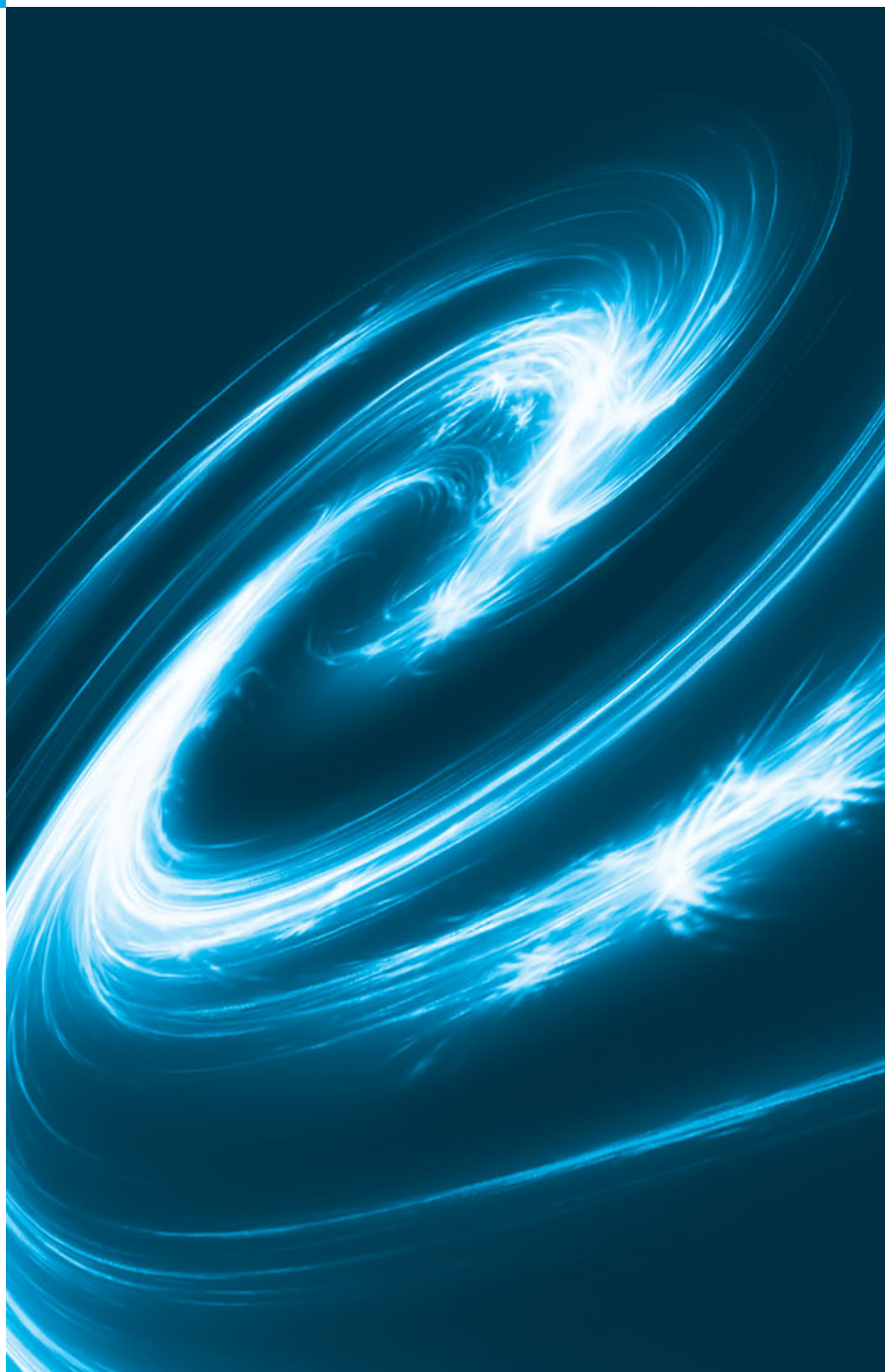
2

Introduction to C Programming



What's in a name?
That which we call a rose
By any other name would
smell as sweet.
—William Shakespeare

"Take some more tea," the March Hare said to Alice, very earnestly. "I've had nothing yet," Alice replied in an offended tone: "so I can't take more." "You mean you can't take less," said the Hatter: "it's very easy to take more than nothing."
—Lewis Carroll

High thoughts must have high language.
—Aristophanes

Objectives

In this chapter, you'll:

- Write simple computer programs in C.
- Use simple input and output statements.
- Use the fundamental data types.
- Learn computer memory concepts.
- Use arithmetic operators.
- Learn the precedence of arithmetic operators.
- Write simple decision-making statements.

condition 54
decision 43
definition 46
destructive 49
document a program 41
embedded parentheses 51
Enter key 47
equality operator 54
escape character 43
escape sequence 43
executable 44
false 54
flow of control 57
format control string 47
function 42
identifier 46
`if` statement 54
`int` type 46
integer 46
integer division 50
interactive computing 48
keyword 57
literal 43
message 43
nested parentheses 51
newline (`\n`) 43
nondestructive 50
operand 48
percent sign (%) 50
`printf` function 43
prompt 47
`puts` function 58
redundant parentheses 54
relational operator 54
right brace (}) 42
rules of operator precedence 51
`scanf` function 47
single-line comment (`//`) 41
statement 43
statement terminator (;) 43
`<stdio.h>` (standard input/output) header 42
straight-line form 51
string 43
true 54
type 49
value 49
variable 46
white space 42

Self-Review Exercises

2.1 Fill in the blanks in each of the following.

a) Every C program begins execution at the function _____.
b) Every function's body begins with _____ and ends with _____.
c) Every statement ends with a(n) _____.
d) The _____ standard library function displays information on the screen.
e) The escape sequence `\n` represents the _____ character, which causes the cursor to position to the beginning of the next line on the screen.
f) The _____ Standard Library function is used to obtain data from the keyboard.
g) The conversion specifier _____ is used in a `scanf` format control string to indicate that an integer will be input and in a `printf` format control string to indicate that an integer will be output.
h) Whenever a new value is placed in a memory location, that value overrides the previous value in that location. This process is said to be _____.
i) When a value is read from a memory location, the value in that location is preserved; this process is said to be _____.
j) The _____ statement is used to make decisions.

2.2 State whether each of the following is *true* or *false*. If *false*, explain why.

a) Function `printf` always begins printing at the beginning of a new line.
b) Comments cause the computer to display the text after `//` on the screen when the program is executed.
c) The escape sequence `\n` when used in a `printf` format control string causes the cursor to position to the beginning of the next line on the screen.
d) All variables must be defined before they're used.
e) All variables must be given a type when they're defined.

f) C considers the variables `number` and `NuMbEr` to be identical.
g) Definitions can appear anywhere in the body of a function.
h) All arguments following the format control string in a `printf` function must be preceded by an ampersand (`&`).
i) The remainder operator (`%`) can be used only with integer operands.
j) The arithmetic operators `*`, `/`, `%`, `+` and `-` all have the same level of precedence.
k) A program that prints three lines of output must contain three `printf` statements.

2.3 Write a single C statement to accomplish each of the following:
a) Define the variables `c`, `thisVariable`, `q76354` and `number` to be of type `int`.
b) Prompt the user to enter an integer. End your prompting message with a colon (`:`) followed by a space and leave the cursor positioned after the space.
c) Read an integer from the keyboard and store the value entered in integer variable `a`.
d) If `number` is not equal to `7`, print `"The variable number is not equal to 7."`
e) Print the message `"This is a C program."` on one line.
f) Print the message `"This is a C program."` on two lines so that the first line ends with `C`.
g) Print the message `"This is a C program."` with each word on a separate line.
h) Print the message `"This is a C program."` with the words separated by tabs.

2.4 Write a statement (or comment) to accomplish each of the following:
a) State that a program will calculate the product of three integers.
b) Define the variables `x`, `y`, `z` and `result` to be of type `int`.
c) Prompt the user to enter three integers.
d) Read three integers from the keyboard and store them in the variables `x`, `y` and `z`.
e) Compute the product of the three integers contained in variables `x`, `y` and `z`, and assign the result to the variable `result`.
f) Print `"The product is"` followed by the value of the integer variable `result`.

2.5 Using the statements you wrote in Exercise 2.4, write a complete program that calculates the product of three integers.

2.6 Identify and correct the errors in each of the following statements:
a)
```
printf( "The value is %d\n", &number );
```
b)
```
scanf( "%d%d", &number1, number2 );
```
c)
```
if ( c < 7 );{
   printf( "C is less than 7\n" );
}
```
d)
```
if ( c => 7 ) {
   printf( "C is greater than or equal to 7\n" );
}
```

Answers to Self-Review Exercises

2.1 a) `main`. b) left brace (`{`), right brace (`}`). c) semicolon. d) `printf`. e) newline. f) `scanf`. g) `%d`. h) destructive. i) nondestructive. j) `if`.

2.2 a) False. Function `printf` always begins printing where the cursor is positioned, and this may be anywhere on a line of the screen.
b) False. Comments do not cause any action to be performed when the program is executed. They're used to document programs and improve their readability.
c) True.
d) True.
e) True.
f) False. C is case sensitive, so these variables are unique.

g) False. A variable's definition must appear before its first use in the code. In Microsoft Visual C++, variable definitions must appear immediately following the left brace that begins the body of `main`. Later in the book we'll discuss this in more depth as we encounter additional C features that can affect this issue.

h) False. Arguments in a `printf` function ordinarily should not be preceded by an ampersand. Arguments following the format control string in a `scanf` function ordinarily should be preceded by an ampersand. We'll discuss exceptions to these rules in Chapter 6 and Chapter 7.

i) True.

j) False. The operators `*`, `/` and `%` are on the same level of precedence, and the operators `+` and `-` are on a lower level of precedence.

k) False. A `printf` statement with multiple `\n` escape sequences can print several lines.

2.3 a)
```
int c, thisVariable, q76354, number;
```
b)
```
printf( "Enter an integer: " );
```
c)
```
scanf( "%d", &a );
```
d)
```
if ( number != 7 ) {
   printf( "The variable number is not equal to 7.\n" );
}
```
e)
```
printf( "This is a C program.\n" );
```
f)
```
printf( "This is a C\nprogram.\n" );
```
g)
```
printf( "This\nis\na\nC\nprogram.\n" );
```
h)
```
printf( "This\tis\ta\tC\tprogram.\n" );
```

2.4 a)
```
// Calculate the product of three integers
```
b)
```
int x, y, z, result;
```
c)
```
printf( "Enter three integers: " );
```
d)
```
scanf( "%d%d%d", &x, &y, &z );
```
e)
```
result = x * y * z;
```
f)
```
printf( "The product is %d\n", result );
```

2.5 See below.

```
1  // Calculate the product of three integers
2  #include <stdio.h>
3
4  int main( void )
5  {
6     int x, y, z, result; // declare variables
7
8     printf( "Enter three integers: " ); // prompt
9     scanf( "%d%d%d", &x, &y, &z ); // read three integers
10    result = x * y * z; // multiply values
11    printf( "The product is %d\n", result ); // display result
12 } // end function main
```

2.6 a) Error: `&number`.
Correction: Eliminate the `&`. We discuss exceptions to this later.

b) Error: `number2` does not have an ampersand.
Correction: `number2` should be `&number2`. Later in the text we discuss exceptions to this.

c) Error: Semicolon after the right parenthesis of the condition in the `if` statement.
Correction: Remove the semicolon after the right parenthesis. [*Note:* The result of this error is that the `printf` statement will be executed whether or not the condition in the

`if` statement is true. The semicolon after the right parenthesis is considered an empty statement—a statement that does nothing.]

d) Error: => is not an operator in C.
Correction: The relational operator => should be changed to >= (greater than or equal to).

Exercises

2.7 Identify and correct the errors in each of the following statements. (*Note:* There may be more than one error per statement.)

a)
```
scanf( "d", value );
```
b)
```
printf( "The product of %d and %d is %d"\n, x, y );
```
c)
```
firstNumber + secondNumber = sumOfNumbers
```
d)
```
if ( number => largest )
   largest == number;
```
e)
```
*/ Program to determine the largest of three integers /*
```
f)
```
Scanf( "%d", anInteger );
```
g)
```
printf( "Remainder of %d divided by %d is\n", x, y, x % y );
```
h)
```
if ( x = y );
   printf( %d is equal to %d\n", x, y );
```
i)
```
print( "The sum is %d\n," x + y );
```
j)
```
Printf( "The value you entered is: %d\n, &value );
```

2.8 Fill in the blanks in each of the following:

a) _____ are used to document a program and improve its readability.
b) The function used to display information on the screen is _____.
c) A C statement that makes a decision is _____.
d) Calculations are normally performed by _____ statements.
e) The _____ function inputs values from the keyboard.

2.9 Write a single C statement or line that accomplishes each of the following:

a) Print the message "`Enter two numbers.`"
b) Assign the product of variables `b` and `c` to variable `a`.
c) State that a program performs a sample payroll calculation (i.e., use text that helps to document a program).
d) Input three integer values from the keyboard and place them in integer variables `a`, `b` and `c`.

2.10 State which of the following are *true* and which are *false*. If *false*, explain your answer.

a) C operators are evaluated from left to right.
b) The following are all valid variable names: `_under_bar_`, `m928134`, `t5`, `j7`, `her_sales`, `his_account_total`, `a`, `b`, `c`, `z`, `z2`.
c) The statement `printf("a = 5;");` is a typical example of an assignment statement.
d) A valid arithmetic expression containing no parentheses is evaluated from left to right.
e) The following are all invalid variable names: `3g`, `87`, `67h2`, `h22`, `2h`.

2.11 Fill in the blanks in each of the following:

a) What arithmetic operations are on the same level of precedence as multiplication? _____.
b) When parentheses are nested, which set of parentheses is evaluated first in an arithmetic expression? _____.
c) A location in the computer's memory that may contain different values at various times throughout the execution of a program is called a _____.

2.12 What, if anything, prints when each of the following statements is performed? If nothing prints, then answer "Nothing." Assume `x = 2` and `y = 3`.

a) `printf( "%d", x );`
b) `printf( "%d", x + x );`
c) `printf( "x=" );`
d) `printf( "x=%d", x );`
e) `printf( "%d = %d", x + y, y + x );`
f) `z = x + y;`
g) `scanf( "%d%d", &x, &y );`
h) `// printf( "x + y = %d", x + y );`
i) `printf( "\n" );`

2.13 Which, if any, of the following C statements contain variables whose values are replaced?

a) `scanf( "%d%d%d%d%d", &b, &c, &d, &e, &f );`
b) `p = i + j + k + 7;`
c) `printf( "Values are replaced" );`
d) `printf( "a = 5" );`

2.14 Given the equation $y = ax^3 + 7$, which of the following, if any, are correct C statements for this equation?

a) `y = a * x * x * x + 7;`
b) `y = a * x * x * ( x + 7 );`
c) `y = ( a * x ) * x * ( x + 7 );`
d) `y = ( a * x ) * x * x + 7;`
e) `y = a * ( x * x * x ) + 7;`
f) `y = a * x * ( x * x + 7 );`

2.15 State the order of evaluation of the operators in each of the following C statements and show the value of `x` after each statement is performed.

a) `x = 7 + 3 * 6 / 2 - 1;`
b) `x = 2 % 2 + 2 * 2 - 2 / 2;`
c) `x = ( 3 * 9 * ( 3 + ( 9 * 3 / ( 3 ) ) ) );`

2.16 *(Arithmetic)* Write a program that asks the user to enter two numbers, obtains them from the user and prints their sum, product, difference, quotient and remainder.

2.17 *(Printing Values with `printf`)* Write a program that prints the numbers 1 to 4 on the same line. Write the program using the following methods.

a) Using one `printf` statement with no conversion specifiers.
b) Using one `printf` statement with four conversion specifiers.
c) Using four `printf` statements.

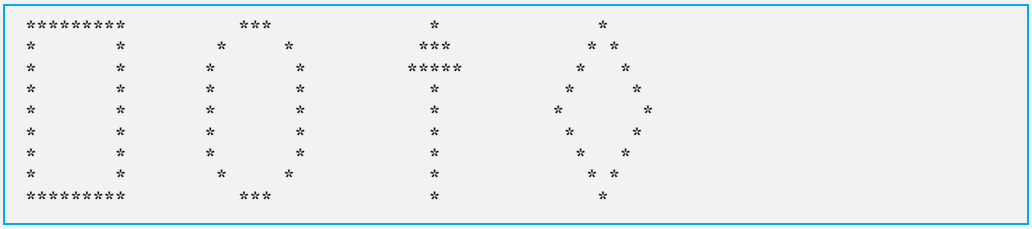
2.18 *(Comparing Integers)* Write a program that asks the user to enter two integers, obtains the numbers from the user, then prints the larger number followed by the words "`is larger`." If the numbers are equal, print the message "`These numbers are equal`." Use only the single-selection form of the `if` statement you learned in this chapter.

2.19 *(Arithmetic, Largest Value and Smallest Value)* Write a program that inputs three different integers from the keyboard, then prints the sum, the average, the product, the smallest and the largest of these numbers. Use only the single-selection form of the `if` statement you learned in this chapter. The screen dialogue should appear as follows:

```
Enter three different integers: 13 27 14
Sum is 54
Average is 18
Product is 4914
Smallest is 13
Largest is 27
```

2.20 ***(Diameter, Circumference and Area of a Circle)*** Write a program that reads in the radius of a circle and prints the circle's diameter, circumference and area. Use the constant value 3.14159 for π. Perform each of these calculations inside the `printf` statement(s) and use the conversion specifier `%f`. [*Note:* In this chapter, we've discussed only integer constants and variables. In Chapter 3 we'll discuss floating-point numbers, i.e., values that can have decimal points.]

2.21 ***(Shapes with Asterisks)*** Write a program that prints the following shapes with asterisks.



```
*********          ***              *              *
*       *        *     *           ***            * *
*       *       *       *         *****          *   *
*       *       *       *           *           *     *
*       *       *       *           *          *       *
*       *       *       *           *           *     *
*       *       *       *           *            *   *
*       *        *     *            *             * *
*********          ***              *              *
```

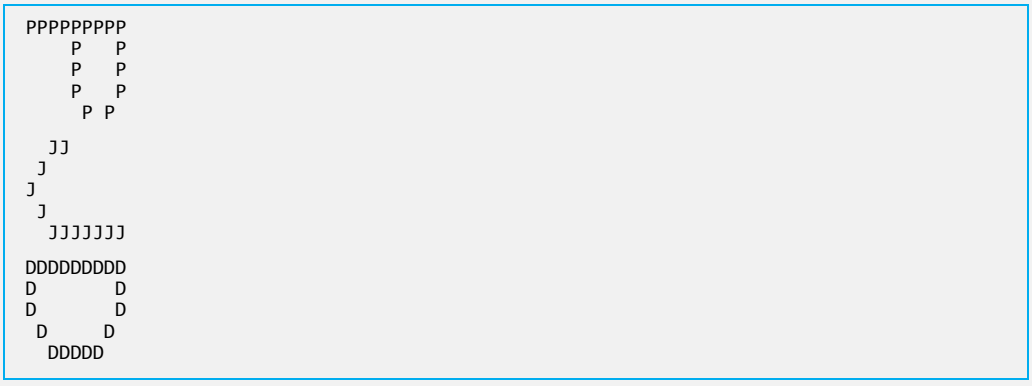
2.22 What does the following code print?

```
printf( "*\n**\n***\n****\n*****\n" );
```

2.23 ***(Largest and Smallest Integers)*** Write a program that reads in three integers and then determines and prints the largest and the smallest integers in the group. Use only the programming techniques you have learned in this chapter.

2.24 ***(Odd or Even)*** Write a program that reads an integer and determines and prints whether it's odd or even. [*Hint:* Use the remainder operator. An even number is a multiple of two. Any multiple of two leaves a remainder of zero when divided by 2.]

2.25 Print your initials in block letters down the page. Construct each block letter out of the letter it represents, as shown below.



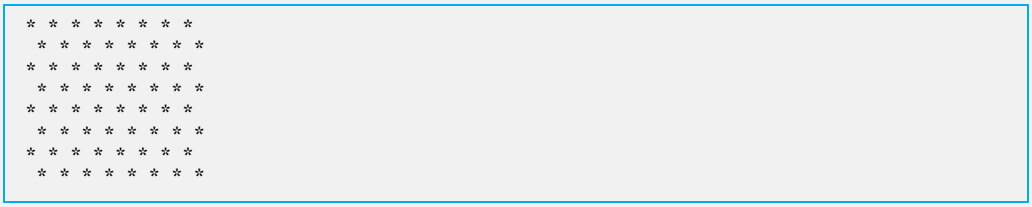
```
PPPPPPPPP
    P   P
    P   P
    P   P
     P P

  JJ
 J
J
 J
  JJJJJJJ

DDDDDDDDD
D        D
D        D
 D      D
  DDDDD
```

2.26 ***(Multiples)*** Write a program that reads in two integers and determines and prints whether the first is a multiple of the second. [*Hint:* Use the remainder operator.]

2.27 ***(Checkerboard Pattern of Asterisks)*** Display the following checkerboard pattern with eight `printf` statements and then display the same pattern with as few `printf` statements as possible.



```
* * * * * * * *
 * * * * * * * *
* * * * * * * *
 * * * * * * * *
* * * * * * * *
 * * * * * * * *
* * * * * * * *
 * * * * * * * *
```

2.28 Distinguish between the terms fatal error and nonfatal error. Why might you prefer to experience a fatal error rather than a nonfatal error?

2.29 ***(Integer Value of a Character)*** Here's a peek ahead. In this chapter you learned about integers and the type `int`. C can also represent uppercase letters, lowercase letters and a considerable variety of special symbols. C uses small integers internally to represent each different character. The set of characters a computer uses together with the corresponding integer representations for those characters is called that computer's character set. You can print the integer equivalent of uppercase `A`, for example, by executing the statement

```
printf( "%d", 'A' );
```

Write a C program that prints the integer equivalents of some uppercase letters, lowercase letters, digits and special symbols. As a minimum, determine the integer equivalents of the following: `A B C a b c 0 1 2 $ * + /` and the blank character.

2.30 ***(Separating Digits in an Integer)*** Write a program that inputs one five-digit number, separates the number into its individual digits and prints the digits separated from one another by three spaces each. [*Hint:* Use combinations of integer division and the remainder operation.] For example, if the user types in `42139`, the program should print

```
4   2   1   3   9
```

2.31 ***(Table of Squares and Cubes)*** Using only the techniques you learned in this chapter, write a program that calculates the squares and cubes of the numbers from 0 to 10 and uses tabs to print the following table of values:

```
number  square  cube
0       0       0
1       1       1
2       4       8
3       9       27
4       16      64
5       25      125
6       36      216
7       49      343
8       64      512
9       81      729
10      100     1000
```

Making a Difference

2.32 ***(Body Mass Index Calculator)*** We introduced the body mass index (BMI) calculator in Exercise 1.14. The formulas for calculating BMI are

$$BMI = \frac{weightInPounds \times 703}{heightInInches \times heightInInches}$$

or

$$BMI = \frac{weightInKilograms}{heightInMeters \times heightInMeters}$$

Create a BMI calculator application that reads the user's weight in pounds and height in inches (or, if you prefer, the user's weight in kilograms and height in meters), then calculates and displays the user's body mass index. Also, the application should display the following information from the Department of Health and Human Services/National Institutes of Health so the user can evaluate his/her BMI:

```
BMI VALUES
Underweight: less than 18.5
Normal:      between 18.5 and 24.9
Overweight:  between 25 and 29.9
Obese:       30 or greater
```

[*Note:* In this chapter, you learned to use the `int` type to represent whole numbers. The BMI calculations when done with `int` values will both produce whole-number results. In Chapter 4 you'll learn to use the `double` type to represent numbers with decimal points. When the BMI calculations are performed with `double`s, they'll both produce numbers with decimal points—these are called "floating-point" numbers.]

2.33 ***(Car-Pool Savings Calculator)*** Research several car-pooling websites. Create an application that calculates your daily driving cost, so that you can estimate how much money could be saved by car pooling, which also has other advantages such as reducing carbon emissions and reducing traffic congestion. The application should input the following information and display the user's cost per day of driving to work:

a) Total miles driven per day.
b) Cost per gallon of gasoline.
c) Average miles per gallon.
d) Parking fees per day.
e) Tolls per day.