiers (保留标识符), 42
reset
 bitset, 644
 shared_ptr, 455, 471
 unique_ptr, 417
reset program (reset程序)
 pointer parameters (指针参数), 189
 reference parameters (引用参数), 188
resize
 invalidates iterator (使迭代器无效), 314
 sequential container (顺序容器), 314
 value initialization (值初始化), 314
restricted word_count program (受限制的word_count程序), 375
result (结果), 120, 150
 * (dereference), lvalue (解引用运算符, 左值), 121
 [] (subscript), lvalue (下标运算符, 左值), 121
 , (comma operator), lvalue (逗号运算符, 左值), 140
 ?: (conditional operator), lvalue (条件运算符, 左值), 134
 cast, lvalue (类型转换, 左值), 144
rethrow (重新抛出), 688
 exception object (异常对象), 688
 throw, 688, 723
return statement (return语句), 199, 199–204
 from main (从main返回), 203
 implicit return from main (从main隐式返回), 200
 local variable (局部变量), 201, 202
return type (返回类型), 2, 24, 182, 184, 226
 array (数组), 184
 array using decltype (使用decltype返回数组), 206
function (函数), 184
function pointer (函数指针), 222
 using decltype (使用decltype), 223
linkage directive (链接指示), 760
main, 2
member function (成员函数), 253
nonreference (非引用), 201
 copy initialized (拷贝初始化), 442
pointer (指针), 184
pointer to function (函数的指针), 184
reference (引用), 201
reference to const (const的引用), 201
reference yields lvalue (引用生成左值), 202
trailing (尾置返回类型), 206, 226, 353, 605
virtual function (虚函数), 538
void, 200
return value (返回值)
 conversion (类型转换), 200
 copy initialized (拷贝初始化), 442
 initialization (初始化), 201
list initialization (列表初始化), 203, 380, 467
local variable, pointer (局部变量, 指针), 201
main, 2–4, 203
pair, 380, 467
reference, local variable (引用, 局部变量), 201
*this, 233, 246
tuple, 639
type checking (类型检查), 200
unique_ptr, 418
reverse, 777
 list and forward_list (list和forward_list), 369
reverse iterator (反向迭代器), 358, 363–365, 372
 ++ (increment) (递增运算符), 363
 -- (decrement) (递减运算符), 363
 base, 364
 compared to iterator (与迭代器对比), 364
reverse_copy, 369, 777
reverse_copy_if, 369
reverse_iterator
 compared to iterator (与iterator对比), 363
 container (容器), 297, 363
rfind, string, 326
right, manipulator (right操纵符), 671
rotate, 777
rotate_copy, 777
rule of three/five (三/五法则), 447, 478
 virtual destructor exception (虚析构函数异常), 552
run-time type identification (运行时类型识别), 730–735,
 763

compared to virtual functions (与虚函数对比), 734
dynamic_cast, 730, **730**
 bad_cast, 731
 to pointer (转换为指针), 730
 to reference (转换为引用), 731
 type-sensitive equality (类型敏感的相等判断), 734
typeid, **732**, 732
 returns **type_info** (返回**type_info**), 732
 runtime binding (运行时绑定), **527**, **576**
runtime_error, 194, 176
 initialization from **string** (从**string**初始化), 175
rvalue, **121**, **150**
 copy initialization, uses move constructor (拷贝初始化, 使用移动构造函数), 477
 result (结果)
 ++ (increment) postfix (后置递增), 131
 -- (decrement) postfix (后置递减), 131
 function nonreference return type (函数非引用返回类型), 201
 rvalue reference (右值引用), **471**, **487**
 cast from **lvalue** (从左值转换), 612
 collapsing rule (折叠规则), 609
 compared to **lvalue reference** (与左值引用对比), 471
 function matching (函数匹配), 477
 initialization (初始化), 470
 member function (成员函数), **483**
 overloaded (重载), 484
 move, 472
 parameter (形参)
 forwarding (转发), 612, 622
 member function (成员函数), 481
 preserves argument type information (保持实参类型信息), 613
 template argument deduction (模板实参推断), 608
 variable (变量), 471

S

Sales_data
 compareIsbn, 345
 += (compound assignment) (复合赋值运算符), 500
 + (addition) (加法运算符), 497
 == (equality) (相等运算符), 497
 != (inequality) (不等运算符), 497
 >> (input operator) (输入运算符), 495
 << (output operator) (输出运算符), 494
 add, 234

addition program (加法程序), 66
avg_price, 232
 bookstore program (书店程序), 229
 using algorithms (使用算法), 360
 class definition (类定义), 64, 240
 combine, 230
compareIsbn, 379
 with associative container (与关联容器一起使用), 379
 constructors (构造函数), 236–239
 converting constructor (转换构造函数), 264
 default constructor (默认构造函数), 235
 exception classes (异常类), 694
 exception version (带异常处理版本)
 += (compound assignment) (复合赋值运算符), 695
 + (addition) (加法运算符), 694
explicit constructor (**explicit**构造函数), 265
 isbn, 231
 operations (操作), 228
 print, 234
 read, 234
 relational operators (关系运算符), 499
Sales_data.h header (**Sales_data.h**头文件), 67
Sales_item, 17
 + (addition) (加法运算符), 18
 >> (input operator) (输入运算符), 18
 << (output operator) (输出运算符), 18
 addition program (加法程序), 19
 bookstore program (书店程序), 21
 isbn, 20
 operations (操作), 17
Sales_item.h header (**Sales_item.h**头文件), 17
 scientific manipulator (**scientific**操纵符), 670
 scope (作用域), **44**, **70**
 base class (基类), 547
 block (块), **44**, **70**, 155
 class (类), 65, **253**, 253–257, 245
 static member (static成员), 270
 compared to object lifetime (与对象生命周期对比), 184
 derived class (派生类), 547
 friend (友元), 242, 252
 function (函数), 184
 global (全局作用域), **44**, **70**
 inheritance (继承), 547–551
 member function (成员函数), 253
 parameters and return type (参数和返回类型), 253
 multiple inheritance (多重继承), 715
 name collisions, using directive (名字冲突,

using 指示), 704
namespace (命名空间), 695–700
nested class (嵌套类), 747
overloaded function (重载函数), 210
statement (语句), 155
template parameter (模板参数), 592
template specialization (模板特例化), 626
using directive (using 指示), 703
virtual function (虚函数), 550
scoped enumeration (限定作用域的枚举类型), 736, 763
enum class, 736
Screen, 243
pos member (pos 成员), 243
concatenating operations (连接操作), 247
do_display, 248
friends (友元), 250
get, 244, 253
get_cursor, 739
Menu function table (Menu 函数表), 743
move, 744
move members (move 成员), 247
set, 247
search, 772
search_n, 771
seed, random-number engine (随机数引擎种子), 663
seekp, seekg, 676–679
self-assignment (自赋值)
copy and swap assignment (拷贝并交换赋值操作), 459
copy assignment (拷贝赋值操作), 454
explicit check (显式检查), 480
HasPtr
reference counted (引用计数), 456
valuelike (类值), 454
Message, 463
move assignment (移动赋值操作), 475
pitfalls (误区), 454
StrVec, 467
semicolon (;) (分号), 3
class definition (类定义), 65
null statement (空语句), 154
separate compilation (分离式编译), 41, 71, 226
compiler options (编译器选项), 186
declaration vs. definition (声明与定义), 41
templates (模板), 581
sequential container (顺序容器), 299, 333
array, 292
deque, 292
forward_list, 292
initialization (初始化), 299–301
list, 292
list initialization (列表初始化), 300
members (成员)
assign, 302
back, 309
clear, 312
emplace, 308
emplace_back, 308
emplace_front, 308
erase, 312
front, 309
insert, 307
pop_back, 311
pop_front, 311
push_back, 118
push_back, 90, 306, 482
push_front, 306
resize, 314
value_type, 298
performance characteristics (性能特点), 292
priority_queue, 330
queue, 330
stack, 329
value initialization (值初始化), 300
vector, 292
set, 374, 398
see also ordered container (参见“有序容器”)
bitset, 644
header (头文件), 374
insert, 383
iterator, 382
key_type requirements (key_type 要求), 378
list initialization (列表初始化), 377
lower_bound, 390
TextQuery class (TextQuery 类), 431
upper_bound, 390
word_count program (word_count 程序), 375
set_difference, 779
set_intersection, 573, 779
set_symmetric_difference, 779
set_union, 779
setfill, manipulator (setfill 操纵符), 671
setprecision, manipulator (setprecision 操纵符), 669
setstate, stream (流 setstate 操作), 281
setw, manipulator (setw 操纵符), 671
shared_ptr, 400, 400–406, 412–417, 436
* (dereference) (解引用运算符), 401
copy and assignment (拷贝和赋值), 402
definition (定义), 400
deleter (删除器), 416, 436
bound at run time (运行时绑定), 599
derived-to-base conversion (派生类到基类的转换), 558

destructor (析构函数), 402
 dynamically allocated array (动态分配数组), 426
 exception safety (异常安全的), 415
 factory program (factory程序), 402
 initialization (初始化), 412
 make_shared, 401
 pitfalls (误区), 416
 reset, 414
 StrBlob, 404
 TextQuery class (TextQuery类), 431
 with new (和new一起使用), 412
 short, 30
 short-circuit evaluation (短路求值), 126, 150
 && (logical AND) (逻辑与运算符), 126
 || (logical OR) (逻辑或运算符), 126
 not in overloaded operator (重载的运算符不支持),
 491
 ShorterString, 508
 () (call operator) (调用运算符), 508
 shorterString, 201
 showbase, manipulator (showbase操纵符), 668
 showpoint, manipulator (showpoint操纵符), 671
 shrink_to_fit
 deque, 318
 string, 318
 vector, 318
 shuffle, 777
 signed, 32, 71
 char, 32
 conversion to unsigned (转换为unsigned值),
 32, 142
 out-of-range value (越界值), 33
 signed type (带符号类型), 31
 single-line (//), comment (单行注释//), 8, 23
 size
 container (容器), 79, 91, 118, 304
 priority_queue, 330
 queue, 330
 returns unsigned (返回unsigned值), 80
 stack, 330
 StrVec, 465
 size_t, 103, 118, 644
 array subscript (数组下标), 103
 size_type, container (容器的size_type), 79, 91,
 118, 297
 SizeComp, 508
 () (call operator) (调用运算符), 508
 sizeof, 139, 150
 array (数组), 139
 data member (数据成员), 139
 sizeof..., parameter pack (参数包的sizeof...), 619
 skipws, manipulator (skipws操纵符), 672
 sliced (切掉), 535, 576
 SmallInt
 + (addition) (加法运算符), 521
 conversion operator, 514
 smart pointer (智能指针), 400, 436
 exception safety (异常安全的), 415
 pitfalls (误区), 416
 smatch, 646, 649, 680, 681
 prefix, 652
 provide context for a match (为匹配提供上下文),
 651
 suffix, 652
 sort, 343, 775
 source file (源文件), 4, 24
 specialization, *see* template specialization (特例化, 参见
 “模板特例化”)
 splice, list, 370
 splice_after, forward_list, 370
 sregex_iterator, 649, 681
 i before e program (查找i在e之前的文本的程序),
 651
 sstream
 file marker (文件标记), 676
 header (头文件), 278, 287
 off_type, 677
 pos_type, 677
 random access (随机访问), 676
 random IO program (随机IO程序), 677
 seek和tell, 676–681
 ssub_match, 649, 652, 681
 example (例子), 655
 stable_partition, 775
 stable_sort, 345, 775
 stack, 330, 333
 emplace, 330
 empty, 330
 equality and relational operators (相等和关系运算
 符), 329
 header (头文件), 330
 initialization (初始化), 329
 pop, 330
 push, 330
 sequential container (顺序容器), 330
 size, 330
 swap, 330
 top, 330
 stack unwinding, exception handling (栈展开, 异常处理),
 684, 723
 standard error (标准错误), 5, 24
 standard header (标准头文件), #include, 5, 18
 standard input (标准输入), 5, 24
 standard library (标准库), 5, 24

- standard output (标准输出), 5, 24
statement (语句), 2, 24
 block, *see* block (语句块, 参见“块”)
 break, 170, 178
 compound (复合语句), 155, 178
 continue, 171, 178
 do while, 169, 178
 expression (表达式), 154, 178
 for, 11, 24, 166, 166–168, 179
 goto, 172, 179
 if, 15, 24, 156, 156–159, 179
 labeled (带标签语句), 172, 179
 null (空语句), 154, 179
 range for (范围for语句), 82, 167, 167–169, 179
 return, 199, 199–204
 scope (作用域), 155
 switch, 159, 159–145, 179
 while, 10, 25, 165, 165–166, 179
statement label (语句标签), 172
static (file static) (文件内静态定义), 701, 723
static member (静态成员)
 Account, 269
 class template (类模板), 591
 accessed through an instantiation (通过实例化访问), 591
 definition (定义), 591
 const data member, initialization (const数据成员, 初始化), 270
 data member (数据成员), 268
 definition (定义), 270
 default argument (默认实参), 271
 definition (定义), 270
 inheritance (继承), 532
 instantiation (实例化), 591
 member function (成员函数), 269
 nested class (嵌套类), 747
 scope (作用域), 270
static object, local (局部静态对象), 185, 226
static type (静态类型), 534, 576
 determines name lookup (确定名字查找), 547, 549
 multiple inheritance (多重继承), 713
static type checking (静态类型检查), 42
static_cast, 145, 145
 lvalue to rvalue (左值到右值的静态转换), 612
std, 7, 25
std::forward, *see* forward (参见forward)
std::move, *see* move (参见move)
stdexcept header (stdexcept头文件), 174, 176
stod, 328
stof, 328
stoi, 328
stol, 328
stold, 328
stoll, 328
store, free (自由空间), 400, 436
stoul, 328
stoull, 328
str, string streams (字符串流), 289
StrBlob, 405
 back, 406
 begin, 422
 check, 406
 constructor (构造函数), 405
 end, 422
 front, 406
 pop_back, 406
 shared_ptr, 404
StrBlobPtr, 421
 ++ (increment) (递增运算符), 502
 -- (decrement) (递减运算符), 502
 * (dereference) (解引用运算符), 504
 -> (arrow operator) (箭头运算符), 504
 check, 421
 constructor (构造函数), 421
 deref, 422
 incr, 422
 weak_ptr, 421
strcat, 110
strcmp, 110
strcpy, 110
stream (流)
 as condition (作为条件), 13, 144, 280
 clear, 281
 explicit conversion to bool (explicit转换为bool值), 516
 file marker (文件标记), 676
 flushing buffer (刷新缓冲区), 282
 format state (格式状态), 666
 istream_iterator, 359
 iterator (迭代器), 357, 359–362, 372
 type requirements (类型要求), 362
 not flushed if program crashes (如果程序崩溃不刷新缓冲区), 282
 ostream_iterator, 359
 random IO (随机IO), 676
 rdstate, 281
 setstate, 281
strict weak ordering (严格弱序), 378, 398
string, 71, 75–84, 118
 see also container (参见“容器”)
 see also sequential container (参见“顺序容器”)
 see also iterator (参见“迭代器”)
 [](subscript) (下标运算符), 84, 118, 310

+= (compound assignment) (复合赋值运算符), 80
 + (addition) (加法运算符), 80
 >> (input operator) (输入运算符), 77, 118
 >> (input operator) as condition (输入运算符作为
 条件), 77
 << (output operator) (输出运算符), 77, 118
 and string literal (和字符串字面值常量), 80–81
 append, 323
 assign, 323
 at, 311
 C-style string (C风格字符串), 111
 c_str, 111
 capacity, 318
 case sensitive (大小写敏感), 325
 compare, 327
 concatenation (连接), 80
 default initialization (默认初始化), 40
difference_type, 100
 equality and relational operators (相等和关系运算
 符), 80
 erase, 323
 find, 325
 find_first_not_of, 325
 find_last_not_of, 326
 find_last_of, 326
 getline, 78, 287
 header (头文件), 66, 67, 75
 initialization (初始化), 75–76, 320–322
 initialization from string literal (从字符串字面值
 常量初始化), 76
 insert, 323
 move constructor (移动构造函数), 468
 numeric conversions (数值类型转换), 327
 random-access iterator (随机访问迭代器), 366
 replace, 323
 reserve, 318
 rfind, 326
 subscript range (下标范围), 85
 substr, 321
TextQuery class (TextQuery类), 431
 string literal (字符串字面值常量), 6, 24, 36
see also C-style string (参见“C风格字符串”)
 and string (和string), 80–81
 concatenation (连接), 36
stringstream, 287, 287–289, 290
 initialization (初始化), 287
 strlen, 109
 struct
see also class (参见“类”)
 default access specifier (默认访问说明符), 240
 default inheritance specifier (默认继承说明符), 546
StrVec, 464
 [](subscript) (下标运算符), 501
 = (assignment) (赋值运算符),
 initializer_list, 499
 alloc_n_copy, 467
 begin, 465
 capacity, 465
 chk_n_alloc, 465
 copy assignment (拷贝赋值运算符), 467
 copy constructor (拷贝构造函数), 467
 default constructor (默认构造函数), 466
 design (设计), 464
 destructor (析构函数), 467
 emplace_back, 622
 end, 465
 free, 467
 memory allocation strategy (内存分配策略), 465
 move assignment (移动赋值运算符), 474
 move constructor (移动构造函数), 473
 push_back, 466
 move-enabled (可移动的), 482
 reallocate, 469
 move iterator version (移动迭代器版本), 480
 size, 466
 subexpression (子表达式), 681
 subscript range (下标范围), 84
 array (数组), 104
 string, 85
 validating (合法性检查), 93
 vector, 94
 substr, string, 321
 suffix, smatch, 652
 sum, program (sum程序), 604
 swap, 457
 array, 303
 container (交换容器), 303
 container nonmember version (交换容器的非成员
 版本), 303
 copy and swap assignment operator (拷贝并交换
 赋值运算符), 459
 priority_queue, 330
 queue, 330
 stack, 330
 typical implementation (典型实现), 457–459
 swap, program (swap程序), 200
 swap_ranges, 774
 switch statement (switch语句), 160, 160–163, 179
 default case label (default case标签),
 162
 break, 160–162, 170
 compared to if (与if对比), 159
 execution flow (执行流), 161
 variable definition (变量定义), 163

`syntax_option_type, regex`, 647
synthesized (合成的)
 copy assignment (拷贝赋值运算符), 443, 487
 copy constructor (拷贝构造函数), 440, 487
 copy control (拷贝控制成员), 239
 as deleted function (删除函数), 450
 as deleted in derived class (派生类中的删除函数), 553
 `Bulk_quote`, 553
 multiple inheritance (多重继承), 712
 virtual base class (虚基类), 721
 virtual base classes (多个虚基类), 721
 `volatile`, 758
 default constructor (默认构造函数), 236, 274
 derived class (派生类), 552
 members of built-in type (内置类型成员), 236
 destructor (析构函数), 446, 487
 move operations (移动操作)
 deleted function (删除函数), 476
 not always defined (不总是定义), 476

T

`\t` (tab character) (制表符), 36
`tellp, tellg`, 676–679
template (模板)
 see also class template (参见“类模板”)
 see also function template (参见“函数模板”)
 see also instantiation (参见“实例化”)
 declaration (声明), 592
 link time errors (链接时错误), 582
 overview (概述), 578
 parameter, *see* template parameter (参数, 参见“模板参数”)
 parameter list (参数列表), 631
 template argument (模板实参), 579, 631
 explicit (显式实参), 584, 630
 template member, *see* member template (模板成员, 参见“成员模板”)
 type alias (类型别名), 590
 type transformation templates (类型转换模板), 605, 631
 type-dependencies (类型依赖), 583
 variadic, *see* variadic template (可变参数, 参见“可变参数模板”)
template argument deduction (模板实参推断), 600, 631
 `compare`, 602
 explicit template argument (显式模板实参), 604
 function pointer (函数指针), 607
 limited conversions (类型转换限制), 601
 low-level `const` (底层 `const`), 613
 rvalue reference parameter (左值引用参数), 608

multiple function parameters (用作多个函数参数类型), 602
parameter with nontemplate type (非模板类型参数), 602
reference parameters (引用参数), 608–610
rvalue reference parameter (右值引用参数), 608
top-level `const` (顶层 `const`), 601
template class, *see* class template (模板类, 参见“类模板”)
template function, *see* function template (模板函数, 参见“函数模板”)
template parameter (模板参数), 578, 631
 default template argument (默认模板实参), 594
 class template (类模板), 594
 function template (函数模板), 594
 name (名字), 592
 restrictions on use (使用上的限制), 592
 nontype parameter (非类型参数), 580, 630
 must be constant expression (必须是常量表达式), 581
 type requirements (类型要求), 580
scope (作用域), 592
template argument deduction (模板实参推断), 602
type parameter (类型参数), 580, 580, 631
 as friend (作为友元), 590
 used in template class (用于模板类中), 584
template parameter pack (模板参数包), 618, 631
 expansion (扩展), 621
 pattern (模式), 621
template specialization (模板特例化), 625, 624–629, 631
 class template (类模板), 626–629
 class template member (类模板成员), 628
 `compare` function template (`compare` 函数模板), 624
 compared to overloading (与重载对比), 625
 declaration dependencies (声明依赖), 626
 function template (函数模板), 625
 `hash<key_type>`, 626, 698
 headers (头文件), 626
 of namespace member (命名空间成员的特例化), 626, 698
partial, class template (类模板部分特例化), 628, 630
scope (作用域), 626
 `template<>`, 625
`template<>`
 default template argument (默认模板实参), 594
 template specialization (模板特例化), 625
temporary (临时量对象), 55, 71
terminate function (`terminate` 函数), 685, 723
 exception handling (异常处理), 175, 178
machine-dependent (机器相关), 175

- terminology (术语)
 - const reference (const引用), 54
 - iterator (迭代器), 97
 - object (对象), 39
 - overloaded new and delete (重载new和delete), 727
 - test, bitset, 644
 - TextQuery, 431
 - class definition (类定义), 432
 - constructor (构造函数), 433
 - main program (main程序), 431
 - program design (程序设计), 430
 - query, 434
 - revisited (再探), 562
 - this pointer (this指针), 231, 274
 - static members (static成员), 269
 - as argument (this指针作为参数), 239
 - in return (this指针在return语句中), 233
 - overloaded (重载)
 - on const (对const重载), 247
 - on lvalue or rvalue reference (对左值或右值引用重载), 483
 - throw, 173, 173, 178, 684, 723
 - execution flow (执行流), 175, 684
 - pointer to local object (局部对象的指针), 686
 - rethrow (重新抛出), 688, 723
 - runtime_error, 174
 - throw(), exception specification (异常说明), 691
 - tie member (tie成员), ostream, 283
 - to_string, 328
 - Token, 751
 - assignment operators (赋值运算符), 751
 - copy control (拷贝控制), 753
 - copyUnion, 753
 - default constructor (默认构造函数), 752
 - discriminant (判别式), 752
 - tolower, 83
 - top
 - priority_queue, 330
 - stack, 330
 - top-level const (顶层const), 57, 71
 - and auto (和auto), 61
 - argument and parameter (实参和形参), 190
 - decltype, 63
 - parameter (形参), 208
 - template argument deduction (模板实参推断), 600
 - toupper, 83
 - ToyAnimal, virtual base class (虚基类), 721
 - trailing return type (尾置返回类型), 206, 226
 - function template (函数模板), 605
 - lambda expression (lambda表达式), 353
 - pointer to array (数组的指针), 206
 - pointer to function (函数的指针), 223
 - transform
 - algorithm (算法), 353, 773
 - program (程序), 393
 - translation unit (编译单元), 3
 - trunc (file mode) (文件模式), 286
 - try block (try块), 172, 174, 178, 684, 724
 - tuple, 637, 680
 - findBook, program (findBook程序), 639
 - equality and relational operators (相等和关系运算符), 638
 - header (头文件), 636
 - initialization (初始化), 637
 - make_tuple, 637
 - return value (返回值), 638
 - value initialization (值初始化), 637
 - type (类型)
 - alias (别名), 60, 71
 - template (模板), 590
 - alias declaration (别名声明), 60
 - arithmetic (算术类型), 30, 69
 - built-in (内置类型), 2, 23, 30–32
 - checking (类型检查), 42, 71
 - argument and parameter (实参和形参), 183
 - array reference parameter (数组引用参数), 195
 - function return value (函数返回类型), 200
 - name lookup (名字查找), 210
 - class (类型), 17, 23
 - compound (复合类型), 45, 45–52, 69
 - conversion, *see* conversion (类型转换, 参见conversion)
 - dynamic (动态类型), 534, 575
 - incomplete (不完全类型), 250, 274
 - integral (整形类型), 30, 70
 - literal (字面值常量), 59
 - class type (类类型), 267
 - specifier (说明符), 37, 71
 - static (静态类型), 534, 576
 - type alias declaration (类型别名声明), 61, 69, 71
 - pointer, to array (数组的指针), 206
 - pointer to function (函数的指针), 222
 - pointer to member (成员的指针), 742
 - template type (模板类型), 590
 - type independence, algorithms (类型无关算法), 337
 - type member, class (类型成员, 类), 243
 - type parameter, *see* template parameter (类型参数, 参见“模板参数”)
 - type transformation templates (类型转换模板), 605, 631
 - type_traits, 606
 - type_info, 763
 - header (头文件), 176
 - name, 735

- no copy or assign (无拷贝或赋值), 735
operations (操作), 735
returned from typeid (从 typeid 返回 type_info), 732
- type_traits**
header (头文件), 605
remove_pointer, 606
remove_reference, 605
and move (和 move), 611
- type transformation templates (类型转换模板), 606
- typedef**, 60, 71
const, 61
and pointer, to const (与 const 的指针), 61
pointer, to array (数组的指针), 206
pointer to function (函数的指针), 222
- typeid operator** (typeid 运算符), 732, 732, 763
returns type_info (返回 type_info), 732
- typeinfo header** (typeinfo 头文件), 731, 732, 735
- typename**
compared to class (与类比较), 580
required for type member (类型成员所需要的), 593
template parameter (模板参数), 579
- ## U
- unary operators (一元运算符), 120, 150
overloaded operator (重载运算符), 490
- unary predicate (一元谓词), 344, 372
- unary_function deprecated (unary_function, 已弃用), 513
- uncaught exception (未捕获的异常), 685
- undefined behavior (未定义行为), 33, 71
base class destructor not virtual (基类析构函数不是虚函数), 552
- bitwise operators and signed values (位运算符和带符号值), 136
- caching end() iterator (缓存 end() 返回的迭代器), 316
- cstring functions (cstring 函数), 109
- dangling pointer (空悬指针), 411
- default initialized members of builtin type (默认初始化内置类型成员), 236
- delete of invalid pointer (delete 无效指针), 409
- destination sequence too small (目的序列太小), 341
- element access empty container (访问空容器中的元素), 309
- invalidated iterator (使迭代器无效), 96, 315
- missing return statement (缺少返回语句), 201
- misuse of smart pointer get (误用智能指针的 get), 414
- omitting [] when deleting array (释放数组时忘记 []), 425
- operand order of evaluation (运算对象求值顺序), 123, 133
- out-of-range subscript (下标越界), 84
- out-of-range value assigned to signed type (赋予带符号类型的值越界), 33
- overflow and underflow (向上溢出和向下溢出), 125
- pointer casts (指针类型转换), 145
- pointer comparisons (指针比较), 109
- return reference or pointer to local variable (返回指向局部变量的引用或指针), 201
- string invalid initializer** (string 的无效初始值), 321
- uninitialized (未初始化的)
dynamic object (动态对象), 407
local variable (局部变量), 184
pointer (指针), 48
variable (变量), 41
- using unconstructed memory (使用未构造的内存), 428
- using unmatched match object (使用未匹配的 match 对象), 653
- writing to a const object (向 const 对象写入值), 145
- wrong deleter with smart pointer (错误的智能指针删除器), 426
- underflow_error**, 176
- unformatted IO (未格式化 IO), 673, 681
istream, 673
multi-byte, istream (多字节 istream), 675
single-byte, istream (单字节 istream), 673
- unget, istream, 673
- uniform_int_distribution, 661
- uniform_real_distribution, 664
- uninitialized (未初始化的), 7, 24, 40, 71
pointer, undefined behavior (未初始化指针, 未定义行为), 48
variable, undefined behavior (未初始化变量, 未定义行为), 41
- uninitialized_copy, 429
move iterator (移动迭代器), 480
- uninitialized_fill, 429
- union, 749, 763
anonymous (匿名), 750, 762
class type member (类类型成员), 750
assignment operators (赋值运算符), 751
copy control (拷贝控制), 753
default constructor (默认构造函数), 752

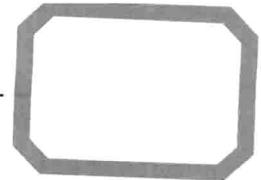
- deleted copy control (删除的拷贝控制成员), 751
- placement new (定位new), 753
- definition (定义), 750
- discriminant (判别式), 752
- restrictions (限制), 749
- unique, 343, 777
 - list and forward_list (list和forward_list), 369
- unique_copy, 359, 777
- unique_ptr, 400, 417–420, 436
 - * (dereference) (解引用运算符), 400
 - copy and assignment (拷贝和赋值), 417
 - definition (定义), 417, 419
 - deleter (删除器), 419, 436
 - bound at compile time (编译时绑定), 600
 - dynamically allocated array (动态分配数组), 425
 - initialization (初始化), 417
 - pitfalls (误区), 417
 - release, 418
 - reset, 418
 - return value (返回值), 418
 - transfer ownership (转移所有权), 418
 - with new (和new一起使用), 417
- unitbuf, manipulator (unitbuf操纵符), 282
- unnamed namespace (未命名的命名空间), 700, 724
 - local to file (文件的局部名字), 700
 - replace file static (代替文件的static声明), 701
- unordered container, 394, 398
 - see also container (参见“容器”)
 - see also associative container (参见“关联容器”)
 - bucket management (桶管理), 395
 - hash<key_type> specialization
 - (hash<key_type>特例化), 626, 698
 - compatible with == (equality) (与相等运算符==兼容), 627
 - key_type requirements (key_type的要求), 396
 - override default hash (覆盖默认哈希函数), 396
- unordered_map, 398
 - see also unordered container (参见“无序容器”)
 - * (dereference) (解引用运算符), 382
 - [](subscript) (下标运算符), 387, 398
 - adds element (添加元素), 387
 - at, 387
 - definition (定义), 376
 - header (头文件), 374
 - list initialization (列表初始化), 377
 - word_count program (word_count程序), 394
- unordered_multimap, 398
 - see also unordered container (参见“无序容器”)
- * (dereference) (解引用运算符), 382
- definition (定义), 376
- has no subscript operator (无下标运算符), 387
- insert, 385
- list initialization (列表初始化), 377
- unordered_multiset, 398
 - see also unordered container (参见“无序容器”)
 - insert, 385
 - iterator, 382
 - list initialization (列表初始化), 377
 - override default hash (覆盖默认哈希函数), 396
- unordered_set, 398
 - see also unordered container (参见“无序容器”)
 - header (头文件), 374
 - iterator, 382
 - list initialization (列表初始化), 377
 - unscoped enumeration (不限定作用域枚举类型), 737, 764
 - as union discriminant (作为union的判别式), 752
 - conversion to integer (转换为整数), 738
 - enum, 736
- unsigned, 31, 71
 - char, 32
 - conversion (类型转换), 33
 - conversion from signed (从signed类型进行转换), 32
 - conversion to signed (转换为signed类型), 142
 - literal (numU or numu) (字面值常量, 数值后跟U或u), 37
 - size return type (size返回类型), 79
- unsigned type (无符号类型), 31
- unwinding, stack (栈展开), 684, 723
- upper_bound
 - algorithm (算法), 772
 - ordered container (有序容器), 390
 - used in Basket (用于Basket), 560
- uppercase, manipulator (uppercase操纵符), 668
- use count, see reference count (使用计数, 参见“引用计数”)
- user-defined conversion, see class type conversion (用户定义的类型转换, 参见“类类型转换”)
- user-defined header (用户自定义头文件), 67–68
 - const和constexpr, 67
 - default argument (默认实参), 213
 - function declaration (函数声明), 186
 - #include, 18
 - inline function (inline函数), 215
 - inline member function definition (inline成员函数定义), 244
 - template definition (模板定义), 582

template specialization (模板特例化), 625
`using =`, see type alias declaration (参见“类型别名声明”)
`using` declaration (`using`声明), 74, 118, 702, 724
 access control (访问控制), 544
 not in header files (不要置于头文件中), 75
 overloaded function (重载函数), 708
 overloaded inherited functions (重载继承的函数), 551
 scope (作用域), 703
`using` directive (`using`指示), 703, 724
 overloaded function (重载函数), 709
pitfalls (误区), 704
scope (作用域), 703, 704
 name collisions (名字冲突), 704
`utility` header (`utility`头文件), 379, 469, 472, 614

V

`valid`, program (`valid`程序), 655
valid but unspecified (有效但未指定), 475
valid pointer (有效指针), 47
value initialization (值初始化), 88, 118
 dynamically allocated, object (动态分配, 对象), 407
 map subscript operator (映射的下标运算符), 387
 `new []`, 424
 `resize`, 314
 sequential container (顺序容器), 300
 `tuple`, 637
 uses default constructor (使用默认构造函数), 262
 `vector`, 88
`value_type`
 associative container (关联容器), 381, 397
 sequential container (顺序容器), 298
valuelike class, copy control (类值类, 拷贝控制), 453
`varargs`, 199
variable (变量), 7, 25, 38, 38–45, 71
 `const`, 53
 `constexpr`, 59
 declaration (声明), 41
 class type (类类型), 263
 define before use (先定义后使用), 42
 defined after label (在标签后定义), 163, 172
 definition (定义), 38, 41
 `extern`, 41
 `extern`和`const`, 54
 initialization (初始化), 39, 40, 70
 is lvalue (左值), 471
 lifetime (生命周期), 184
 local (局部变量), 184, 226
 preprocessor (预处理器), 70

variadic template (可变参数模板), 618, 631
declaration dependencies (声明依赖), 621
forwarding (转发), 622
 usage pattern (使用模式), 624
function matching (函数匹配), 620
pack expansion (包扩展), 621–622
parameter pack (参数包), 618
`print` program (`print`程序), 620
recursive function (递归函数), 620
`sizeof...`, 619
`vector`, 86–94, 118, 333
 see also container (参见“容器”)
 see also sequential container (参见“顺序容器”)
 see also iterator (参见“迭代器”)
 `[](subscript)` (下标运算符), 92, 118, 310
 `=` (assignment) (赋值), list initialization (列表初始化), 129
 `at`, 310
 `capacity`, 318
 capacity program (容量程序), 318
 definition (定义), 87
 `difference_type`, 100
 `erase`, changes container size (改变容器大小), 343
 header (头文件), 86, 294
 initialization (初始化), 87–90, 299–301
 initialization from array (从数组初始化), 111
 list initialization (列表初始化), 88, 300
 memory management (内存管理), 317
 overview (概述), 292
 `push_back`, invalidates iterator (使迭代器失效), 316
 random-access iterator (随机访问迭代器), 366
 `reserve`, 318
 subscript range (下标范围), 94
 `TextQuery` class (`TextQuery`类), 431
 value initialization (值初始化), 88, 300
viable function (可行函数), 217, 226
 see also function matching (参见“函数匹配”)
virtual base class (虚基类), 717, 724
 ambiguities (二义性), 719
 `Bear`, 718
 class derivation list (类派生列表), 718
 conversion (类型转换), 719
 derived class constructor (派生类构造函数), 720
 `iostream`, 717
 name lookup (名字查找), 719
 order of destruction (析构顺序), 721
 order of initialization (初始化顺序), 720
 `ostream`, 717
 `Raccoon`, 718
 `ToyAnimal`, 721



ZooAnimal, 717
 virtual function (虚函数), 526, 528, 535–541, 576
 compared to run-time type identification (与运行时类型识别比较), 734
 default argument (默认实参), 539
 derived class (派生类), 529
 destructor (析构函数), 552
 exception specification (异常说明), 692
 final specifier (final说明符), 538
 in constructor, destructor (在构造函数和析构函数中调用虚函数), 556
 multiple inheritance (多重继承), 715
 overloaded function (重载函数), 551
 override (覆盖), 528, 576
 override specifier (override说明符), 526, 529, 538
 overriding run-time binding (覆盖运行时绑定), 539
 overview (虚函数概述), 528
 pure (纯虚函数), 540
 resolved at run time (运行时解析), 536, 537
 return type (返回类型), 537
 scope (作用域), 550
 type-sensitive equality (类型敏感的相等运算符), 734
 virtual inheritance, *see* virtual base class (虚继承, 参见虚基类)
 Visual Studio (编译器), 4
void, 30, 71
 return type (返回类型), 200
void*, 50, 71
 conversion from pointer (从指针进行类型转换), 143
volatile, 757, 764
 pointer (指针), 757
 synthesized copy-control members (合成的拷贝控制成员), 758
 vowel counting, program (元音计数程序), 160

W
wcerr, 278
wchar_t, 30
 literal (字面值常量), 37
wchar_t streams (**wchar_t**流), 278

weak_ptr, 400, 420–421, 436
 definition (定义), 420
 initialization (初始化), 420
 lock, 420
 StrBlobPtr, 421
wfstream, 278
what, exception, 175, 693
while statement (**while**语句), 10, 25, 165, 165–166, 178
 condition (条件), 10, 165
wide character streams (宽字符流), 278
wifstream, 278
window, console (窗口, 控制台), 5
Window_mgr, 250
wiostream, 278
wistream, 278
wistringstream, 278
wofstream, 278
word (机器字), 31, 71
word_count program (**word_count** 程序)
 map, 375
 set, 375
unordered_map, 394
word_transform program (**word transform** 程序), 392
WordQuery, 564, 568
wostream, 278
wostream, 278
wregex, 649
write, ostream (**写ostream**), 676
wistringstream, 278

X
\xnnn (hexadecimal escape sequence, 十六进制转义字符), 36

Z
ZooAnimal
 program (**ZooAnimal**程序), 710
 virtual base class (虚基类), 717