Input, Output, and Miscellaneous Operators

Lecture 7
Sections 2.2, 3.1 - 3.2, 3.6

Robb T. Koether

Hampden-Sydney College

Mon, Sep 10, 2018

Input and Output Streams

- Input and output use streams.
- A stream is a mechanism that allows us to pass information back and forth between our program and the input and output devices.
- Think of a stream as a sequence of characters sent by one device and received by the other.
- Input streams are objects of the `istream` class.
- Output streams are objects of the `ostream` class.

Buffered Input and Output

- An input buffer is a portion of memory where the data in the input stream (characters typed at the keyboard) are stored until the program is ready to read them.
- An output buffer is a portion of memory where the data output by the program are stored until the program is ready to display them.
- Unbuffered output moves directly to the output device, character by character.

Standard Input

- Standard input refers to the keyboard.
- Standard input is an `istream` object named `cin`.
- Standard input is buffered.
- The buffer contains the sequence of characters typed at the keyboard.
- `cin` analyzes the characters in the buffer to determine the value of the input, according to the data type being read.

The Extraction Operator

The Input Operator

```
char c;
int a;
float b;
cin >> c >> a >> b;
```

- The operator `>>` is the extraction, or input, operator.
- Values may be extracted from an input stream to named objects only.

Standard Output

- Standard output refers to the text window displayed on the monitor.
- Standard output is an `ostream` object named `cout`.
- Standard output is buffered.
- `cout` converts values into their character representations and stores the characters in the buffer.
- At appropriate times, the characters in the buffer are displayed at the monitor.

The Insertion Operator

The Input Operator

```
int a = 123;
int b = 456;
cout << "The sum of " << a << " and " << b
     << " is " << a + b << endl;
```

- The operator << is the insertion, or output, operator.
- Values of constants, named objects, and expressions my be inserted into an output stream.

Compound Assignment Operators

- The operator `+=` means "add to."
- The statement

  ```
  x += y;
  ```

  is equivalent to

  ```
  x = x + y;
  ```

Compound Assignment Operators

Compound Assignment Operators

```
x += y;
x -= y;
x *= y;
x /= y;
x %= y;     // Integers only
```

- Common compound-assignment operators:

Examples: Compound Assignment

Find the Value

```
int a = 12;
a += 8;
a -= 5;
a *= 4;
a /= 5;
a %= 8;
```

- What is the value of `a`?

Increment and Decrement Operators

- To increment is to add 1.
- To decrement is to subtract 1.
- The increment operator is $++$.
- The decrement operator is $--$.
- These operators may be applied only to named objects.

Pre- and Post-Increment

- To pre-increment an object means to increment it before using it in the expression.
- To pre-increment, write the operator before the object: `++x`
- To post-increment an object means to increment it after using it in the expression.
- To post-increment, write the operator after the object: `x++`
- The same goes for decrement.

Pre- and Post-Increment

Pre- and Post-Incredment

```
x = 3;
x++;                    // This is fine
y = ++x;                // Not recommended
z = x++;                // Not recommended
w = (++(++x))++;        // Never do this
v = x+++x+++x;          // Or this
```

- What are the values of `y`, `z`, `w`, and `v`?
- To avoid trouble, never use `++` or `--` in conjunction with any other operator.

Sample Programs

- Examples
  - `IOBuffer.cpp`
  - `SumOfList.cpp`

Assignment

Assignment

- Read Sections 2.2, 3.1 - 3.2, 3.6.