10

Operator Overloading; Class `string`

Objectives

In this chapter you'll:

- Learn how operator overloading can help you craft valuable classes.
- Overload unary and binary operators.
- Convert objects from one class to another.
- Use overloaded operators and additional features of class `string`.
- Create `PhoneNumber`, `Date` and `Array` classes that provide overloaded operators.
- Perform dynamic memory allocation with `new` and `delete`.
- Understand how keyword `explicit` prevents a constructor from being used for implicit conversions.
- Experience a "light-bulb moment" when you'll truly appreciate the elegance and beauty of the class concept.

Self-Review Exercises

10.1 Fill in the blanks in each of the following:

a) Suppose `a` and `b` are integer variables and we form the sum `a + b`. Now suppose `c` and `d` are floating-point variables and we form the sum `c + d`. The two `+` operators here are clearly being used for different purposes. This is an example of _____.

ANS: operator overloading.

b) Keyword _____ introduces an overloaded-operator function definition.

ANS: `operator`.

c) To use operators on class objects, they must be overloaded, with the exception of operators _____, _____ and _____.

ANS: assignment (`=`), address (`&`), comma (`,`).

d) The _____, _____ and _____ of an operator cannot be changed by overloading the operator.

ANS: precedence, associativity, "arity."

e) The operators that cannot be overloaded are _____, _____, _____ and _____.

ANS: `.`, `?:`, `.*`, and `::`.

f) The _____ operator reclaims memory previously allocated by `new`.

ANS: `delete`.

g) The _____ operator dynamically allocates memory for an object of a specified type and returns a(n) _____ to that type.

ANS: `new`, pointer.

h) Append a(n) _____ after the closing quote (") of a string literal to indicate that it's a C++14 `string`-object literal.

ANS: `s`.

10.2 Explain the multiple meanings of the operators << and >>.

ANS: Operator >> is both the right-shift operator and the stream extraction operator, depending on its context. Operator << is both the left-shift operator and the stream insertion operator, depending on its context.

10.3 In what context might the name `operator/` be used?

ANS: For operator overloading: It would be the name of a function that would provide an overloaded version of the `/` operator for a specific class.

10.4 (True/False) Only existing operators can be overloaded.

ANS: True.

10.5 How does the precedence of an overloaded operator compare with the precedence of the original operator?

ANS: The precedence is identical.

Exercises

10.6 *(Memory Allocation and Deallocation Operators)* Compare and contrast dynamic memory allocation and deallocation operators `new`, `new []`, `delete` and `delete[]`.

ANS: Operator `new` creates an object and dynamically allocates space in memory for that object. Operator `delete` destroys a dynamically allocated object and frees the space that was occupied by the object. Operators `new []` and `delete []` are used to allocate and deallocate arrays dynamically. Operator `delete []` ensures that the destrctor is called for each element in the array before the array is reclaimed.

10.8 *(`Complex` Class)*

ANS: [*Note:* In the solution, the == operator compares double values—a practice that is error-prone due to the way computers handle floating-point values. You may want to modify the == operator to take the difference between two doubles and determining if the difference is less than a threshold value such as `0.00001`.]

10.9 *(`HugeInt` Class)*

ANS: [*Note:* This solution does not handle negative values, or cases when a value causes overflow. Also, the solution overloads the - operator, but this is not necessary—students need to implement subtraction of `HugeInt`s to implement division, but they can use a `private` helper function if they wish.]