Defined Terms

术语

argument（实参）

A value passed to a function when it is called.

传递给被调用函数的值。

block（块）

Sequence of statements enclosed in curly braces.

花括号括起来的语句序列。

buffer（缓冲区）

A region of storage used to hold data. IO facilities often store input (or output) in a buffer and read or write the buffer independently of actions in the program. Output buffers usually must be explicitly flushed to force the buffer to be written. By default, reading cin flushes cout; cout is also flushed when the program ends normally.

一段用来存放数据的存储区域。IO 设备常存储输入（或输出）到缓冲区，并独立于程序动作对缓冲区进行读写。输出缓冲区通常必须显式刷新以强制输出缓冲区内容。默认情况下，读 cin 会刷新 cout；当程序正常结束时，cout 也被刷新。

built-in type（内置类型）

A type, such as int, defined by the language.

C++ 语言本身定义的类型，如 int。

cerr

ostream object tied to the standard error, which is often the same stream as the standard output. By default, writes to cerr are not buffered. Usually used for error messages or other output that is not part of the normal logic of the program.

绑定到标准错误的 ostream 对象，这通常是与标准输出相同的流。默认情况下，输出 cerr 不缓冲，通常用于不是程序正常逻辑部分的错误信息或其他输出。

cin

istream object used to read from the standard input.

用于从标准输入中读入的 istream 对象。

class

C++ mechanism for defining our own data structures. The class is one of the most fundamental features in C++. Library types, such as istream and ostream, are classes.

用于自定义数据结构的 C++ 机制。类是 C++ 中最基本的特征。标准库类型，如 istream 和 ostream，都是类。

class type

A type defined by a class. The name of the type is the class name.

由类所定义的类型，类型名就是类名。

clog

ostream object tied to the standard error. By default, writes to clog are buffered. Usually used to report information about program execution to a log file.

绑定到标准错误的 ostream 对象。默认情况下，写到 clog 时是带缓冲的。通常用于将程序执行信息写入到日志文件中。

comments（注释）

Program text that is ignored by the compiler. C++ has two kinds of comments: single-line and paired. Single-line comments start with a //. Everything from the // to the end of the line is a comment. Paired comments begin with a /* and include all text up to the next */.

编译器会忽略的程序文本。C++ 有单行注释和成对注释两种类型的注释。单行注释以 // 开头，从 // 到行的结尾是一条注释。成对注释以 /* 开始包括到下一个 */ 为止的所有文本。

condition（条件）

An expression that is evaluated as true or false. An arithmetic expression that evaluates to zero is false; any other value yields true.

求值为真或假的表达式。值为 0 的算术表达式是假，其他所有非 0 值都是真。

cout

ostream object used to write to the standard output. Ordinarily used to write the output of a program.

用于写入到标准输出的 ostream 对象，一般情况下用于程序的输出。

curly brace（花括号）

Curly braces delimit blocks. An open curly ({) starts a block; a close curly (}) ends one.

花括号对语句块定界。左花括号“{”开始一个块，右花括号“}”结束块。

data structure（数据结构）

A logical grouping of data and operations on that data.

数据及数据上操作的逻辑组合。

edit-compile-debug（编辑—编译—调试）

The process of getting a program to execute properly.

使得程序正确执行的过程。

end-of-file（文件结束符）

System-specific marker in a file that indicates that there is no more input in the file.

文件中与特定系统有关的标记，表示这个文件中不再有其他输入。

expression（表达式）

an

The smallest unit of computation. An expression consists of one or more operands and usually an operator. Expressions are evaluated to produce a result. For example, assuming i and j are ints, then i + j is an arithmetic addition expression d yields the sum of the two int values. Expressions are covered in more detail in Chapter 5.

最小的计算单元。表达式包含一个或多个操作数并经常含有一个操作符。表达式被求值并产生一个结果。例如，假定 i 和 j 都为 int 型，则 i + j 是一个算术加法表达式并求这两个 int 值的和。表达式将在第五章详细介绍。

for statement（for 语句）

Control statement that provides iterative execution. Often used to step through a data structure or to repeat a calculation a fixed number of times.

提供迭代执行的控制语句，通常用于步进遍历数据结构或对一个计算重复固定次数。

function（函数）

A named unit of computation.

有名字的计算单元。

function body（函数体）

Statement block that defines the actions performed by a function.

定义函数所执行的动作的语句块。

function name（函数名）

Name by which a function is known and can be called.

函数的名字标识，函数通过函数名调用。

header（头文件）

A mechanism whereby the definitions of a class or other names may be made available to multiple programs. A header is included in a program through a #include directive.

使得类或其他名字的定义在多个程序中可用的一种机制。程序中通过 #include 指示包含头文件。

if statement（if 语句）

Conditional execution based on the value of a specified condition. If the condition is true, the if body is executed. If not, control flows to the statement following the else if there is one or to the statement following the if if there is no else.

根据指定条件的值执行的语句。如果条件为真，则执行 if 语句体；否则控制流执行 else 后面的语句，如果没有 else 将执行 if 后面的语句。

iostream（输入输出流）

library type providing stream-oriented input and output.

提供面向流的输入和输出的标准库类型。

istream（输入流）

Library type providing stream-oriented input.

提供面向流的输入的标准库类型。

library type（标准库类型）

A type, such as istream, defined by the standard library.

标准库所定义的类型，如 istream。

main function（主函数）

Function called by the operating system when executing a C++ program. Each program must have one and only one function named main.

执行 C++ 程序时，操作系统调用的函数。每一个程序有且仅有一个主函数 main。

manipulator（操纵符）

Object, such as std::endl, that when read or written "manipulates" the stream itself. Section A.3.1 (p. 825) covers manipulators in more detail.

在读或写时“操纵”流本身的对象，如 std::endl。A.3.1 节详细讲述操纵符。

member function（成员函数）

Operation defined by a class. Member functions ordinarily are called to operate on a specific object.

类定义的操作。成员函数通常在特定的对象上进行操作。

method（方法）

Synonym for member function.

成员函数的同义词。

namespace（命名空间）

Mechanism for putting names defined by a library into a single place. Namespaces help avoid inadvertent name clashes. The names defined by the C++ library are in the namespace std.

将库所定义的名字放至单独一个地方的机制。命名空间有助于避免无意的命名冲突。C++ 标准库所定义的名字在命名空间 std 中。

ostream（输出流）

Library type providing stream-oriented output.

提供面向流的输出的库类型。

parameter list（形参表）

Part of the definition of a function. Possibly empty list that specifies what arguments can be used to call the function.

函数定义的组成部分。指明可以用什么参数来调用函数，可能为空。

preprocessor directive（预处理指示）

An instruction to the C++ preprocessor. #include is a preprocessor directive. Preprocessor directives must appear on a single line. We'll learn more about the preprocessor in Section 2.9.2.

C++ 预处理器的指示。#include 是一个预处理器指示。预处理器指示必须出现在单独的行中。第 2.9.2 节将对预处理器作详细的介绍。

return type（返回类型）

Type of the value returned by a function.

函数返回值的类型。

source file（源文件）

Term used to describe a file that contains a C++ program.

用来描述包含在 C++ 程序中的文件的术语。

standard error（标准错误）

An output stream intended for use for error reporting. Ordinarily, on a windowing operating system, the standard output and the standard error are tied to the window in which the program is executed.

用于错误报告的输出流。通常，在视窗操作系统中，将标准输出和标准错误绑定到程序的执行窗口。

standard input（标准输入）

The input stream that ordinarily is associated by the operating system with the window in which the program executes.

和程序执行窗口相关联的输入流，通常这种关联由操作系统设定。

standard library（标准库）

Collection of types and functions that every C++ compiler must support. The library provides a rich set of capabilities including the types that support IO. C++ programmers tend to talk about "the library," meaning the entire standard library or about particular parts of the library by referring to a library type. For example, programmers also refer to the "iostream library," meaning the part of the standard library defined by the iostream classes.

每个 C++ 编译器必须支持的类型和函数的集合。标准库提供了强大的功能，包括支持 IO 的类型。C++ 程序员谈到的“标准库”，是指整个标准库，当提到某个标准库类型时也指标准库中某个特定的部分。例如，程序员提到的“iostream 库”，专指标准库中由 iostream 类定义的那部分。

standard output（标准输出）

The output stream that ordinarily is associated by the operating system with the window in which the program executes.

和程序执行窗口相关联的输出流，通常这种关联由操作系统设定。

statement（语句）

The smallest independent unit in a C++ program. It is analogous to a sentence in a natural language. Statements in C++ generally end in semicolons.

C++ 程序中最小的独立单元，类似于自然语言中的句子。C++ 中的语句一般以分号结束。

std

Name of the namespace used by the standard library. std::cout indicates that we're using the name cout defined in the std namespace.

标准库命名空间的名字，std::cout 表明正在使用定义在 std 命名空间中的名字 cout。

string literal（字符串字面值）

Sequence of characters enclosed in double quotes.

以双引号括起来的字符序列。

uninitialized variable（未初始化变量）

Variable that has no initial value specified. There are no uninitialized variables of class type. Variables of class type for which no initial value is specified are initialized as specified by the class definition. You must give a value to an uninitialized variable before attempting to use the variable's value. Uninitialized variables can be a rich source of bugs.

没有指定初始值的变量。类类型没有未初始化变量。没有指定初始值的类类型变量由类定义初始化。在使用变量值之前必须给未初始化的变量赋值。未初始化变量是造成 bug 的主要原因之一。

variable（变量）

A named object.

有名字的对象。

while statement（while 语句）

An iterative control statement that executes the statement that is the while body as long as a specified condition is true. The body is executed zero or more times, depending on the truth value of the condition.

一种迭代控制语句，只要指定的条件为真就执行 while 循环体。while 循环体执行 0 次还是多次，依赖于条件的真值。

() operator[()操作符]

The call operator: A pair of parentheses "()" following a function name. The operator causes a function to be invoked. Arguments to the function may be passed inside the parentheses.

调用操作符。跟在函数名后且成对出现的圆括号。该操作符导致函数被调用，给函数的实参可在括号里传递。

++ operator（++操作符）

Increment operator. Adds one to the operand; ++i is equivalent to i = i+ 1.

自增操作符。将操作数加 1，++i 等价于 i = i + 1。

+= operator（+= 操作符）

A compound assignment operator. Adds right-hand operand to the left and stores the result back into the left-hand operand; a += b is equivalent to a =a + b.

复合赋值操作符，将右操作数和左操作数相加，并将结果存储到左操作数中；a += b 等价于 a = a + b。

. operator（. 操作符）

Dot operator. Takes two operands: the left-hand operand is an object and the right is the name of a member of that object. The operator fetches that member from the named object.

点操作符。接受两个操作数：左操作数是一个对象，而右边是该对象的一个成员的名字。这个操作符从指定对象中取得成员。

:: operator (:: 操作符)

Scope operator. We'll see more about scope in Chapter 2. Among other uses, the scope operator is used to access names in a namespace. For example, std::cout says to use the name cout from the namespace std.

作用域操作符。在第二章中，我们将看到更多关于作用域的介绍。在其他的使用过程中，:: 操作符用于在命名空间中访问名字。例如，std::cout 表示使用命名空间 std 中的名字 cout。

= operator (= 操作符)

Assigns the value of the right-hand operand to the object denoted by the left-hand operand.

表示把右操作数的值赋给左操作数表示的对象。

<< operator (<< 操作符)

Output operator. Writes the right-hand operand to the output stream indicated by the left-hand operand: cout << "hi" writes hi to the standard output. Output operations can be chained together: cout << "hi << "bye" writes hibye.

输出操作符。把右操作数写到左操作数指定的输出流：cout << "hi" 把 hi 写入到标准输出流。输出操作可以链接在一起使用：cout << "hi << "bye" 输出 hibye。

>> operator (>> 操作符)

Input operator. Reads from the input stream specified by the left-hand operand into the right-hand operand: cin >> i reads the next value on the standard input into i. Input operations can be chained together: cin >> i >> j reads first into i and then into j.

输入操作符。从左操作数指定的输入流读入数据到右操作数：cin >> i 把标准输入流中的下一个值读入到 i 中。输入操作能够链接在一起使用：cin >> i >> j 先读入 i 然后再读入 j。

== operator (== 操作符)

The equality operator. Tests whether the left-hand operand is equal to the right-hand.

等于操作符，测试左右两边的操作数是否相等。

!= operator（!=操作符）

Assignment operator. Tests whether the left-hand operand is not equal to the right-hand.

不等于操作符。测试左右两边的操作数是否不等。

<= operator（<= 操作符）

The less-than-or-equal operator. Tests whether the left-hand operand is less than or equal to the right-hand.

小于或等于操作符。测试左操作数是否小于或等于右操作数。

< operator（< 操作符）

The less-than operator. Tests whether the left-hand operand is less than the right-hand.

小于操作符。测试左操作数是否小于右操作数。

>= operator（>= 操作符）

Greater-than-or-equal operator. Tests whether the left-hand operand is greater than or equal to the right-hand.

大于或等于操作符。测试左操作数是否大于或等于右操作数。

> operator（> 操作符）

Greater-than operator. Tests whether the left-hand operand is greater than the right-hand.

大于操作符。测试左操作数是否大于右操作数。

Defined Terms

术语

C-style strings（C 风格字符串）

C programs treat pointers to null-terminated character arrays as strings. In C++, string literals are C-style strings. The C library defines a set of functions that operate on such strings, which C++ makes available in the cstring header. C++ programs should use C++ library strings in preference to C-style strings, which are inherently error-prone. A sizeable majority of security holes in networked programs are due to bugs related to using C-style strings and arrays.

C 程序把指向以空字符结束的字符数组的指针视为字符串。在 C++ 中，字符串字面值就是 C 风格字符串。C 标准库定义了一系列处理这种字符串的库函数，C++ 中将这些标准库函数放在 cstring 头文件中。由于 C 风格字符串本质上容易出错，C++ 程序应该优先使用 C++ 标准库类 string 而少用 C 风格字符串。网络程序中大量的安全漏洞都源于与使用 C 风格字符串和数组相关的缺陷。

compiler extension（编译器扩展）

Feature that is added to the language by a particular compiler. Programs that rely on compiler extensions cannot be moved easily to other compilers.

特定编译器为语言添加的特性。依赖于编译器扩展的程序很难移植到其他的编译器。

compound type（复合类型）

Type that is defined in terms of another type. Arrays, pointers, and references are compound types.

使用其他类型定义的类型。数组、指针和引用都是复合类型。

const void*

A pointer type that can point to any const type. See void*.

可以指向任意 const 类型的指针类型，参见 void *。

delete expression（delete 表达式）

A delete expression frees memory that was allocated by new:

delete 表达式用于释放由 new 动态分配的内存：

```
delete [] p;
```

where p must be a pointer to the first element in a dynamically allocated array. The bracket pair is essential: It indicates to the compiler that the pointer points at an array, not at a single object. In C++ programs, delete replaces the use of the C library free function.

在此表达式中，p 必须是指向动态创建的数组中第一个元素的指针，其中方括号必不可少：它告诉编译器该指针指向数组，而非单个对象。C++ 程序使用 delete 取代 C 语言的标准库函数 free。

dimension（维数）

The size of an array.

数组大小。

dynamically allocated（动态分配的）

An object that is allocated on the program's free store. Objects allocated on the free store exist until they are explicitly deleted.

在程序自由存储区中建立的对象。该对象一经创建就一直存在，直到显式释放为止。

free store（自由存储区）

Memory pool available to a program to hold dynamically allocated objects.

程序用来存储动态创建对象的内存区域。

heap（堆）

Synonym for free store.

自由存储区的同义词。

new expression（new 表达式）

Allocates dynamic memory. We allocate an array of n elements as follows:

用于分配动态内存的表达式。下面的语句分配了一个有 n 个元素的数组：

```
new type[n];
```

The array holds elements of the indicated type. new returns a pointer to the first element in the array. C++ programs use new in place of the C library malloc function.

该数组存放 type 类型的元素。new 返回指向该数组第一个元素的指针。C++ 程序使用 new 取代 C 语言的标准库函数 malloc。

pointer（指针）

An object that holds the address of an object.

存放对象地址的对象。

pointer arithmetic（指针算术操作）

The arithmetic operations that can be applied to pointers. An integral type can be added to or subtracted from a pointer, resulting in a pointer positioned that many elements ahead or behind the original pointer. Two pointers can be subtracted, yielding the difference between the pointers. Pointer arithmetic is valid only on pointers that denote elements in the same array or an element one past the end of that array.

可用于指针的算术操作。允许在指针上做加上或减去整型值的操作，以获得当前指针之前或之后若干个元素处的地址。两个指针可做减法操作，得到它们之间的差值。只有当指针指向同一个数组或其超出末端的位置时，指针的算术操作才有意义。

precedence（优先级）

Defines the order in which operands are grouped with operators in a compound expression.

在复杂的表达式中，优先级确定了操作数分组的次序。

ptrdiff_t

Machine-dependent signed integral type defined in cstddef header that is large enough to hold the difference between two pointers into the largest possible array.

在 cstddef 头文件中定义的与机器相关的有符号整型，该类型具有足够的大小存储两个指针的差值，这两个指针指向同一个可能的最大数组。

size_t

Machine-dependent unsigned integral type defined in cstddef header that is large enough to hold the size of the largest possible array.

在 cstddef 头文件中定义的与机器相关的无符号整型，它具有足够的大小存储一个可能的最大数组。

* operator（* 操作符）

Dereferencing a pointer yields the object to which the pointer points. The dereference operator returns an lvalue; we may assign to the value returned from the dereference operator, which has the effect of assigning a new value to the underlying element.

对指针进行解引用操作获得该指针所指向的对象。解引用操作符返回左值，因此可为其结果赋值，等效于为该指针所指向的特定对象赋新值。

++ operator（++ 操作符）

When used with a pointer, the increment operator "adds one" by moving the pointer to refer to the next element in an array.

用于指针时，自增操作符给指针“加 1”，移动指针使其指向数组的下一个元素。

[] operator（[] 操作符）

The subscript operator takes two operands: a pointer to an element of an array and an index. Its result is the element that is offset from the pointer by the index. Indices count from zerothe first element in an array is element 0, and the last is element size of the array minus 1. The subscript operator returns an lvalue; we may use a subscript as the left-hand operand of an assignment, which has the effect of assigning a new value to the indexed element.

下标操作符接受两个操作数：一个是指向数组元素的指针，一个是下标 n。该操作返回偏离指针当前指向 n 个位置的元素值。数组下标从 0 开始计数——数组第一个元素的下标为 0，最后一个元素的下标是数组长度减

1。下标操作返回左值，可用做赋值操作的左操作数，等效于为该下标引用的元素赋新值。

& operator（& 操作符）

The address-of operator. Takes a single argument that must be an lvalue. Yields the address in memory of that object.

取地址操作符需要一个操作数，其唯一的操作数必须是左值对象，该操作返回操作数对象在内存中的存储地址。

void*

A pointer type that can point to any nonconst type. Only limited operations are permitted on void* pointers. They can be passed or returned from functions and they can be compared with other pointers. They may not be dereferenced.

可以指向任何非 const 对象的指针类型。void* 指针只提供有限的几种操作：可用作函数形参类型或返回类型，也可与其他指针做比较操作，但是不能进行解引用操作。

Defined Terms

术语

ambiguous call（有二义性的调用）

Compile-time error that results when there is not a single best match for a call to an overloaded function.

一种编译错误，当调用重载函数，找不到唯一的最佳匹配时产生。

arguments（实参）

Values supplied when calling a function. These values are used to initialize the corresponding parameters in the same way that variables of the same type are initialized.

调用函数时提供的值。这些值用于初始化相应的形参，其方式类似于初始化同类型变量的方法。

automatic objects（自动对象）

Objects that are local to a function. Automatic objects are created and initialized anew on each call and are destroyed at the end of the block in which they are defined. They no longer exist once the function terminates.

局部于函数的对象。自动对象会在每一次函数调用时重新创建和初始化，并在定义它的函数块结束时撤销。一旦函数执行完毕，这些对象就不再存在了。

best match（最佳匹配）

The single function from a set of overloaded functions that has the best match for the arguments of a given call.

在重载函数集合里找到的与给定调用的实参达到最佳匹配的唯一函数。

call operator（调用操作符）

The operator that causes a function to be executed. The operator is a pair of parentheses and takes two operands: The name of the function to call and a (possibly empty) comma-separated list of arguments to pass to the function.

使函数执行的操作符。该操作符是一对圆括号，并且有两个操作数：被调用函数的名字，以及由逗号分隔的（也可能为空）形参表。

candidate functions（候选函数）

The set of functions that are considered when resolving a function call. The candidate functions are all the functions with the name used in the call for which a declaration is in scope at the time of the call.

在解析函数调用时考虑的函数集合。候选函数包括了所有在该调用发生的作用域中声明的、具有该调用所使用的名字的函数。

const member function（常量成员函数）

Function that is member of a class and that may be called for const objects of that type. const member functions may not change the data members of the object on which they operate.

类的成员函数，并可以由该类类型的常量对象调用。常量成员函数不能修改所操纵的对象的数据成员。

constructor（构造函数）

Member function that has the same name as its class. A constructor says how to initialize objects of its class. Constructors have no return type. Constructors may be overloaded.

与所属类同名的类成员函数。构造函数说明如何初始化本类的对象。构造函数没有返回类型，而且可以重载。

constructor initializer list（构造函数初始化列表）

List used in a constructor to specify initial values for data members. The initializer list appears in the definition of a constructor between the parameter list and the constructor body. The list consists of a colon followed by a comma-separated list of member names, each of which is followed by that member's initial value in parentheses.

在构造函数中用于为数据成员指定初值的表。初始化列表出现在构造函数的定义中，位于构造函数体与形参表之间。该表由冒号和冒号后面的一组用逗号分隔的成员名组成，每一个成员名后面跟着用圆括号括起来的该成员的初值。

default constructor（默认构造函数）

The constructor that is used when no explicit initializer is supplied. The compiler will synthesize a default constructor if the class defines no other constructors.

在没有显式提供初始化式时调用的构造函数。如果类中没有定义任何构造函数，编译器会自动为这个类合成默认构造函数。

function（函数）

A callable unit of computation.

可调用的计算单元。

function body（函数体）

Block that defines the actions of a function.

定义函数动作的语句块。

function matching（函数匹配）

Compiler process by which a call to an overloaded function is resolved. Arguments used in the call are compared to the parameter list of each overloaded function.

确定重载函数调用的编译器过程。调用时使用的实参将与每个重载函数的形参表作比较。

function prototype（函数原型）

Synonym for function declaration. The name, return type, and parameter types of a function. To call a function, its prototype must have been declared before the point of call.

函数声明的同义词。包括了函数的名字、返回类型和形参类型。调用函数时，必须在调用点之前声明函数原型。

inline function（内联函数）

Function that is expanded at the point of call, if possible. Inline functions avoid the normal function-calling overhead by replacing the call by the function's code.

如果可能的话，将在调用点展开的函数。内联函数直接以函数代码替代了函数调用点展开的函数。内联函数直接以函数代码替代了函数调用语句，从而避免了一般函数调用的开销。

local static objects（局部静态对象）

Local object that is created and initialized once before the function is first called and whose value persists across invocations of the function.

在函数第一次调用前就已经创建和初始化的局部对象，其值在函数的调用之间保持有效。

local variables（局部变量）

Variables defined inside a function. Local variables are accessible only within the function body.

在函数内定义的变量，仅能在函数体内访问。

object lifetime（对象生命期）

Every object has an associated lifetime. Objects that are defined inside a block exist from when their definition is encountered until the end of the block in which they are defined. Local static objects and global objects defined outside any function are created during program startup and are destroyed when the main function ends. Dynamically created objects that are created through a new expression exist until the memory in which they were created is freed through a corresponding delete.

每个对象皆有与之关联的生命期。在块中定义的对象从定义时开始存在，直到它的定义所在的语句块结束为止。静态局部对象和函数外定义的全局变量则在程序开始执行时创建，当 main 函数结束时撤销。动态创建的对象由 new 表达式创建，从此开始存在，直到由相应的 delete 表达式释放所占据的内存空间为止。

overload resolution（重载确定）

A synonym for function matching.

函数匹配的同义词。

overloaded function（重载函数）

A function that has the same name as at least one other function. Overloaded functions must differ in the number or type of their parameters.

和至少一个其他函数同名的函数。重载函数必须在形参的个数或类型上有所不同。

parameters（形参）

Variables local to a function whose initial values are supplied when the function is called.

函数的局部变量，其初值由函数调用提供。

recursive function（递归函数）

Function that calls itself directly or indirectly.

直接或间接调用自己的函数。

return type（返回类型）

The type of the value returned from a function.

函数返回值的类型。

synthesized default constructor（合成默认构造函数）

If there are no constructors defined by a class, then the compiler will create (synthesize) a default constructor. This constructor default initializes each data member of the class.

如果类没有定义任何构造函数，则编译器会为这个类创建（合成）一个默认构造函数。该函数以默认的方式初始化类中的所有数据成员。

temporary object（临时对象）

Unnamed object automatically created by the compiler in the course of evaluating an expression. The phrase temporary object is usually abreviated as temporary. A temporary persists until the end of the largest expression that encloses the expression for which it was created.

在求解表达式的过程中由编译器自动创建的没有名字的对象。“临时对象”这个术语通常简称为“临时”。临时对象一直存在直到最大表达式结束为止，最大表达式指的是包含创建该临时对象的表达式的最大范围内的表达式。

this pointer（this 指针）

Implicit parameter of a member function. this points to the object on which the function is invoked. It is a pointer to the class type. In a const member function the pointer is a pointer to const.

成员函数的隐式形参。this 指针指向调用该函数的对象，是指向类类型的指针。在 const 成员函数中，该指针也指向 const 对象。

viable functions（可行函数）

The subset of overloaded functions that could match a given call. Viable functions have the same number of parameters as arguments to the call and each argument type can potentially be converted to the corresponding parameter type.

重载函数中可与指定的函数调用匹配的子集。可行函数的形参个数必须与该函数调用的实参个数相同，而且每个实参类型都可潜在地转换为相应形参的类型。

Defined Terms

术语

base class（基类）

A class that is the parent of another class. The base class defines the interface that a derived class inherits.

是其他类的父类。基类定义了派生类所继承的接口。

condition state（条件状态）

Flags and associated functions usable by any of the stream classes that indicate whether a given stream is usable. States and functions to get and set these states are listed in Table 8.2 (p. 288).

流类用于指示给定的流是否用的标志以及相关函数。表 8.2 列出了流的状态以及获取和设置这些状态的函数。

derived class（派生类）

A derived class is one that shares an interface with its parent class.

与父类共享接口的类。

file mode（文件模式）

Flags defined by the fstream classes that are specified when opening a file and control how a file can be used. Listed in Table 8.3 (p. 297).

由 fstream 类定义的标志，在打开文件和控制文件如何使用时指定。表 8.3 列出了所有的文件模式。

fstream

Stream object that reads or writes a named file. In addition to the normal iostream operations, the fstream class also defines open and close members. The open member function takes a C-style character string that names the file to open and an optional open mode argument. By default ifstreams are opened with in mode, ofstreams with out mode, and fstreams with in and out mode set. The close member closes

the file to which the stream is attached. It must be called before another file can be opened.

用来读或写已命名文件的流对象。除了普通的 iostream 操作外，fstream 类还定义了 open 和 close 成员。open 成员函数有一个表示打开文件名的 C 风格字符串参数和一个可选的打开模式参数。默认时，ifstream 对象以 in 模式打开，ofstream 对象以 out 模式打开，而 fstream 对象则同时以 in 和 out 模式打开。close 成员关闭流关联的文件，必须在打开另一个文件前调用。

inheritance（继承）

Types that are related by inheritance share a common interface. A derived class inherits properties from its base class. Chapter 15 covers inheritance.

有继承关系的类型共享相同的接口。派生类继承其基类的属性。第十五章将继承。

object-oriented library（面向对象标准库）

A set of classes related by inheritance. Generally speaking, the base class of an object-oriented library defines an interface that is shared by the classes derived from that base class. In the IO library, the istream and ostream classes serve as base classes for the types defined in the fstream and sstream headers. We can use an object of a derived class as if it were an object of the base class. For example, we can use the operations defined for istream on an ifstream object.

有继承关系的类的集合。一般来说，面向对象标准库的基类定义了接口，由继承这个 基类的各个派生类共享。在 IO 标准库中，istream 和 ostream 类是 fstream 和 sstream 头文件中定义的类型的基类。派生类的对象可当做基类对象使用。例如，可在 ifstream 对象上使用 istream 定义的操作。

stringstream

Stream object that reads or writes a string. In addition to the normal iostream operations, it also defines an overloaded member named str. Calling str with no arguments returns the string to which the stringstream is attached. Calling it with a string attaches the stringstream to a copy of that string.

读写字符串的流对象。除了普通的 iostream 操作外，它还定义了名为 str 的重载成员。无实参地调用 str 将返回 stringstream 所关联的 string 值。用 string 对象做实参调用它则将该 stringstream 对象与实参副本相关联。

Defined Terms

术语

abstract data type（抽象数据类型）

A data structure that uses encapsulation to hide its implementation, allowing programmers using the type to think abstractly about what the type does rather than concretely about how the type is represented. Classes in C++ can be used to define abstract data types.

使用封装来隐藏其实现的数据结构，允许使用类型的程序员抽象地考虑该类型做什么，而不是具体地考虑类型如何表示。C++ 中的类可用来定义抽象数据类型。

access label（访问标号）

A public or private label that defines whether the following members are accessible to users of the class or only to the friends and members of the class. Each label sets the access protection for the members declared up to the next label. Labels may appear multiple times within the class.

public 或 private 标号，指定后面的成员可以被类的使用者访问或者只能被类的友元和成员访问。每个标号为在该标号到下一个标号之间声明的成员设置访问保护。标号可以在类中出现多次。

class（类）

C++ mechanism for defining our own abstract data types. Classes may have data, function or type members. A class defines a new type and a new scope.

是 C++ 中定义抽象数据类型的一种机制，可以有数据、函数或类型成员。一个类定义了新的类型和新的作用域。

class declaration（类声明）

A class may be declared before it is defined. A class declaration is the keyword class (or struct) followed by the class name followed by a semicolon. A class that is declared but not defined is an incomplete type.

类可以在定义之前声明。类声明用关键字 class（或 struct）表示，后面加类名字和一个分号。已声明但没有定义的类是一个不完全的类型。

class keyword（class 关键字）

In a class defined following the class keyword, the initial implicit access label is private.

用在 class 关键字定义的类中，初始的隐式访问标号是 private。

class scope（类作用域）

Each class defines a scope. Class scopes are more complicated than other scopesmember functions defined within the class body may use names that appear after the definition.

每个类定义一个作用域。类作用域比其他作用域复杂得多——在类的定义体内定义的成员函数可以使用出现在该定义之后的名字。

concrete class（具体类）

A class that exposes its implementation.

暴露其实现细节的类。

const member function（常量成员函数）

A member function that may not change an object's ordinary (i.e., neither static nor mutable) data members. The this pointer in a const member is a pointer to const. A member function may be overloaded based on whether the function is const.

一种成员函数，不能改变对象的普通（即，既不是 static 也不是 mutable）数据成员。const 成员中的 this 指针指向 const 对象。成员函数是否可以被重载取决于该函数是否为 const。

constructor initializer list（构造函数初始化列表）

Specifies initial values of the data members of a class. The members are initialized to the values specified in the initializer list before the body of the constructor executes. Class members that are not initialized in the initializer list are implicitly initialized by using their default constructor.

指定类的数据成员的初始值。在构造函数体现执行前，用初始化列表中指定的值初始化成员。没有在初始化列表中初始化的类成员，使用它们的默认构造函数隐式初始化。

conversion constructor（转换构造函数）

A nonexplicit constructor that can be called with a single argument. A conversion constructor is used implicitly to convert from the argument's type to the class type.

可用单个实参调用的非 explicit 构造函数。隐式使用转换构造函数将实参的类型转换为类类型。

data abstraction（数据抽象）

Programming technique that focuses on the interface to a type. Data abstraction allows programmers to ignore the details of how a type is represented and to think instead about the operations that the type can perform. Data abstraction is fundamental to both object-oriented and generic programming.

注重类型接口的编程技术。数据抽象允许程序员忽略类型如何表示的细节，而只考虑该类型可以执行的操作。数据抽象是面向对象编程和泛型编程的基础。

default constructor（默认构造函数）

The constructor that is used when no initializer is specified.

没有指定初始化时使用的构造函数。

encapsulation（封装）

Separation of implementation from interface; encapsulation hides the implementation details of a type. In C++, encapsulation is enforced by preventing general user access to the private parts of a class.

实现与接口的分离。封闭隐藏了类型的实现细节。在 C++ 中，实施封装可以阻止普通用户访问类的 private 部分。

explicit constructor（显式构造函数）

Constructor that can be called with a single argument but that may not be used to perform an implicit conversion. A constructor is made explicit by prepending the keyword explicit to its declaration.

可以用单个实参调用但不能用于执行隐式转换的构造函数。通过将关键字 explicit 放在构造函数的声明之前而将其设置为 explicit。

forward declaration（前向声明）

Declaration of an as yet undefined name. Most often used to refer to the declaration of a class that appears prior to the definition of that class. See incomplete type.

对尚未定义的名字的声明。大多用于引用出现在类定义之前的类声明。参见不完全类型。

friend（友元）

Mechanism by which a class grants access to its nonpublic members. Both classes and functions may be named as friends. friends have the same access rights as members.

类授权访问其非 public 成员的机制。类和函数都可以被指定为友元。友元拥有与成员一样的访问权。

incomplete type（不完全类型）

A type that has been declared but not yet defined. It is not possible use an incomplete type to define a variable or class member. It is legal to define references or pointers to incomplete types.

已声明但未定义的类型。不能使用不完全类型来定义变量或类成员。定义指向不完全类型的引用或指针是合法的。

member function（成员函数）

Class member that is a function. Ordinary member functions are bound to an object of the class type through the implicit this pointer. Static member functions are not bound to an object and have no this pointer. Member functions may be overloaded, provided that the versions of the function are distinguished by number or type of their parameters.

类的函数成员。普通成员函数通过隐式的 this 指针绑定到类类型的对象。static 成员函数不与对象绑定且没有 this 指针。成员函数可以被重载，只要该函数的版本可由形参的数目或类型来区别。

mutable data member（可变数据成员）

Data member that is never const, even when it is a member of a const object. A mutable member can be changed inside a const function.

一种永远也不能为 const 对象的数据成员，即使作为 const 对象的成员，也不能为 const 对象。mutable 成员可以在 const 函数中改变。

name lookup（名字查找）

The process by which the use of a name is matched to its corresponding declaration.

将名字的使用与其相应的声明相匹配的过程。

private members（私有成员）

Members defined after a private access label; accessible only to the friends and other class members. Data members and utility functions used by the class that are not part of the type's interface are usually declared private.

在 private 访问标号之后定义的成员，只能被友元和其他的类成员访问。类所使用的数据成员和实用函数在不作为类型接口的组成部分时，通常声明为 private。

public members（公用成员）

Members defined after a public access label; public members are accessible to any user of the class. Ordinarily, only the functions that define the interface to the class should be defined in the public sections.

在 public 访问标号之后定义的成员，可被类的任意使用者访问。一般而言，只有定义类接口的函数应定义在 public 部分。

static member（静态成员）

Data or function member that is not a part of any object but is shared by all objects of a given class.

不是任意对象的组成部分、但由给定类的全体对象所共享的数据或函数成员。

struct keyword（struct 关键字）

In a class defined following the struct keyword, the initial implicit access label is public.

用在 struct 关键字定义的类中，初始的隐式访问标号为 public。

synthesized default constructor（合成的默认构造函数）

The default constructor created (synthesized) by the compiler for classes that do not define any constructors. This constructor initializes members of class type by running that class's default constructor; members of built-in type are uninitialized.

编译器为没有定义任何构造函数的类创建（合成）的构造函数。这个构造函数通过运行该类的默认构造函数来初始化类类型的成员，内置类型的成员不进行初始化。

Defined Terms

术语

assignment operator（赋值操作符）

The assignment operator can be overloaded to define what it means to assign one object of a class type to another of the same type. The assignment operator must be a member of its class and should return a reference to its object. The compiler synthesizes the assignment operator if the class does not explicitly define one.

赋值操作符可以重载，对将某个类类型对象赋值给另一同类型对象的含义进行定义。赋值操作符必须是类的成员并且必须返回对所属类对象的引用。如果类没有定义赋值操作符，则编译器将会合成一个。

copy constructor（复制构造函数）

Constructor that initializes a new object as a copy of another object of the same type. The copy constructor is applied implicitly to pass objects to or from a function by value. If we do not define the copy constructor, the compiler synthesizes one for us.

将新对象初始化为另一同类型对象的副本的构造函数。显式应用复制构造函数进行对象的按值传递，向函数传递对象或从函数返回对象。如果没有定义复制构造函数，编译器就为我们合成一个。

copy control（复制控制）

Special members that control what happens when object of class type are copied, assigned, and destroyed. The compiler synthesizes appropriate definitions for these operations if the class does not otherwise define them.

控制类类型对象的复制、赋值和撤销的特殊成员。如果类没有另外定义它们，则编译器将为这些操作合成适当定义。

destructor（析构函数）

Special member function that cleans up an object when the object goes out of scope or is deleted. The compiler automatically destroys each member. Members of class type are destroyed by invoking their destructor; no explicit work is done to destroy members of built-in or compound type. In particular, the object pointed to by a pointer member is not deleted by the automatic work done by the destructor.

当对象超出作用域或删除对象时，清除对象的特殊成员函数。编译器将自动撤销每个成员。类类型的成员通过调用它们的析构函数而撤销，撤销内置或复合类型的成员无须做显式工作。特别地，析构函数的自动工作不会删除指针成员所指向的对象。

memberwise assignment（逐个成员复制）

Term used to describe how the synthesized assignment operator works. The assignment operator assigns, member by member, from the old object to the new. Members of built-in or compound type are assigned directly. Those that are of class type are assigned by using the member's assignment operator.

用于描述合成赋值操作符如何工作的术语。赋值操作符将成员依次从旧对象赋值给新对象。内置或复合类型成员直接赋值，类类型成员使用成员的赋值操作符进行赋值。

memberwise initialization（逐个成员初始化）

Term used to described how the synthesized copy constructor works. The constructor copies, member by member, from the old object to the new. Members of built-in or compound type are copied directly.

Those that are of class type are copied by using the member's copy constructor.

用于描述合成复制构造函数如何工作的术语。构造函数将成员依次从旧对象复制到新对象。内置或复合类型成员直接复制，类类型成员使用成员的复制构造函数进行复制。

overloaded operator（重载操作符）

Function that redefines one of the C++ operators to operate on object(s) of class type. This chapter showed how to define the assignment operator; Chapter 14 covers overloaded operators in more detail.

将一个 C++ 操作符重定义以操作类类型对象的函数。本章介绍了如何定义赋值操作符，第十四章将更详细地介绍重载操作符。

reference count（引用计数）

Synonym for use count.

使用计数的同义词。

Rule of Three（三法则）

Shorthand for the rule of thumb that if a class needs a nontrivial destructor then it almost surely also needs to define its own copy constructor and an assignment operator.

一个经验原则的简写形式，即，如果一个类需要析构函数，则该类几乎也必然需要定义自己的复制构造函数和赋值操作符。

smart pointer（智能指针）

A class that behaves like a pointer but provides other functionality as well. One common form of smart pointer takes a pointer to a dynamically allocated object and assumes responsibility for deleting that object. The user allocates the object, but the smart pointer class deletes it. Smart pointer classes require that the class implement the copy-control members to manage a pointer to the shared object. That object is deleted only when the last smart pointer pointing to it is destroyed. Use counting is the most popular way to implement smart pointer classes.

一个行为类似指针但也提供其他功能的类。智能指针的一个通用形式接受指向动态分配对象的指针并负责删除该对象。用户分配对象，但由智能指

针类删除它。智能指针类需要实现复制控制成员来管理指向共享对象的指针。只有在撤销了指向共享对象的最后一个智能指针后，才能删除该共享对象。使用计数是实现智能指针类最常用的方式。

synthesized assignment operator（合成赋值操作符）

A version of the assignment operator created (synthesized) by the compiler for classes that do not explicitly define one. The synthesized assignment operator memberwise assigns the right-hand operand to the left.

由编译器为没有显式定义赋值操作符的类创建(合成)的赋值操作符版本。合成赋值操作符将右操作数逐个成员地赋值给左操作数。

synthesized copy constructor（合成复制构造函数）

The copy constructor created (synthesized) by the compiler for classes that do not explicitly define the copy constructor. The synthesized copy constructor memberwise initializes the new object from the existing one.

由编译器为没有显式定义复制构造函数的类创建(合成)的复制构造函数。合成复制构造函数将原对象逐个成员地初始化新对象。

use count（使用计数）

Programming technique used in copy-control members. A use count is stored along with a shared object. A separate class is created that points to the shared object and manages the use count. The constructors, other than the copy constructor, set the state of the shared object and set the use count to one. Each time a new copy is madeeither in the copy constructor or the assignment operatorthe use count is incremented. When an object is destroyed either in the destructor or as the left-hand side of the assignment operatorthe use count is decremented. The assignment operator and the destructor check whether the decremented use count has gone to zero and, if so, they destroy the object.

复制控制成员中使用的编程技术。使用计数与共享对象一起存储。需要创建一个单独类指向共享对象并管理使用计数。由构造函数，而不是复制构造函数，设置共享对象的状态并将使用计数置为 1。每当由复制构造函数或赋值操作符生成一个新副本时，使用计数加 1。由析构函数撤销对象或作为赋值操作符的左操作数撤销对象时，使用计数减 1。赋值操作符和析构函数检查使用计数是否已减至 0。如果是，则撤销对象。

value semantics（值语义）

Description of the copy-control behavior of classes that mimic the way arithmetic types are copied. Copies of valuelike objects are independent: Changes made to a copy have no effect on the original object. A valuelike class that has a pointer member must define its own copy-control members. The copy-control operations copy the object to which the pointer points. Valuelike classes that contain only other valuelike classes or built-in types often can rely on the synthesized copy-control members.

对类模拟算术类型复制方式的复制控制控制行为的描述。值型副本是独立的：对副本的改变不会影响原有对象。具有指针成员的值型类必须定义自己的复制控制成员。复制控制操作复制指针所指向的对象。只包含其他值型类或内置类型的值型类，通常可以依赖合成的复制控制成员。

Defined Terms

术语

binary function object（二元函数对象）

A class that has a function-call operator and represents one of the binary operators, such as one of the arithmetic or relational operators.

具有函数调用操作符且表示一个二元操作符（例如一个算术操作符或关系操作符）的类。

binder（绑定器）

An adaptor that binds an operand of a specified function object. For example, bind2nd(minus<int>(), 2) generates a unary function object that subtracts two from its operand.

绑定指定函数对象的一个操作数的适配器。例如，bind2nd(minus<int>(), 2) 产生一个一元函数对象，从操作灵长中减去 2。

class-type conversion（类类型转换）

Conversions to or from class types. Non-explicit constructors that take a single parameter define a conversion from the parameter type

to the class type. Conversion operators define conversions from the class type to the type specified by the operator.

到类类型或从类类型的转换。接受一个形参的非显式构造函数定义从形参类型到类类型的转换。转换操作符定义从类类型到操作符所指定类型的转换。

conversion operators（转换操作符）

Conversion operators are member functions that define conversions from the class type to another type. Conversion operators must be a member of their class. They do not specify a return type and take no parameters. They return a value of the type of the conversion operator. That is, operator int returns an int, operator Sales_item returns a Sales_item, and so on.

转换操作符是定义从类类型到另一类型的转换的成员函数。转换操作符必须是类的成员，而且不能指定返回类型不能接受形参。转换操作符返回转换操作符类型的值，即，operator int 返回 int，operator Sales_item 返回 Sales_item，依此类推。

function adaptor（函数适配器）

Library type that provides a new interface for a function object.

为函数对象提供新接口的标准库的类型。

function object（函数对象）

Object of a class that defines an overloaded call operator. Function objects can be used where functions are normally expected.

定义了重载调用操作符的类的对象。函数对象可以用在需要函数的地方。

negator（求反器）

An adaptor that negates the value returned by the specified function object. For example, not2(equal_to<int>()) generates a function object that is equivalent to not_equal_to<int>.

将指定函数对象的返回值求反的适配器。例如，not2(equal_to<int>()) 产生与 not_equal_to<int> 等价的函数对象。

smart pointer（智能指针）

A class that defines pointer-like behavior and other functionality, such as reference counting, memory management, or more thorough checking. Such classes typically define overloaded versions of dereference (operator*) and member access (operator->).

一个类，定义了指针式行为和其他功能，如，引用计数、内存管理、更全面的检查等。这种类通常定义了解引用操作符（operator*）和成员访问操作符（operator->）的重载版本。

unary function object（一元函数对象）

A class that has a function-call operator and represents one of the unary operators, unary minus or logical NOT.

具有函数调用操作符且表示一个一元操作符的类，如一元减或逻辑非。

Defined Terms

术语

abstract base class（抽象基类）

Class that has or inherits one or more pure virtual functions. It is not possible to create objects of an abstract base-class type. Abstract base classes exist to define an interface. Derived classes will complete the type by defining type-specific implementations for the pure virtuals defined in the base.

具有或继承了一个或多个纯虚函数的类。不能创建抽象基类类型的对象。抽象基类的存在定义了一个接口，派生类将为基类中定义的纯虚函数定义特定类型的实现，以完成类型。

base class（基类）

Class from which another class inherits. The members of the base class become members of the derived class.

从中派生其他类的类。基类的成员成为派生类的成员。

class derivation list（类派生列表）

Used by a class definition to indicate that the class is a derived class. A derivation list includes an optional access level and names the base class. If no access label is specified, the type of

inheritance depends on the keyword used to define the derived class. By default, if the derived class is defined with the struct keyword, then the base class is inherited publicly. If the class is defined using the class keyword, then the base class is inherited privately.

类定义用它指出类是派生类。派生列表包含可选的访问级别和基类的名字，如果不指定访问标号，继承的类型取决于用来定义派生类的保留字，默认情况下，如果派生类用保留字 struct 定义，则仅有继承基类，如果派生类用保留字 class 定义，则私有继承基类。

derived class（派生类）

A class that inherits from another class. The members of the base class are also members of the derived class. A derived class can redefine the members of its base and can define new members. A derived-class scope is nested in the scope of its base class(es), so the derived class can access members of the base class directly. Members defined in the derived with the same name as members in the base hide those base members; in particular, member functions in the derived do not overload members from the base. A hidden member in the base can be accessed using the scope operator.

继承了其他类的类。基类的成员也是派生类的成员，派生类可以重定义其基类的成员还可以定义新成员。派生类作用域嵌套的基类作用域中，因此派生类可以直接访问基类的成员。派生类中定义的与基类成员同名的成员屏蔽基类成员，具体而言，派生类中的成员函数不重载基类成员。基类中的被屏蔽成员可以用作用域操作符访问。

direct base class（直接基类）

Synonym for immediate base class.

immediate base class 的同义词。

dynamic binding（动态绑定）

Delaying until run time the selection of which function to run. In C++, dynamic binding refers to the run-time choice of which virtual function to run based on the underlying type of the object to which a reference or pointer is bound.

延迟到运行时才选择运行哪个函数。在 C++ 中，动态绑定指的是在运行时基于引用或指针绑定的对象的基础类型而选择运行哪个 virtual 函数。

dynamic type（动态类型）

T

ype at run time. Pointers and references to base-class types can be bound to objects of derived type. In such cases the static type is reference (or pointer) to base, but the dynamic type is reference (or pointer) to derived.

运行时类型。基类类型的指针和引用可以绑定到派生类型的对象，在这种情况下，静态类型是基类引用（或指针），但动态类型是派生类引用（或指针）。

handle class（句柄类）

Class that provides an interface to another class. Commonly used to allocate and manage a pointer to an object of an inheritance hierarchy.

提供到其他类的接口的类。一般用于分配和管理继承层次对象的指针。

immediate base class（直接基类）

A base class from which a derived class inherits directly. The immediate base is the class named in the derivation list. The immediate base may itself be a derived class.

被派生类直接继承的基类。直接基类是在派生列表中指定的基类，直接基类本身可以是派生类。

indirect base class（间接基类）

A base class that is not immediate. A class from which the immediate base class inherits, directly or indirectly, is an indirect base class to the derived class.

非直接基类。直接基类直接或间接继承的类是派生类的间接基类。

inheritance hierarchy（继承层次）

Term used to describe the relationships among classes related by inheritance that share a common base class.

描述因共享公共基类的继承而相关联的类之间关系的术语。

object-oriented programming（面向对象编程）

Term used to describe programs that use data abstraction, inheritance, and dynamic binding.

描述使用数据抽象、继承和动态绑定的程序的术语。

polymorphism（多态性）

A term derived from a Greek word that means "many forms." In object-oriented programming, polymorphism refers to the ability to obtain type-specific behavior based on the dynamic type of a reference or pointer.

从意为“多种形态”的希腊单词派生的术语。在面向对象编程中，多态性指的是基于引用或指针的动态类型获得类型明确的行为的能力。

private inheritance（私有继承）

A form of implementation inheritance in which the public and protected members of a private base class are private in the derived.

实现继承的一种形式，在这种形式中，private 基类的 public 和 protected 成员在派生类中均为 private。

protected access label（受保护访问标号）

Members defined after a protected label may be accessed by class members and friends and by the members (but not friends) of a derived class. protected members are not accessible to ordinary users of the class.

定义在 protected 标号之后的成员可以被类成员、友元和派生类成员（非友元）访问。类的普通用户不能访问 protected 成员。

protected inheritance（受保护继承）

In protected inheritance the protected and public members of the base class are protected in the derived class.

在受保护继承中，基类的 protected 和 public 成员在派生类中均为 protected。

public inheritance（公有继承）

The public interface of the base class is part of the public interface of the derived class.

基类的 public 接口是派生类的 public 接口的一部分。

pure virtual（纯虚函数）

A virtual function declared in the class header using =0 at the end of the function's parameter list. A pure virtual is one that need not be (but may be) defined by the class. A class with a pure virtual is an abstract class. If a derived class does not define its own version of an inherited pure virtual, then the derived class is abstract as well.

类的头文件中在函数形参表末尾 =0 声明的虚函数。类不必（但可以）定义纯虚函数。带纯虚函数的类为抽象类。如果派生类没有定义所继承的纯虚函数的自身版本，则派生类也是抽象类。

refactoring（重构）

Redesigning programs to collect related parts into a single abstraction, replacing the original code by uses of the new abstraction. In OO programs, refactoring frequently happens when redesigning the classes in an inheritance hierarchy. Refactoring often occurs in response to a change in requirements. In general, classes are refactored to move data or function members to the highest common point in the hierarchy to avoid code duplication.

重新设计程序将相关部分集合到一个抽象中，用新抽象的使用代替原来的代码。在面向对象编程中，重新设计继承层次中的类时经常发生重构。响应需求的改变通常需要进行重构。一般而言，对类进行重构时，需要将数据或函数成员移到继承层次的最高公共点以避免代码重复。

sliced（切掉的）

Term used to describe what happens when an object of derived type is used to initialize or assign an object of the base type. The derived portion of the object is "sliced down," leaving only the base portion, which is assigned to the base.

描述用派生类型对象对基类类型对象进行初始化或赋值时情况的术语。对象的派生部分被“切掉”，只留下基类部分，赋给基类对象。

static type（静态类型）

Compile-time type. Static type of an object is the same as its dynamic type. The dynamic type of an object to which a reference

or pointer refers may differ from the static type of the reference or pointer.

编译时类型。对象的静态类型与动态类型相同。引用或指针所引用的对象的动态类型可以不同于引用或指针的静态类型。

virtual function（虚函数）

A member function that defines type-specific behavior. Calls to a virtual made through a reference or pointer are resolved at run time, based on the type of the object to which the reference or pointer is bound.

定义类型明确的行为的成员函数。通过引用或指针进行的虚函数调用，在运行时基于引用或指针定制对象而确定。

Defined Terms

class template（类模板）

A class definition that can be used to define a set of type-specific classes. Class templates are defined using the template keyword followed by a comma-separated list of one or more parameters enclosed in < and > brackets.

可以用来定义一组特定类型的类的类定义。类模板用 template 关键字后接用尖括号（<>）括住、以逗号分隔的一个或多个形参的列表来定义。

export keyword（导出关键字）

Keyword used to indicate that the compiler must remember the location of the associated template definition. Used by compilers that support the separate-compilation model of template instantiation. The export keyword ordinarily goes with a function definition; a class is normally declared as exported in the associated class implementation file. A template may be defined with the export keyword only once in a program.

用来指出编译器必须记住相关模板定义位置的关键字，支持模板实例化的分别编译模型的编译器使用它。export 关键字一般与函数定义一起出现，类通常在相关类实现文件中声明为 export。在一个程序中，一个模板只能用 export 关键字定义一次。

function template（函数模板）

A definition for a function that can be used for a variety of types. A function template is defined using the template keyword followed by a comma-separated list of template one or more parameters enclosed in < and > brackets.

可用于不同类型的函数的定义。函数模板用 template 关键字后接用尖括号（<>）括住、以逗号分隔的一个或多个形参的列表来定义。

generic handle class（泛型句柄类）

A class that holds and manages a pointer to another class. A generic handle takes a single type parameter and allocates and manages a pointer to an object of that type. The handle class defines the necessary copy control members. It also defines the dereference (*) and arrow (->) member access operators to provide access to the underlying object. A generic handle requires no knowledge of the type it manages.

保存和管理指向其他类的指针的类。泛型句柄接受单个类型形参，并且分配和管理指向该类型对象的指针。句柄类定义了必要的复制控制成员，它还定义了解引用操作符（*）和箭头成员访问操作符（->）。泛型句不需要了解它管理的类型。

inclusion compilation model（包含编译模型）

Mechanism used by compilers to find template definitions that relies on template definitions being included in each file that uses the template. Typically, template definitions are stored in a header, and that header must be included in any file that uses the template.

编译器用来寻找模板定义的机制，它依赖于模板定义被包含在每个使用模板的文件中。一般而言，模板定义存储在一个头文件中，使用模板的任意文件必须包含该文件。

instantiation（实例化）

Compiler process whereby the actual template argument(s) are used to generate a specific instance of the template in which the parameter(s) are replaced by the corresponding argument(s). Functions are instantiated automatically based on the arguments used in a call. Template arguments must be provided explicitly whenever a class template is used.

用实际模板实参产生模板特定实例的编译器过程，在该实例中，用对应实参代替形参。函数基于调用中使用的实参自动实例化，使用类模板时必须显式提供模板实参。

member template（成员模板）

A member of a class or class template that is a function template. A member template may not be virtual.

类或类模板的是函数模板的成员。成员模板不能为虚。

nontype parameter（非类型形参）

A template parameter that represents a value. When a function template is instantiated, each nontype parameter is bound to a constant expression as indicated by the arguments used in the call. When a class template is instantiated, each nontype parameter is bound to a constant expression as indicated by the arguments used in the class instantiation.

表示值的模板形参。在实例化函数模板的时候，将每个非类型形参绑定到一个常量表达式，该常量表达式作为调用中使用的实参给出。在实例化类模板的时候，将每个非类型形参绑定到一个常量表达式，该常量表达式作为类实例化中使用的实参给出。

partial specialization（部分特化）

A version of a class template in which some some but not all of the template parameters are specified.

类模板的一个版本，其中指定了某些但非全部的模板形参。

separate compilation model（分别编译模型）

Mechanism used by compilers to find template definitions that allows template definitions and declarations to be stored in independent files. Template declarations are placed in a header, and the definition appears only once in the program, typically in a source file. The compiler implements whatever programming environment support is necessary to find that source file and instantiate the versions of the template used by the program.

编译器用来查找模板定义的机制，它允许将模板定义和声明存储在独立的文件中。模板声明放在一个头文件中，而定义只在程序中出现一次，一般

在源文件中，无论什么编程环境所支持的编译器都必须找到该源文件，并实例化程序所使用的模板版本。

template argument（模板实参）

Type or value specified when using a template type, such as when defining an object or naming a type in a cast.

在使用模板类型（如定义对象或强制类型转换中指定类型）的时候，指定的类型或值。

template argument deduction（模板实参推断）

Process by which the compiler determines which function template to instantiate. The compiler examines the types of the arguments that were specified using a template parameter. It automatically instantiates a version of the function with those types or values bound to the template parameters.

编译器用来确定实例化哪个函数模板的过程。编译器检查用模板形参指定的实参的类型，它用绑定到模板形参的类型或值自动实例化函数的一个版本。

template parameter（模板形参）

A name specifed in the template parameter list that may be used inside the definition of a template. Template parameters can be type or non-type parameters. To use a class template, actual arguments must be specified for each template parameter. The compiler uses those types or values to instantiate a version of the class in which uses of the parameter(s) are replaced by the actual argument(s). When a function template is used, the compiler deduces the template arguments from the arguments in the call and instantiates a specific function using the deduced template arguments.

在模板形参表中指定的、可以在模板定义内部使用的名字。模板形参可以是类型形参或非类型形参。要使用模板，必须为每个模板形参指定实参，编译器使用这些类型或值来实例化类的一个版本，其中形参的使用以实参代替。当使用函数模板的时候，编译器从函数调用中的实参推断模板实参，并使用推断得到的模板实参实例化特定函数。

template parameter list（模板形参表）

List of type or nontype parameters (separated by commas) to be used in the definition or declaration of a template.

在模板定义或声明中使用的类型形参或非类型形参(以逗号分隔)的列表。

template specialization（模板特化）

Redefinition of a class template or a member of a class template in which the template parameters are specified. A template specialization may not appear until after the class that it specializes has been defined. A template specialization must appear before any use of the template for the specialized arguments is used.

类模板或类模板的成员的重定义，其中指定了模板形参。在定义了被特化的模板之前，不能出现该模板的特化。在使用任意实参特化的模板之前，必须先出现模板特化。

type parameter（类型形参）

Name used in a template parameter list to represent a type. When the template is instantiated, each type parameter is bound to an actual type. In a function template, the types are inferred from the argument types or are explicitly specified in the call. Type arguments must be specified for a class template when the class is used.

模板形参表中使用的表示类型的名字。当实例化模板的时候，将每个类型形参绑定到实际类型。在函数模板中，从实参类型中推断类型或者在调用中显式指定类型。类模板的类型实参必须在使用类的时候指定。