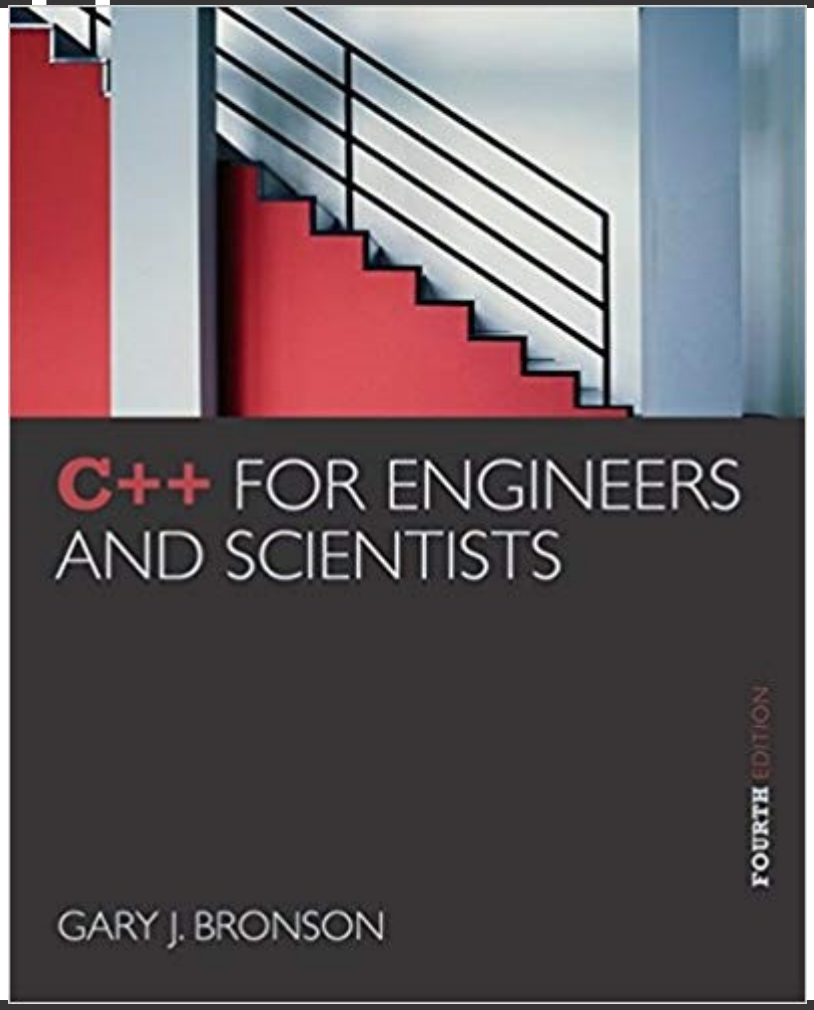
ELEG 1043
Computer Applications in Engineering



Chapter 3: Assignment, Formatting, and Interactive Input

C++ FOR ENGINEERS AND SCIENTISTS

Acknowledgement

- Some of the slides or images are from various sources. The copyright of those materials belongs to their original owners.

Objectives

In this chapter, you will learn about:

- Assignment operations
- Formatting numbers for program output
- Using mathematical library functions
- Program input using the **`cin`** object
- Symbolic constants
- A case study involving acid rain
- Common programming errors

Objectives

In this chapter, you will learn about:

- Assignment operations
- Formatting numbers for program output
- Using mathematical library functions
- Program input using the `cin` object
- Symbolic constants
- A case study involving acid rain
- Common programming errors

Assignment Operations

- **Assignment Statement:** Assigns the value of the expression on the right side of the = to the variable on the left side of the =
  - int a = 2;
    left right
- Another assignment statement using the same variable will overwrite the previous value with the new value

  Examples:

```
slope = 3.7;
slope = 6.28; (Overwrite)
```

Assignment Operations (continued)

- Right side of an assignment statement may contain any expression that can be evaluated to a value

  Examples:

```
newtotal = 18.3 + total;
taxes = .06*amount;
average = sum / items;
```

- Only one variable can be on the left side of an assignment statement

Assignment Operations (continued)

Program 3.1

```
// This program calculates the volume of a cylinder,
// given its radius and height
#include <iostream>
using namespace std;

int main()
{
  double radius, height, volume;
  radius = 2.5;
  height = 16.0;
  volume = 3.1416 * radius * radius * height;
  cout << "The volume of the cylinder is " << volume << endl;

  return 0;
}
```

Assignment Operations (continued)

- **Assignment operator:** The = sign
- **C++ statement:** Any expression terminated by a semicolon
- Multiple assignments in the same expression are possible

  Example:

```
a = b = c = 25;
```

Assignment Operations (continued)

- **Coercion:** Forcing a data value to another data type
  - Value of the expression on the right side of an assignment statement will be coerced (converted) to the data type of the variable on the left side during evaluation

```
double a = 4.3;
int b = (int) a;
```

  - Variable on the left side may also be used on the right side of another assignment statement

Assignment Operations (continued)

- **Accumulation statement:** Has the effect of accumulating, or totaling

  Syntax:

```
variable = variable + newValue;
```

```
int a = 0;
a = a + 2;
```

Assignment Operations (continued)

- Additional assignment operators provide short cuts: **+=**, **-=**, ***=**, **/=**, **%=**

  Example:

```
sum = sum + 10;
```

  is equivalent to: **sum += 10;**

```
price *= rate +1;
```

  is equivalent to:

```
price = price * (rate + 1);
```

Assignment Operations (continued)

- **Increment operator ++:** Unary operator for the special case when a variable is increased by 1
- **Prefix increment operator** appears before the variable
  - Example: `++i`
- **Postfix increment operator** appears after the variable
  - Example: `i++`

Assignment Operations (continued)

- Example: `k = ++n;  //prefix increment`

  is equivalent to:

```
n = n + 1;  //increment n first
k = n;      //assign n's value to k
```

- Example: `k = n++;  //postfix increment`

  is equivalent to

```
k = n;      //assign n's value to k
n = n + 1;  //and then increment n
```

Assignment Operations (continued)

- **Decrement operator `--`:** Unary operator for the special case when a variable is decreased by 1
- **Prefix decrement operator** appears before the variable
  - Example: `--i;`
- **Postfix decrement operator** appears after the variable
  - Example: `i--;`

Formatting Numbers for Program Output

- Proper output formatting contributes to ease of use and user satisfaction
- `cout` with stream manipulators can control output formatting

Formatting Numbers for Program Output (continued)

- Formatting floating-point numbers requires three field-width manipulators to:
  - Set the total width of the display
  - Force a decimal place
  - Set the number of significant digits after the decimal point
- Example:

```
#include <iomanip>
cout << "|" << setw(10) << fixed
     << setprecision(3) << 25.67 << "|";
```

produces this output: `|    25.670|`

Formatting Numbers for Program Output (continued)

- Manipulators affect only output; the value stored internally does not change
- Manipulators can also be set using the **`ostream`** class methods
- Separate the **`cout`** object name from the method name with a period

  Example:

```
cout.precision(2)
```

Formatting Numbers for Program Output (continued)

Method	Comment	Example
`precision(n)`	Equivalent to `setprecision()`	`cout.precision(2)`
`fill('x')`	Equivalent to `setfill()`	`cout.fill('*')`
`setf(ios::fixed)`	Equivalent to `cout.setf (ios::fixed)`	`setiosflags(ios::fixed)`
`setf(ios::showpoint)`	Equivalent to `cout.setf (ios::showpoint)`	`setiosflags(ios::showpoint)`
`setf(ios::left)`	Equivalent to `left`	`cout.setf(ios::left)`
`setf(ios::right)`	Equivalent to `right`	`cout.setf(ios::right)`
`setf(ios::flush)`	Equivalent to `endl`	`cout.setf(ios::flush)`

Table 3.4 `ostream` Class Functions