Supplement I.A: Glossary

For Introduction to C++ Programming
By Y. Daniel Liang

Chapter 1

- **assembly language** A low-level programming language in which a mnemonic is used to represent each of the machine language instructions.
- **Bcc32** Borland C++ compiler.
- **binary numbers** Numbers consists of 0's and 1's.
- **bit** A binary number 0 or 1.
- **block** A sequence of statements enclosed in braces ({}).
- **byte** A unit of storage. Each byte consists of 8 bits. The size of hard disk and memory is measured in bytes. A megabyte is roughly a million bytes.
- **C++Builder** A C++ IDE by Borland.
- **cable modem** Uses the TV cable line maintained by the cable company. A cable modem is as fast as a DSL.
- **central processing unit** (**CPU**) A small silicon semiconductor chip with millions of transistors that executes instructions.
- **comment** Comments document what a program is and how it is constructed. They are not programming statements and are ignored by the compiler. In Java, comments are preceded by two slashes (//) in a line or enclosed between /* and */ in multiple lines.
- **compiler** A software program that translates source code (e.g., Java source code) into a machine language program.
- **Dev-C++** A freeware C++ IDE.
- **dot pitch** The amount of space between pixels. The smaller the dot pitch, the better the display.
- **DSL** (**digital subscriber line**) Uses a phone line and can transfer data in a speed 20 times faster than a regular modem.
- **g++** GNU C++ compiler.
- **hardware** The physical aspect of the computer that can be seen.
- **hexadecimal numbers** Numbers with radix 16.
- **high-level programming language** Are English-like and easy to learn and program.
- **Integrated Development Environment** (**IDE**) Software that helps programmers write code efficiently. IDE tools integrate editing, compiling, building, debugging, and

online help in one graphical user interface.

- **keyword** (or **reserved word**) A word defined as part of Java language, which have a specific meaning to the compiler and cannot be used for other purposes in the program.
- **machine language** Is a set of primitive instructions built into every computer. The instructions are in the form of binary code, so you have to enter binary codes for various instructions.
- **main class** A class that contains a main function.
- **memory** Stores data and program instructions for CPU to execute.
- **modem** A regular modem uses a phone line and can transfer data in a speed up to 56,000 bps (bits per second).
- **network interface card** (**NIC**) A device to connect a computer to a local area network (LAN). The LAN is commonly used in business, universities, and government organizations. A typical type of NIC, called 10BaseT, can transfer data at 10 Mbps.
- **operating system** (**OS**) A program that manages and controls a computer's activities (e.g., Windows, Linux, Solaris).
- **pixel** Tiny dots that form an image on the screen.
- **resolution** Specifies the number of pixels per square inch. The higher the resolution, the sharper and clearer the image is.
- **software** The invisible instructions that control the hardware and make it work.
- **source code** A program written in a programming language such as Java.
- **source file** A file that stores the source code.
- **storage devices** The permanent storage for data and programs. Memory is volatile, because information is lost when the power is off. Program and data are stored on secondary storage and moved to memory when the computer actually uses them.
- **statement** A unit of code that represents an action or a sequence of actions.
- **stream insertion operator** The << operator used by the cout operator for sending output.
- **Visual C++.NET** A C++ IDE by Microsoft.

Chapter 2

- **algorithm** Statements that describe how a problem is solved in terms of the actions to be executed, and specifies the order in which these actions should be executed. Algorithms can help the programmer plan a program before writing it in a programming language.
- **assignment operator** (=) Assigns a value to a variable.
- **assignment statement** A simple statement that assigns a value to a variable using an assignment operator (=). When a value is assigned to a variable, it replaces the previous value of the variable, which is destroyed.
- **backslash** (\) -- A character that precedes another character to denote the following character has a special meaning. For example, '\t' denote a tab character. The backslash character is also used to denote a Unicode character like '\u00FF'.
- **C-Style cast** (int)x is called the C-style cast.
- **casting** The process of converting a primitive data type value into another primitive type.
- **char type** -- A primitive data type that represents a character.
- **const** The keyword for declaring a constant.
- **constant** A variable declared final in Java. A local constant is a constant declared inside a function.
- **data type** Used to define variables to indicate what kind of value the variable can hold.
- **debugger** A program that facilitates debugging. It enables the program to be executed one statement at a time and enables the contents of the variable to be examined during execution.
- **debugging** The process of finding and fixing errors in a program.
- **declaration** Defines variables, functions, and classes in a program.
- **decrement operator** (**--**) Subtracts one from a numeric variable or a char variable.
- **double type** A primitive data type to represent double precision floating-point numbers with 14 to 15 significant digits of accuracy.
- **encoding** Representing a character using a binary code.
- **float type** A primitive data type to represent single precision floating-point numbers with 6 to 7 significant digits of accuracy. The double type is used to represent double precisions with 14 to 15 significant digits of accuracy.

- **floating-point number** A number that includes a fractional part.
- **expression** Represents a computation involving values, variables, and operators, which evaluates to a value.
- **expression statement** If an expression is used as a statement, it is called an expression statement.
- **identifier** A name of a variable, function, class, interface, or package.
- **increment operator** (**++**) Adds one to a numeric variable or a char variable.
- **incremental development and testing** A programming functionology that develop and test program incrementally. This approach is efficient and productive. It help eliminate and isolate errors.
- **indentation** The use of tabs and spaces to indent the source code to make it easy to read and understand.
- **int type** A primitive data type to represent an integer in the range from -2^{31} (-2147483648) to $2^{31}-1$ (2147483647).
- **literal** A constant value that appears directly in the program. A literal may be numeric, character, boolean, or null for object type.
- **local variable** A variable defined inside a function. An initial value must be assigned to a local variable before it is referenced.
- **logic error** An error that causes the program to produce incorrect result.
- **long type** A primitive data type to represent an integer in the range from -2^{63} to $2^{63}-1$.
- **narrowing** (**of types**) Casting a variable of a type with a larger range to a variable of a type with a smaller range.
- **operator** Operations for primitive data type values. Examples of operators are +, -, *, /, and %.
- **primitive data type** The primitive data types are byte, short, int, long, float, double, boolean, and char.
- **runtime error** An error that causes the program to terminate abnormally.
- **short type** A primitive data type that represents an integer in the range from -2^{15} (-32768) to $2^{15}-1$ (32767).
- **syntax error** An error in the program that violates syntax rules of the language.
- **Unix epoch** January 1, 1970 GMT is known as the *Unix epoch* because 1970 was the year when the Unix operating system was formally introduced.

- **variable** Variables are used to store data and computational results in the program.
- **widening** (**of types**) Casting a variable of a type with a smaller range to a variable of a type with a larger range.
- **whitespace** Characters ' ', '\t', '\f', '\r', and '\n' are whitespaces characters.

Chapter 3

- **boolean expression** An expression that evaluates to a Boolean value.
- **boolean value** true or false.
- **boolean type** A primitive data type for Boolean values (true or false).
- **break statement** Break out of the switch statement.
- **conditional operator** The symbols ? and : appear together in a conditional expression: booleanExpression ? expression1 : expression2;
- **fall-through behavior** In a switch statement, if Once a case is matched, the statements starting from the matched case are executed until a break statement or the end of the switch statement is reached. This phenomenon is referred to as the fall-through behavior.
- **operator associativity** Defines the order in which operators will be evaluated in an expression if the operators has the same precedence order.
- **operator precedence** Defines the order in which operators will be evaluated in an expression.
- **selection statement** A statement that uses if or switch statement to control the execution of the program.
- **short-circuit evaluation** Evaluation that stops when the result of the expression has been determined, even if not all the operands are evaluated. The evaluation involving && or || are examples of short-circuit evaluation.

Chapter 4

- **break statement** Break out of the current loop.
- **continue statement** Break out of the current iteration.
- **infinite loop** A loop that runs indefinitely due to a bug in the loop.
- **iteration** One time execution of the loop body.
- **loop** A structure that control repeated executions of a block of statements.
- **loop-continuation-condition** A Boolean expression that controls the execution of the body. After each iteration, the loop-continuation-condition is reevaluated. If the condition is true, the execution of the loop body is repeated. If the condition is false, the loop terminates.
- **loop body** The part of the loop that contains the statements to be repeated.
- **nested loop** Consists of an outer loop and one or more inner loops. Each time the outer loop is repeated, the inner loops are reentered, and all required iterations are performed.
- **off-by-one error** A common in the loop because the loop is executed one more or one less time than it should have been.
- **sentinel value** A special input value that signifies the end of the input.

Chapter 5

- **actual parameter** (i.e., argument) The variables or data to substitute formal parameters when invoking a function.
- **argument** Same as actual parameter
- **ambiguous invocation** There are two or more possible functions to match an invocation of a function, neither is more specific than the other(s). Therefore, the invocation is ambiguous.
- **divide and conquer** The concept of function abstraction can be applied to the process of developing programs. When writing a large program, you can use the "divide and conquer" strategy to decompose it into subproblems. The subproblems can be further decomposed into smaller, more manageable problems.
- **formal parameter (i.e., parameter)** The variables defined in the function signature.
- **information hiding** A software engineering concept for hiding the detail implementation of a function for the client.
- **function** A collection of statements grouped together to perform an operation. See class function; instance function.
- **function abstraction** A technique in software development that hides detailed implementation. Function abstraction is defined as separating the use of a function from its implementation. The client can use a function without knowing how it is implemented. If you decide to change the implementation, the client program will not be affected.
- **function overloading** Function overloading means that you can define functions with the same name in a class as long as there is enough difference in their parameter profiles.
- **function prototype** Complete function declaration with body.
- **function signature** The combination of the name of a function and the list of its parameters.
- **global variable** A variable declared outside all functions and are accessible to all functions in its scope.
- **header file** A file with extension .h which can be included for use by other files.
- **local variable** A variable declared inside a function.
- **pass-by-reference** (i.e., call-by-reference) A term

used when the address of the actual parameter is passed to the function.

- **pass-by-value** (i.e., call-by-value) A term used when a copy of the value of the argument is passed to the function. For a parameter of a primitive type, the actual value is passed; for a parameter of a reference type, the reference for the object is passed.
- **parameter** Variables defined in the function signature.
- **return type** The type for the return value of a function.
- **return value** A value returned from a function using the return statement.
- **scope of variable** The portion of the program where the variable can be accessed.
- **static local variable** A variable declared in a function as static, whose value is retained for use in the next function call.
- **stepwise refinement** When writing a large program, you can use the "divide and conquer" strategy, also known as stepwise refinement, to decompose it into subproblems. The subproblems can be further decomposed into smaller, more manageable problems.
- **stub** A simple, but not a complete version of the function. The use of stubs enables you to test invoking the function from a caller.

Chapter 6

- **array** A data structure for storing a collection of data of the same type.
- **array initializer** A short hand notation to create and initialize an array.
- **binary search** An efficient function to search a key in an array. Binary search first compares the key with the element in the middle of the array and reduces the search range by half. For binary search to work, the array must be pre-sorted.
- **const array** An array whose contents cannot be changed.
- **index** An integer value used to specify the position of an element in the array. The array index is an int value starting with 0 for the first element, 1 for the second, and so on.
- **indexed variable** arrayRefVar[index] is referred to as an indexed variable that access an element in the array through an index.
- **insertion sort** An approach to sort array. Suppose that you want to sort a list in ascending order. The insertion-sort algorithm sorts a list of values by repeatedly inserting a new element into a sorted sublist until the whole list is sorted.
- **linear search** A function to search an element in an array. Linear search compares the key with the element in the array sequentially.
- **multidimensional array** An array with more than one dimension.
- **selection sort** An approach to sort array. It finds the largest number in the list and places it last. It then finds the largest number remaining and places it next to last, and so on until the list contains only a single number.

Chapter 7

- **address operator** The C++ operator & for obtaining the address of a variable.
- **C-string** A character array with the last character '\0'.
- **delete operator** delete a dynamically created variable using the delete operator.
- **dereference operator** The C++ operator *pointer for accessing the value in a variable through a pointer.
- **heap** Memory space for storing dynamically created variable.
- **indirection operator** Same as dereference operator.
- **memory leak** If a dynamically created variable is not referenced, its content cannot be accessed. It is useless, but cannot be discarded. This is called memory leak.
- **data field encapsulation** To prevent direct modifications of properties through the object reference, you can declare the field private, using the private modifier. Data field encapsulation makes the class easy to maintain.
- **default constructor** If a class does not define any constructors explicitly, a no-arg constructor with empty body is assumed. This is called a default constructor.
- **new operator** The C++ operator for creating a variable dynamically.
- **null terminator** The last character in a C-string.
- **pointer-based string** Same as C-string.

Chapter 8

- **base case** A simple case where recursion stops.
- **infinite recursion** Recursion never stops.
- **recursive function** A function that invokes itself directly or indirectly.
- **recursive helper function** Sometimes the original function needs to be modified to receive additional parameters in order to be invoked recursively. A recursive helper function can be declared for this purpose.
- **stopping condition** Same as base case.

Chapter 9

- **accessor function** (**getter**) The function for retrieving a private field in an object.
- **arrow operator** (**->**) The operator for accessing the object through a pointer.
- **class** An encapsulated collection of data and functions that operate on data. A class may be instantiated to create an object that is an instance of the class.
- **class abstraction** A technique in software development that hides detailed implementation. Class abstraction hides the implementation of the class from the client, if you decide to change the implementation, the client program will not be affected.
- **class encapsulation** Combining of functions and data into a single data structure.
- **class's contract** Refers to the collection of functions and fields that are accessible from outside a class, together with the description of how these members are expected to behave.
- **constructor** A special function for initializing objects when creating objects using the new operator. The constructor has exactly the same name as its defining class. Constructors can be overloaded, making it easier to construct objects with different initial data values.
- **data field encapsulation** To prevent direct modifications of properties through the object reference, you can declare the field private, using the private modifier. Data field encapsulation makes the class easy to maintain.
- **default constructor** If a class does not define any constructors explicitly, a no-arg constructor with empty body is assumed. This is called a default constructor.
- **dot operator** (**.**) An operator used to access members of an object. If the member is static, it can be accessed through the class name using the dot operator.
- **instance** An object of a class.
- **instance function** A nonstatic function in a class. Instance functions belong to instances and can only be invoked by them.
- **instance variable** A nonstatic data member of a class. An instance variable belongs to an instance of the class.
- **instantiation** The process of creating an object of a class.

- **no-arg constructor** A constructor without arguments.
- **object-oriented programming (OOP)** An approach to programming that involves organizing objects and their behavior into classes of reusable components.
- **Unified Modeling Language** (**UML**) A graphical notation for describing classes and their relationships.
- **private** A modifier for members of a class. A private member can only be referenced inside the class.
- **public** A modifier for classes, data, and functions that can be accessed by all programs.
- **scope resolution operator (::)** specifies the class for the functions and constructors.
- **this** Refers to the object itself.

Chapter 10

- **aggregation** A special form of association that represents an ownership relationship between two classes.
- **composition** A form of relationship that represents exclusive ownership of the class by the aggregated class.
- **copy constructor** Each class has a special constructor, called copy constructor, which is used to create an object initialized with another object's data.
- **deep copy** When cloning an object, all its fields are cloned recursively.
- **destructor** Destructors are the opposite of constructors. A constructor is invoked when an object is created and a destructor is invoked when the object is destroyed. Every class has a default destructor if the destructor is not explicitly defined.
- **friend keyword** Destructors are the opposite of constructors. A constructor is invoked when an object is create
- **has-a relationship** Same as composition.
- **immutable object** The contents of the object cannot be changed.
- **immutable class** A class is immutable if it contains all private data fields and no mutator functions and no accessor functions that would return a reference to a mutable data field object.
- **include guard** A C++ directive to avoid duplicate inclusion of the header files.
- **instance field** A data member belongs to an object. Each object has its independent value.
- **instance function** A function that must be invoked from an object.
- **mutator function** (**setter**) A function that changes the value of a private field in an object.
- **static field** A data member declared using the static modifier. A static variable is shared by all instances of that class. Static variables are used to communicate between different objects of the same class and to handle global states among these objects.
- **static function** A function that can be invoked without creating an instance of the class. To define static functions, put the modifier static in the function declaration.

- **shallow copy** When cloning an object, all its fields are copied. If a field is a reference type, its data members are not copied.
- **vector** A C++ class that represents a resizable array.

Chapter 11

- **abstract class** When you are designing classes, a superclass should contain common features that are shared by subclasses. Sometimes the superclass is so abstract that it cannot have any specific instances. These classes are called abstract classes and are declared using the abstract modifier. Abstract classes are like regular classes with data and functions, but you cannot create instances of abstract classes.
- **abstract function** A function signature without implementation. Its implementation is provided by its subclasses. An abstract function is denoted with an abstract modifier and must be contained in an abstract class. In a nonabstract subclass extended from an abstract class, all abstract functions must be implemented, even if they are not used in the subclass.
- **base class** A class inherited by a subclass.
- **constructor chaining** Constructing an instance of a class invokes all the constructor, chaining base classes along the inheritance chain.
- **derived class** A class that inherits from or extends a base class.
- **destructor chaining** Destructing an instance of a class invokes all the destructors along the inheritance chain with the derived class's destructor invoked first.
- **downcasting** Casting an object from a superclass type to subclass.
- **dynamic binding** A function may be defined in a superclass, but is overridden in a subclass. Which implementation of the function is used on a particular call will be determined dynamically at runtime. This capability is known as dynamic binding.
- **generic programming** Allows functions to be used generically for a wide range of object arguments through polymorphism.
- **inheritance** Declaring a new class by extending an existing class.
- **is-a relationship** Same as inheritance.
- **override** In C++, redefining a virtual function in a derived class is called overriding a function.
- **polymorphism** Refers to the feature that an object of a subclass can be used by any code designed to work with an object of its superclass.
- **protected** A modifier for members of a class. A protected member of a class can be used in the class in

which it is declared or any subclass derived from that class.

- **pure virtual function** A function declaration without implementation. Also known as abstract function.
- **redefine** Implement the function in a subclass that is declared in a superclass.
- **upcasting** Casting an object from a subclass type to superclass.
- **virtual function** A function declared with the keyword virtual. In C++, redefining a virtual function in a derived class is called overriding a function.

Chapter 12

- **absolute file name** Complete file name with drive letter such as c:\example\Test.cpp.
- **binary I/O** Binary I/O interprets data as raw binary values.
- **file open mode** Specify how to open the file (such as input, output, append, or binary).
- **fstream** File stream class.
- **ifstream** Input file stream class.
- **ofstream** Input file stream class.
- **random access file** The file that can be both read and written in any order.
- **relative file name** File names with drive letters.
- **sequential access file** The file is read or written sequentially from beginning to end.
- **stream** A stream is an object that facilitates input or output. For input, it is called an input stream. For output, it is called an output stream.
- **stream state** Indicating the state of the stream such as end, error, etc.
- **text I/O** Text I/O interprets data in sequences of characters.

Chapter 14

- **exception** An unexpected event indicating that a program has failed in some way. Exceptions can be handled in a try-catch block.
- **exception specification** Specifies the type of exception a function may throw.
- **rethrow exception** After catching an exception in a catch clause, you may rethrow the exception.
- **standard exception** C++ provides several standard exception classes such as runtime_error, overflow_error.
- **throw exception** The statement to throw an exception.
- **unchecked function** C++ provides a function named unexpected. This function is invoked when an exception is thrown, but the exception is not declared in the exception specification.

Chapter 15

- **template class** A generic class with generic types.
- **template function** A generic function with generic types.
- **template prefix** Specifies generic types before a class or function.
- **type parameter** Generic type

Chapter 16

- **circular doubly linked list** A circular, doubly linked list is doubly linked list, except that the forward pointer of the last node points to the first node and the backward pointer of the first pointer points to the last node.
- **circular linked list** A *circular, singly linked list* is like a singly linked list, except that the pointer of the last node points back to the first node.
- **deque** A queue operation to remove an element from the queue.
- **doubly linked list** A doubly linked list contains the nodes with two pointers. One points to the next node and the other points to the previous node. These two pointers are conveniently called *a forward pointer* and *a backward pointer*. So, a doubly linked list can be traversed forward and backward.
- **endeque** A queue operation to append an element to the queue.
- **linked list** In a linked list, each element is contained in a structure, called the node. When a new element is added to the list, a node is created to contain the element. All the nodes are chained through pointers.
- **peek** A stack operation to read the top element.
- **push** A stack operation to add an element to the top.
- **queue** Represents a waiting list, where insertions take place at the back (also referred to as the tail of) of a queue and deletions take place from the front (also referred to as the head of) of a queue.
- **singly linked list** same as linked list.
- **stack** A special type of the list where insertions and deletions take place only at the one end, referred to as the *top* of a stack.

Chapter 17

- **binary search tree** A binary tree with no duplicate elements. For every node in the tree the value of any node in its left subtree is less than the value of the node and the value of any node in its right subtree is greater than the value of the node.
- **binary tree** A data structure to support searching, sorting, inserting, and deleting data efficiently.
- **heap** **A** binary tree with the following properties: It is a complete binary tree. Each node is greater than or equal to any of its children.
- **inorder traversal** Visit the left subtree of the current node first, then the current node itself, and finally the right subtree of the current node.
- **postorder traversal** Visit the left subtree of the current node first, then the right subtree of the current node, and finally the current node itself.
- **preorder traversal** Visit the current node first, then the left subtree of the current node, and finally the right subtree of the current node.
- **priority queue** In a priority queue, elements are assigned with priorities. The element with the highest priority is accessed or removed first.
- **queue** A regular queue is a first-in and first-out data structure. Elements are appended to the end of the queue and are removed from the beginning of the queue.
- **tree traversal** A process of visiting each node in the tree exactly once.

Chapter 18

- **average-time analysis** An average-case analysis attempts to determine the average amount of time among all possible input of the same size.
- **best-time analysis** An input that results in the shortest execution time is called the best-case input. The analysis to find the best-case time is known as worst-time analysis.
- **big O notation** Comparing algorithms by examining their growth rates. This notation allows you to ignore constants and smaller terms while focusing on the dominating terms.
- **bubble sort** The bubble sort algorithm makes several passes through the array. On each pass, successive neighboring pairs are compared. If a pair is in

decreasing order, its values are swapped; otherwise, the values remain unchanged. The technique is called a bubble sort or sinking sort because the smaller values gradually "bubble" their way to the top and the larger values sink to the bottom.

- **constant time** The Big O notation estimates the execution time of an algorithm in relation to the input size. If the time is not related to the input size, the algorithm is said to take constant time with the notation $O(1)$.
- **exponential time** An algorithm with the $O(c^n)$ time complexity is called an exponential algorithm. As the input size increases, the time for the exponential algorithm grows exponentially. The exponential algorithms are not practical for large input size.
- **growth rate** measures how fast the time complexity of an algorithm grows as the input size grows.
- **heap sort** Heap sort uses a binary heap to sort an array.
- **logarithmic time** An algorithm with the $O(\log n)$ time complexity is called a logarithmic algorithm.
- **quadratic time** An algorithm with the $O(n^2)$ time complexity is called a quadratic algorithm.
- **merge sort** The merge sort algorithm can be described recursively as follows: The algorithm divides the array into two halves and applies merge sort on each half recursively. After the two halves are sorted, merge them.
- **quick sort** Quick sort, developed by C. A. R. Hoare (1962), works as follows: The algorithm selects an element, called the pivot, in the array. Divide the array into two parts such that all the elements in the first part are less than or equal to the pivot and all the elements in the second part are greater than the pivot. Recursively apply the quick sort algorithm to the first part and then the second part.
- **worst-time analysis** An input that results in the longest execution time is called the worst-case input. The analysis to find the worst-case time is known as worst-time analysis.

Chapter 19

- **associative container** Associative containers are non-linear containers that can locate elements stored in the container quickly. Such containers can store

sets of values or *key/value* pairs. The four associative containers are set, multiset, map, and multimap.

- **bidirectional iterator** A bidirectional iterator is a forward iterator with the capability of moving backward. The iterator can be moved freely back or forth one at a time.
- **container** Classes in the STL are container classes. A container object such as a vector is used to store a collection of data, often referred to as elements.
- **container adapter** Container adapters are constrained versions of sequence containers. They are adapted from sequence containers for handling special cases. The three container adapters are stack, queue, and priority_queue.
- **deque** The term deque stands for double-ended queue. A deque provides efficient operations to support insertion and deletion on both ends of a deque.
- **first-class container** Sequence containers and associative containers are known as the first-class container.
- **forward iterator** A forward iterator combines all the functionalities of input and output iterators to support both read and write operations.
- **input iterator** An input iterator is used for reading an element from a container. It can move only in a forward direction one element at a time.
- **istream_iterator** Used for reading an element from the console.
- **iterator** Similar to a pointer, used extensively in the first-class containers for accessing and manipulating the elements.
- **list** The class list is implemented as a doubly-linked list. It supports efficient insertion and deletion operations anywhere on the list.
- **map** Each element in a map is a pair. The first value in the pair is the key and the second value is associated with the key. A map provides quick access to value using the key.
- **multiset** Stores elements (may contain duplicates).
- **multimap** Same as map, except that a multimap allows duplicate keys, but a map does
- **ostream_iterator** Used for outputting an element to the console.
- **output iterator** An output iterator is used for writing an element to a container.
- **priority queue** A STL class. In a priority queue, elements are assigned with priorities. The element with

the highest priority is accessed or removed first. A priority queue has a largest-in, first-out behavior.

- **random-access iterator** A random access iterator is a bidirectional iterator with the capability of accessing any element in any order, i.e., to jump forward or backward by a number of elements.
- **queue** A STL container. A queue is a first-in/first-out container.
- **sequence container** The sequence containers (also known as sequential containers) represent linear data structures. The three sequence containers are vector, list, and deque (pronounced deck).
- **set** An STL associative containter. Stores nonduplicate elements.
- **STL algorithm** Algorithms are used in the functions to manipulate data such as sorting, searching, and comparing elements.
- **Vector** An STL sequence container for storing elements. The vector class is implemented using array. It is efficient to appending and searching operations.