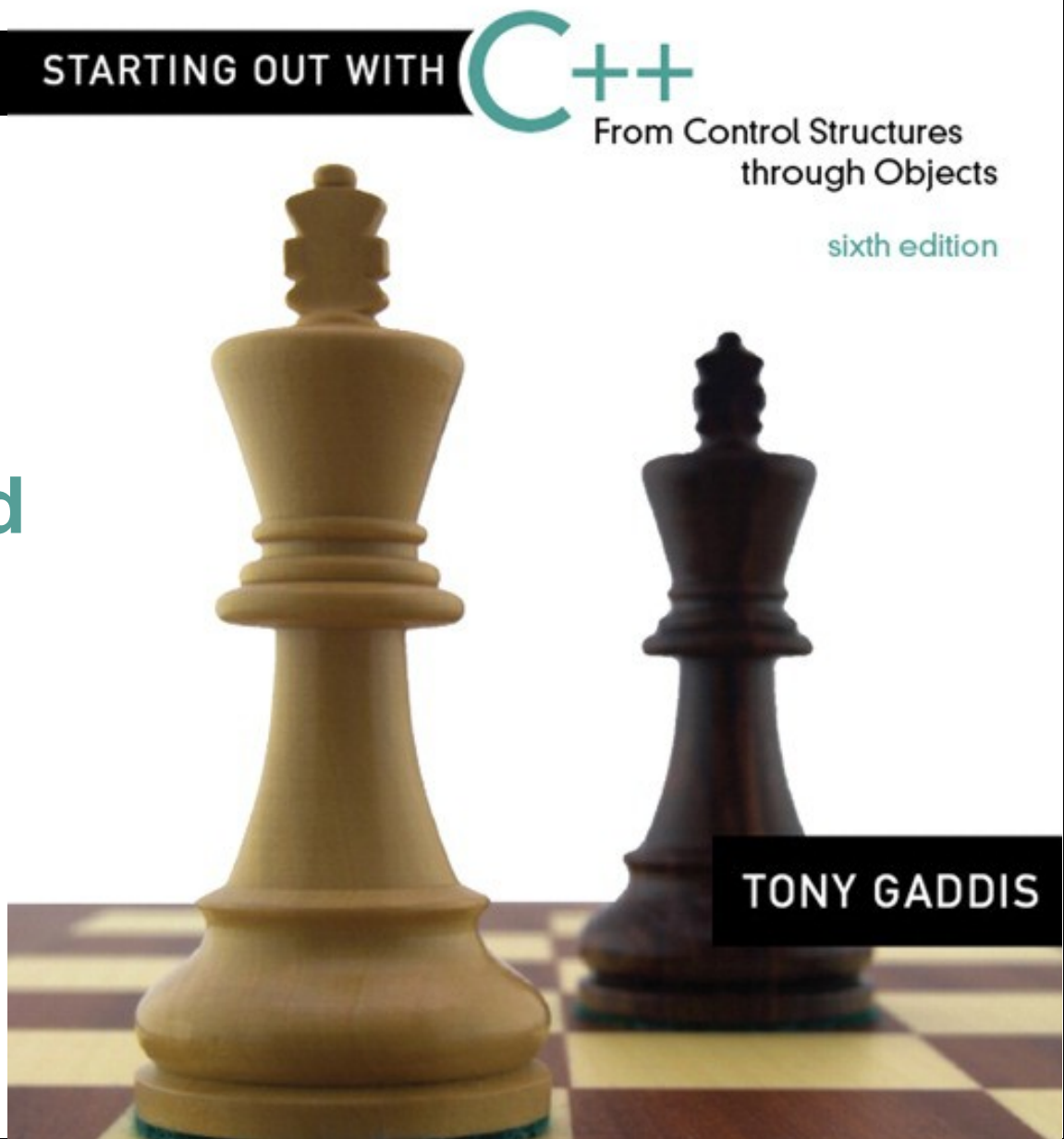
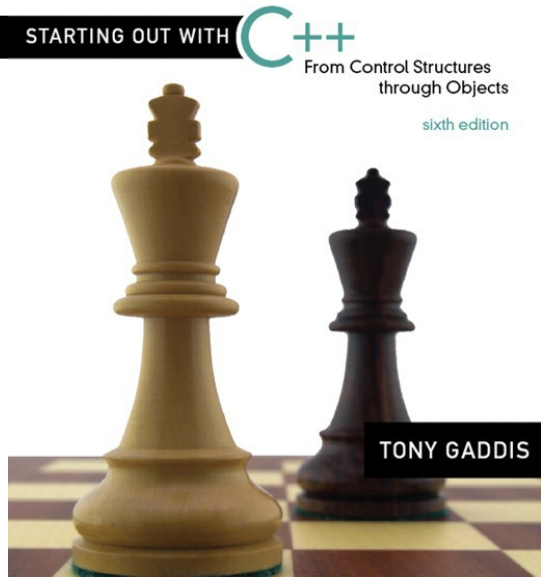


Chapter 3:

Expressions and Interactivity



3.1

The `cin` Object

The `cin` Object

- Standard input object
- Like `cout`, requires `iostream` file
- Used to read input from keyboard
- Information retrieved from `cin` with `>>`
- Input is stored in one or more variables

Program 3-1

```
1  // This program asks the user to enter the length and width of
2  // a rectangle. It calculates the rectangle's area and displays
3  // the value on the screen.
4  #include <iostream>
5  using namespace std;
6
7  int main()
8  {
9     int length, width, area;
10
11    cout << "This program calculates the area of a ";
12    cout << "rectangle.\n";
13    cout << "What is the length of the rectangle? ";
14    cin >> length;
15    cout << "What is the width of the rectangle? ";
16    cin >> width;
17    area = length * width;
18    cout << "The area of the rectangle is " << area << ".\n";
19    return 0;
20 }
```

Program Output with Example Input Shown in Bold

```
This program calculates the area of a rectangle.
What is the length of the rectangle? 10 [Enter]
What is the width of the rectangle? 20 [Enter]
The area of the rectangle is 200.
```

The `cin` Object

- **`cin`** converts data to the type that matches the variable:

```
int height;
cout << "How tall is the room? ";
cin >> height;
```

Displaying a Prompt

- A prompt is a message that instructs the user to enter data.
- You should always use **`cout`** to display a prompt before each **`cin`** statement.

```
cout << "How tall is the room? ";
cin >> height;
```

The `cin` Object

- Can be used to input more than one value:

```
cin >> height >> width;
```

- Multiple values from keyboard must be separated by spaces

- Order is important: first value entered goes to first variable, etc.

Program 3-2

```
1  // This program asks the user to enter the length and width of
2  // a rectangle. It calculates the rectangle's area and displays
3  // the value on the screen.
4  #include <iostream>
5  using namespace std;
6
7  int main()
8  {
9     int length, width, area;
10
11    cout << "This program calculates the area of a ";
12    cout << "rectangle.\n";
13    cout << "Enter the length and width of the rectangle ";
14    cout << "separated by a space.\n";
15    cin >> length >> width;
16    area = length * width;
17    cout << "The area of the rectangle is " << area << endl;
18    return 0;
19 }
```

Program Output with Example Input Shown in Bold

```
This program calculates the area of a rectangle.
Enter the length and width of the rectangle separated by a space.
10 20 [Enter]
The area of the rectangle is 200
```

Reading Strings with `cin`

- Can be used to read in a string
- Must first declare an array to hold characters in string:

```
char myName[21];
```

- `nyName` is name of array, `21` is the number of characters that can be stored (the size of the array), including the NULL character at the end
- Can be used with `cin` to assign a value:

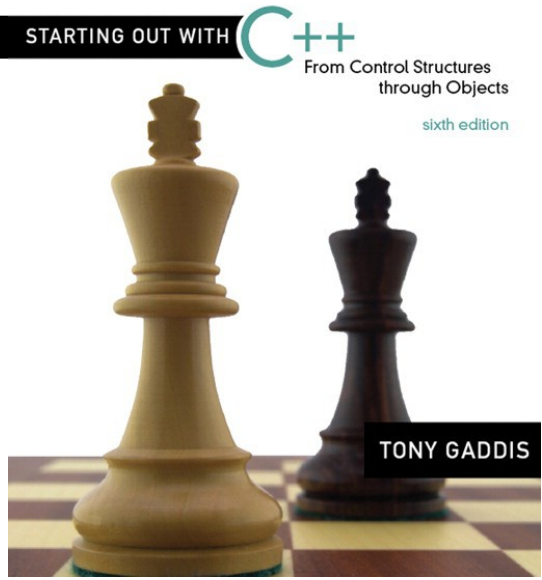
```
cin >> myName;
```

Program 3-4

```
1  // This program demonstrates how cin can read a string into
2  // a character array.
3  #include <iostream>
4  using namespace std;
5
6  int main()
7  {
8     char name[21];
9
10    cout << "What is your name? ";
11    cin >> name;
12    cout << "Good morning " << name << endl;
13    return 0;
14 }
```

Program Output with Example Input Shown in Bold

```
What is your name? Charlie [Enter]
Good morning Charlie
```



3.2

Mathematical Expressions

Mathematical Expressions

- Can create complex expressions using multiple mathematical operators
- An expression can be a literal, a variable, or a mathematical combination of constants and variables
- Can be used in assignment, `cout`, other statements:

```
area = 2 * PI * radius;
cout << "border is: " << 2*(l+w);
```

Order of Operations

In an expression with more than one operator, evaluate in this order:

- `-` (unary negation), in order, left to right
- `*` `/` `%`, in order, left to right
- `+` `-`, in order, left to right

In the expression `2 + 2 * 2 - 2`

Order of Operations

Table 3-2 Some Expressions

Expression	Value
`5 + 2 * 4`	13
`10 / 2 - 3`	2
`8 + 12 * 2 - 4`	28
`4 + 17 % 2 - 1`	4
`6 - 3 * 2 + 7 - 1`	6

Associativity of Operators

- – (unary negation) associates right to left
- *, /, %, +, – associate right to left
- parentheses ( ) can be used to override the order of operations:

```
 2 + 2   *   2 – 2   = 4
(2 + 2)  *   2 – 2   = 6
 2 + 2   *  (2 – 2)  = 2
(2 + 2)  *  (2 – 2)  = 0
```

Grouping with Parentheses

Table 3-4 More Expressions

Expression	Value
`(5 + 2) * 4`	28
`10 / (5 - 3)`	5
`8 + 12 * (6 - 2)`	56
`(4 + 17) % 2 - 1`	0
`(6 - 3) * (2 + 7) / 3`	9

Algebraic Expressions

- Multiplication requires an operator:

  $Area = lw$ is written as `Area = l * w;`

- There is no exponentiation operator:

  $Area = s^2$ is written as `Area = pow(s, 2);`

- Parentheses may be needed to maintain order of operations:

$$m = \frac{y_2 - y_1}{x_2 - x_1}$$

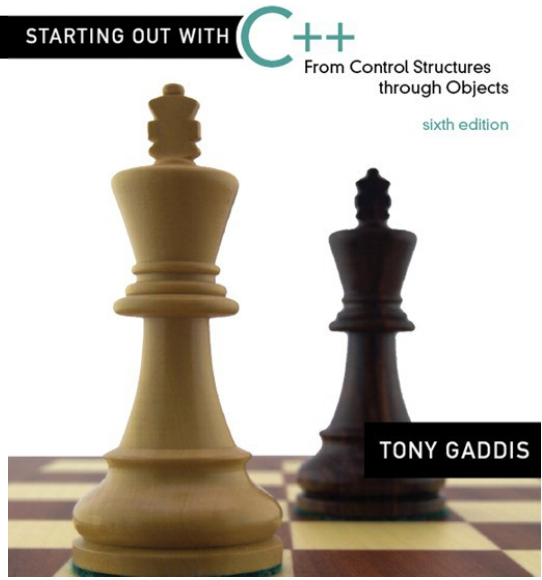
is written as

```
m = (y2-y1) /(x2-x1);
```

Algebraic Expressions

Table 3-5 Algebraic and C++ Multiplication Expressions

Algebraic Expression	Operation	C++ Equivalent
$6B$	`6 times B`	`6 * B`
$(3)(12)$	`3 times 12`	`3 * 12`
$4xy$	`4 times x times y`	`4 * x * y`



3.3

When You Mix Apples and Oranges: *Type Conversion*

When You Mix Apples and Oranges: *Type Conversion*

- Operations are performed between operands of the same type.
- If not of the same type, C++ will convert one to be the type of the other
- This can impact the results of calculations.

Hierarchy of Types

Highest: `long double`
`double`
`float`
`unsigned long`
`long`
`unsigned int`
Lowest: `int`

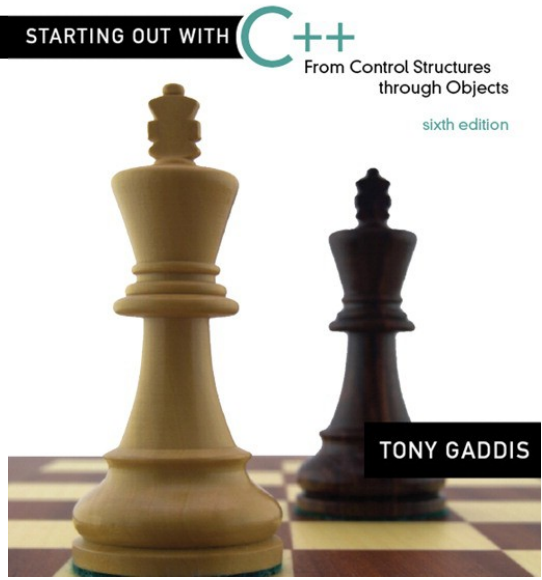
Ranked by largest number they can hold

Type Coercion

- Type Coercion: automatic conversion of an operand to another data type
- Promotion: convert to a higher type
- Demotion: convert to a lower type

Coercion Rules

1) `char`, `short`, `unsigned short` automatically promoted to `int`
2) When operating on values of different data types, the lower one is promoted to the type of the higher one.
3) When using the = operator, the type of expression on right will be converted to type of variable on left

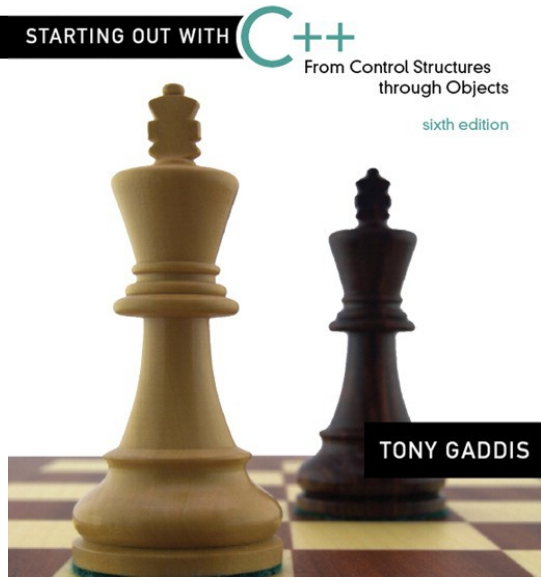


3.4

Overflow and Underflow

Overflow and Underflow

- Occurs when assigning a value that is too large (overflow) or too small (underflow) to be held in a variable
- Variable contains value that is ‘wrapped around’ set of possible values
- Different systems may display a warning/error message, stop the program, or continue execution using the incorrect value



3.5

Type Casting

Type Casting

- Used for manual data type conversion
- Useful for floating point division using ints:

```
double m;
m = static_cast<double>(y2-y1)
                        /(x2-x1);
```

- Useful to see `int` value of a `char` variable:

```
char ch = 'C';
cout << ch << " is "
     << static_cast<int>(ch);
```

Program 3-11

```
1  // This program uses a type cast to avoid integer division.
2  #include <iostream>
3  using namespace std;
4
5  int main()
6  {
7     int books;         // Number of books to read
8     int months;        // Number of months spent reading
9     double perMonth;   // Average number of books per month
10
11    cout << "How many books do you plan to read? ";
12    cin >> books;
13    cout << "How many months will it take you to read them? ";
14    cin >> months;
15    perMonth = static_cast<double>(books) / months;
16    cout << "That is " << perMonth << " books per month.\n";
17    return 0;
18 }
```

Program Output with Example Input Shown in Bold

```
How many books do you plan to read? 30 [Enter]
How many months will it take you to read them? 7 [Enter]
That is 4.28571 books per month.
```

C-Style and Prestandard Type Cast Expressions

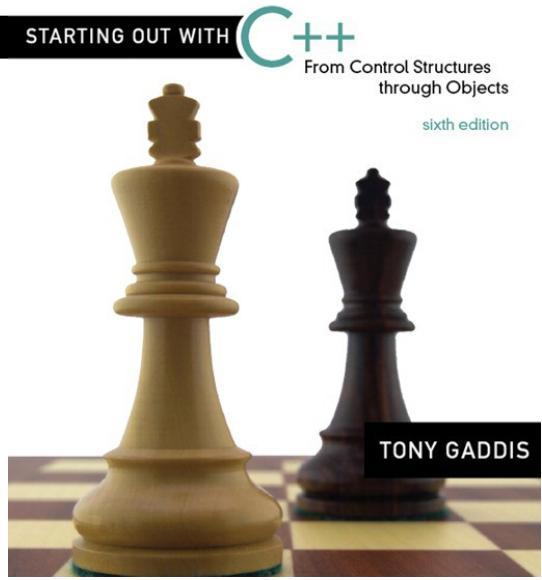
- C-Style cast: data type name in `()`

```
cout << ch << " is " << (int)ch;
```

- Prestandard C++ cast: value in `()`

```
cout << ch << " is " << int(ch);
```

- Both are still supported in C++, although `static_cast` is preferred



3.6

Named Constants

Named Constants

- Named constant (constant variable): variable whose content cannot be changed during program execution
- Used for representing constant values with descriptive names:

```
const double TAX_RATE = 0.0675;
const int NUM_STATES = 50;
```

- Often named in uppercase letters

Program 3-13

```
1  // This program calculates the area of a circle.
2  // The formula for the area of a circle is PI times
3  // the radius squared. PI is 3.14159.
4  #include <iostream>
5  #include <cmath>   // needed for pow function
6  using namespace std;
7
8  int main()
9  {
10    const double PI = 3.14159;
11    double area, radius;
12
13    cout << "This program calculates the area of a circle.\n";
14    cout << "What is the radius of the circle? ";
15    cin >> radius;
16    area = PI * pow(radius, 2.0);
17    cout << "The area is " << area << endl;
18    return 0;
19 }
```

Constants and Array Sizes

- It is a common practice to use a named constant to indicate the size of an array:

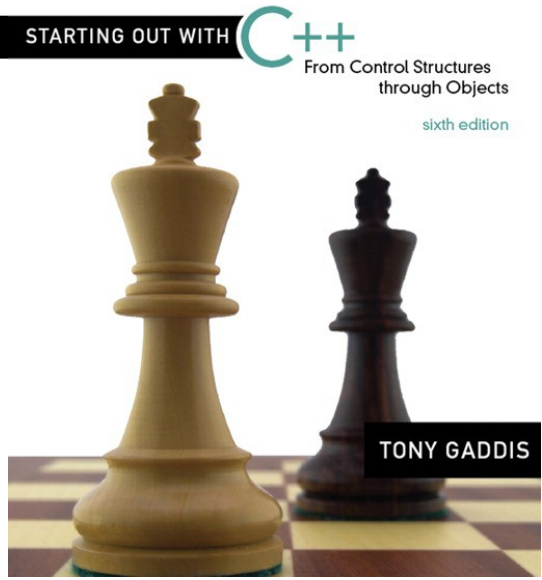
```
const int SIZE = 21;
char name[SIZE];
```

const vs. #define

- `#define` – C-style of naming constants:

```
#define NUM_STATES 50
```

  - Note no semicolon at end
- Interpreted by pre-processor rather than compiler
- Does not occupy memory location like `const`



3.7

Multiple Assignment and Combined Assignment

Multiple Assignment and Combined Assignment

- The = can be used to assign a value to multiple variables:

```
x = y = z = 5;
```

- Value of = is the value that is assigned
- Associates right to left:

```
x = (y = (z = 5));
```

value is 5 — value is 5 — value is 5

Combined Assignment

- Look at the following statement:

```
sum = sum + 1;
```

This adds 1 to the variable **`sum`**.

Other Similar Statements

Table 3-8 (Assume x = 6)

Statement	What It Does	Value of x After the Statement
`x = x + 4;`	Adds 4 to x	10
`x = x - 3;`	Subtracts 3 from x	3
`x = x * 10;`	Multiplies x by 10	60
`x = x / 2;`	Divides x by 2	3
`x = x % 4`	Makes x the remainder of x / 4	2

Combined Assignment

- The combined assignment operators provide a shorthand for these types of statements.
- The statement

```
sum = sum + 1;
```

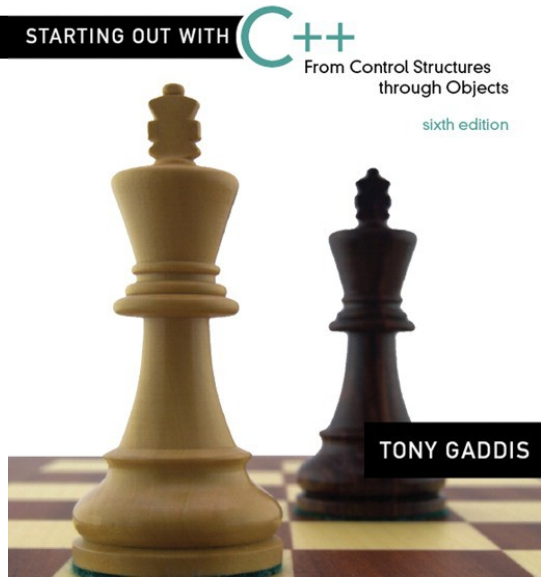
is equivalent to

```
sum += 1;
```

Combined Assignment Operators

Table 3-9

Operator	Example Usage	Equivalent to
`+=`	`x += 5;`	`x = x + 5;`
`-=`	`y -= 2;`	`y = y - 2;`
`*=`	`z *= 10;`	`z = z * 10;`
`/=`	`a /= b;`	`a = a / b;`
`%=`	`c %= 3;`	`c = c % 3;`



3.8

Formatting Output

Formatting Output

- Can control how output displays for numeric, string data:
  - size
  - position
  - number of digits
- Requires `iomanip` header file

Stream Manipulators

- Used to control how an output field is displayed
- Some affect just the next value displayed:
  - `setw(x)`: print in a field at least `x` spaces wide. Use more spaces if field is not wide enough

Program 3-17

```
1  // This program displays three rows of numbers.
2  #include <iostream>
3  #include <iomanip>      // Required for setw
4  using namespace std;
5
6  int main()
7  {
8     int num1 = 2897, num2 = 5,    num3 = 837,
9         num4 = 34,   num5 = 7,    num6 = 1623,
10        num7 = 390,  num8 = 3456, num9 = 12;
11
12    // Display the first row of numbers
13    cout << setw(6) << num1 << setw(6)
14         << num2 << setw(6) << num3 << endl;
15
16    // Display the second row of numbers
17    cout << setw(6) << num4 << setw(6)
18         << num5 << setw(6) << num6 << endl;
19
20    // Display the third row of numbers
21    cout << setw(6) << num7 << setw(6)
22         << num8 << setw(6) << num9 << endl;
23    return 0;
24 }
```

Continued…

Program 3-17 *(continued)*

Program Output

```
2897     5   837
  34     7  1623
 390  3456    12
```

Stream Manipulators

- Some affect values until changed again:
  - `fixed`: use decimal notation for floating-point values
  - `setprecision(x)`: when used with `fixed`, print floating-point value using `x` digits after the decimal. Without `fixed`, print floating-point value using x significant digits
  - `showpoint`: always print decimal for floating-point values

Program 3-21

```
1  // This program asks for sales figures for 3 days. The total
2  // sales are calculated and displayed in a table.
3  #include <iostream>
4  #include <iomanip>
5  using namespace std;
6
7  int main()
8  {
9     double day1, day2, day3, total;
10
11    // Get the sales for each day.
12    cout << "Enter the sales for day 1: ";
13    cin >> day1;
14    cout << "Enter the sales for day 2: ";
15    cin >> day2;
16    cout << "Enter the sales for day 3: ";
17    cin >> day3;
18
```

Continued…

```
19      // Calculate the total sales.
20      total = day1 + day2 + day3;
21
22      // Display the sales figures.
23      cout << "\nSales Figures\n";
24      cout << "-------------\n";
25      cout << setprecision(2) << fixed;
26      cout << "Day 1: " << setw(8) << day1 << endl;
27      cout << "Day 2: " << setw(8) << day2 << endl;
28      cout << "Day 3: " << setw(8) << day3 << endl;
29      cout << "Total: " << setw(8) << total << endl;
30      return 0;
31  }
```

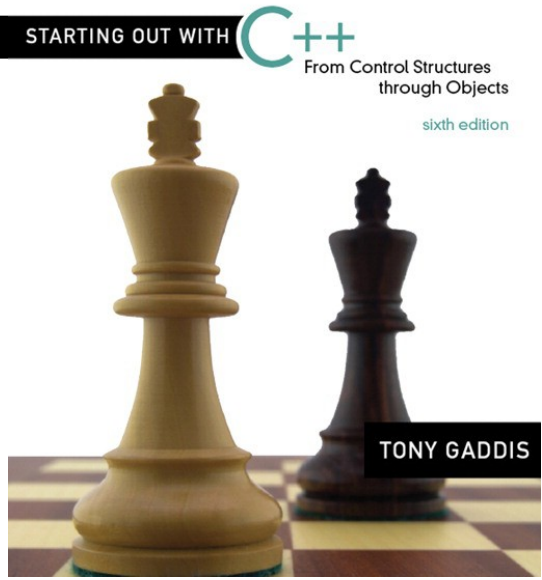
Program Output with Example Input Shown in Bold

```
Enter the sales for day 1:  1321.87 [Enter]
Enter the sales for day 2:  1869.26 [Enter]
Enter the sales for day 3:  1403.77 [Enter]

Sales Figures
-------------
Day 1:    1321.87
Day 2:    1869.26
Day 3:    1403.77
Total:    4594.90
```

Stream Manipulators

Table 3-12

Stream Manipulator	Description
`setw(n)`	Establishes a print field of n spaces.
`fixed`	Displays floating-point numbers in fixed point notation.
`showpoint`	Causes a decimal point and trailing zeroes to be displayed, even if there is no fractional part.
`setprecision(n)`	Sets the precision of floating-point numbers.
`left`	Causes subsequent output to be left justified.
`right`	Causes subsequent output to be right justified.



3.9

Formatted Input

Formatted Input

- Can format field width for use with `cin`
- Useful when reading string data to be stored in a character array:

```
const int SIZE = 10;
char firstName[SIZE];
cout << "Enter your name: ";
cin >> setw(SIZE) >> firstName;
```

- `cin` reads one less character than specified with the `setw()` manipulator

Formatted Input

- To read an entire line of input, use `cin.getline()`:

```
const int SIZE = 81;
char address[SIZE];
cout << "Enter your address: ";
cin.getline(address, SIZE);
```

- `cin.getline` takes two arguments:
  - Name of array to store string
  - Size of the array

Program 3-23

```
1  // This program demonstrates cin's getline member function.
2  #include <iostream>
3  using namespace std;
4
5  int main()
6  {
7     const int SIZE = 81;
8     char sentence[SIZE];
9
10    cout << "Enter a sentence: ";
11    cin.getline(sentence, SIZE);
12    cout << "You entered " << sentence << endl;
13    return 0;
14 }
```

Program Output with Example Input Shown in Bold

```
Enter a sentence: To be, or not to be, that is the question. [Enter]
You entered To be, or not to be, that is the question.
```

Formatted Input

- To read a single character:
  - Use `cin`:

```
char ch;
cout << "Strike any key to continue";
cin >> ch;
```

Problem: will skip over blanks, tabs, <CR>

  - Use `cin.get()`:

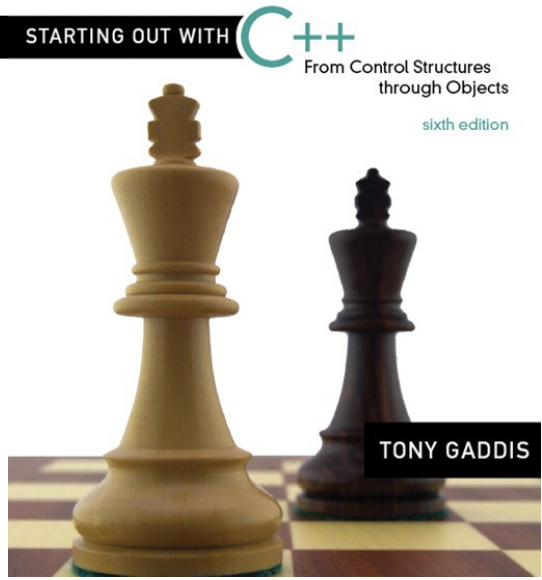
```
cin.get(ch);
```

Will read the next character entered, even whitespace

Formatted Input

- Mixing `cin >>` and `cin.get()` in the same program can cause input errors that are hard to detect
- To skip over unneeded characters that are still in the keyboard buffer, use `cin.ignore()`:

```
cin.ignore(); // skip next char
cin.ignore(10, '\n'); // skip the next
                      // 10 char. or until a '\n'
```

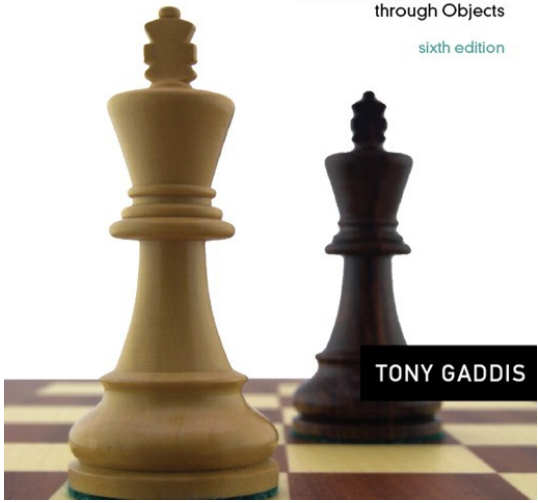


3.10

More About Member Functions

More About Member Functions

- Member Function: procedure that is part of an object
- `cout`, `cin` are objects
- Some member functions of the `cin` object:
  - `getline`
  - `get`
  - `ignore`



3.11

More Mathematical Library Functions

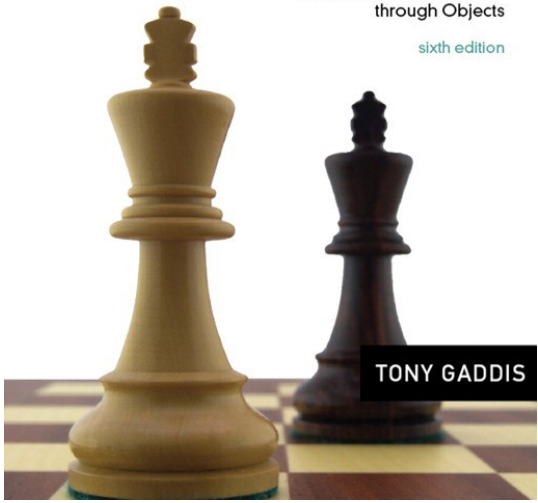
More Mathematical Library Functions

- Require `cmath` header file
- Take `double` as input, return a `double`
- Commonly used functions:

`sin`	Sine
`cos`	Cosine
`tan`	Tangent
`sqrt`	Square root
`log`	Natural (e) log
`abs`	Absolute value (takes and returns an int)

More Mathematical Library Functions

- These require `cstdlib` header file
- `rand()`: returns a random number (`int`) between `0` and the largest int the compute holds. Yields same sequence of numbers each time program is run.
- `srand(x)`: initializes random number generator with `unsigned int x`



3.12

Hand Tracing a Program

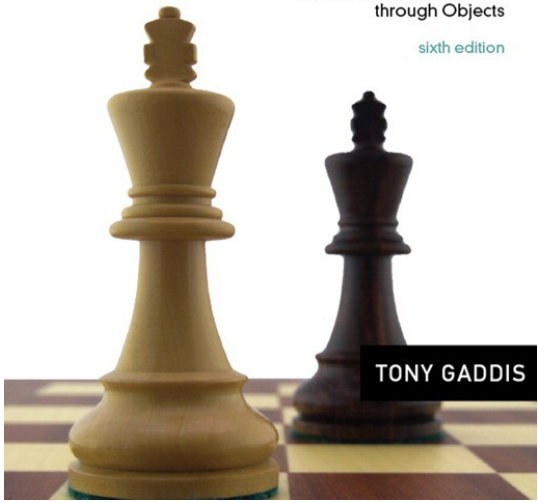
Hand Tracing a Program

- Hand trace a program: act as if you are the computer, executing a program:
  - step through and ‘execute’ each statement, one-by-one
  - record the contents of variables after statement execution, using a hand trace chart (table)
- Useful to locate logic or mathematical errors

Program 3-27 **(with hand trace chart filled)**

```
1  // This program asks for three numbers, then
2  // displays the average of the numbers.
3  #include <iostream>
4  using namespace std;

5  int main()

6  {
7     double num1, num2, num3, avg;

8     cout << "Enter the first number: ";

9     cin >> num1;

10    cout << "Enter the second number: ";

11    cin >> num2;

12    cout << "Enter the third number: ";

13    cin >> num3;

14    avg = num1 + num2 + num3 / 3;

15    cout << "The average is " << avg << endl;

16    return 0;

17 }
```

num1	num2	num3	avg
?	?	?	?
?	?	?	?
10	?	?	?
10	?	?	?
10	20	?	?
10	20	?	?
10	20	30	?
10	20	30	40
10	20	30	40



3.14

Introduction to File Input and Output

Introduction to File Input and Output

- Can use files instead of keyboard, monitor screen for program input, output
- Allows data to be retained between program runs
- Steps:
  - *Open* the file
  - *Use* the file (read from, write to, or both)
  - *Close* the file

Files: What is Needed

- Use `fstream` header file for file access
- File stream types:
  - `ifstream` for input from a file
  - `ofstream` for output to a file
  - `fstream` for input from or output to a file
- Define file stream objects:

```
ifstream infile;
ofstream outfile;
```

Opening Files

- Create a link between file name (outside the program) and file stream object (inside the program)
- Use the `open` member function:

```
infile.open("inventory.dat");
outfile.open("report.txt");
```

- Filename may include drive, path info.
- Output file will be created if necessary; existing file will be erased first
- Input file must exist for `open` to work

Using Files

- Can use output file object and << to send data to a file:

```
outfile << "Inventory report";
```

- Can use input file object and >> to copy data from file to variables:

```
infile >> partNum;
infile >> qtyInStock >>
qtyOnOrder;
```

Closing Files

- Use the `close` member function:

```
infile.close();
outfile.close();
```

- Don't wait for operating system to close files at program end:
  - may be limit on number of open files
  - may be buffered output data waiting to send to file

Program 3-29

```
1  // This program writes data to a file.
2  #include <iostream>
3  #include <fstream>
4  using namespace std;
5
6  int main()
7  {
8     ofstream outputFile;
9     outputFile.open("demofile.txt");
10
11    cout << "Now writing data to the file.\n";
12
13    // Write 4 great names to the file
14    outputFile << "Bach\n";
15    outputFile << "Beethoven\n";
16    outputFile << "Mozart\n";
17    outputFile << "Schubert\n";
18
19    // Close the file
20    outputFile.close();
21    cout << "Done.\n";
22    return 0;
23 }
```

Continued…

Program Screen Output

```
Now writing data to the file.
Done.
```

Output to File `demofile.txt`

```
Bach
Beethoven
Mozart
Schubert
```

Program 3-30

```
1  // This program reads data from a file.
2  #include <iostream>
3  #include <fstream>
4  using namespace std;
5
6  int main()
7  {
8     ifstream inFile;
9     const int SIZE = 81;
10    char name[SIZE];
11
12    inFile.open("demofile.txt");
13    cout << "Reading data from the file.\n\n";
14
15    inFile >> name;          // Read name 1 from the file
16    cout << name << endl;    // Display name 1
17
```

Continued…

Program 3-30 *(continued)*

```
18      inFile >> name;          // Read name 2 from the file
19      cout << name << endl;    // Display name 2
20
21      inFile >> name;          // Read name 3 from the file
22      cout << name << endl;    // Display name 3
23
24      inFile >> name;          // Read name 4 from the file
25      cout << name << endl;    // Display name 4
26
27      inFile.close();          // Close the file
28      cout << "\nDone.\n";
29      return 0;
30  }
```

Program Screen Output

```
Reading data from the file.

Bach
Beethoven
Mozart
Schubert

Done.
```