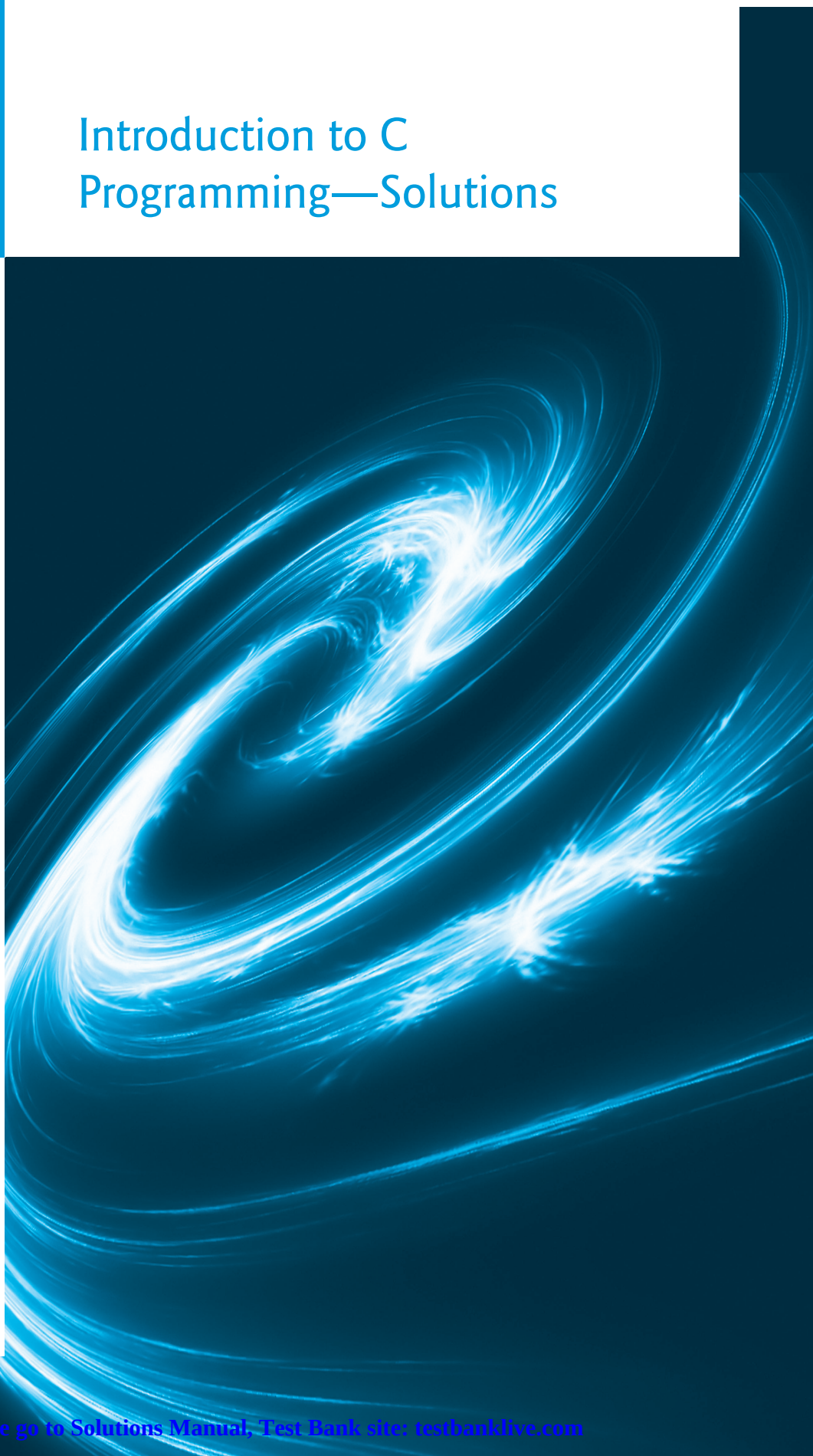
2 Introduction to C Programming—Solutions



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

When faced with a decision, I always ask, "What would be the most fun?"
—Peggy Walker

"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.

Self-Review Exercises

2.1 Fill in the blanks in each of the following.

a) Every C program begins execution at the function _____.

ANS: `main`.

b) Every function's body begins with _____ and ends with _____.

ANS: left brace, right brace.

c) Every statement ends with a(n) _____.

ANS: semicolon.

d) The _____ standard library function displays information on the screen.

ANS: `printf`.

e) The escape sequence `\n` represents the _____ character, which causes the cursor to position to the beginning of the next line on the screen.

ANS: newline.

f) The _____ Standard Library function is used to obtain data from the keyboard.

ANS: `scanf`.

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.

ANS: `%d`.

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 _____.

ANS: destructive.

i) When a value is read from a memory location, the value in that location is preserved; this process is said to be _____.

ANS: nondestructive.

j) The _____ statement is used to make decisions.

ANS: `if`.

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.

ANS: False. Function `printf` always begins printing where the cursor is positioned, and this may be anywhere on a line of the screen.

b) Comments cause the computer to print the text after the `//` on the screen when the program is executed.

ANS: 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) 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.

ANS: True.

d) All variables must be defined before they're used.

ANS: True.

e) All variables must be given a type when they're defined.

ANS: True.

f) C considers the variables `number` and `NuMbEr` to be identical.

ANS: False. C is case sensitive, so these variables are different.

g) Definitions can appear anywhere in the body of a function.

ANS: 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) All arguments following the format control string in a `printf` function must be preceded by an ampersand (`&`).

ANS: 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 will discuss exceptions to these rules in Chapter 6 and Chapter 7.

i) The remainder operator (`%`) can be used only with integer operands.

ANS: True.

j) The arithmetic operators `*`, `/`, `%`, `+` and `-` all have the same level of precedence.

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

k) A program that prints three lines of output must contain three `printf` statements.

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

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`.

ANS:
```
int c, thisVariable, q76354, number;
```

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.

ANS:
```
printf( "Enter an integer: " );
```

c) Read an integer from the keyboard and store the value entered in integer variable `a`.

ANS:
```
scanf( "%d", &a );
```

d) If `number` is not equal to `7`, print `"The variable number is not equal to 7."`

ANS:
```
if ( number != 7 ) {
   printf( "The variable number is not equal to 7.\n" );
}
```

e) Print the message `"This is a C program."` on one line.

ANS:
```
printf( "This is a C program.\n" );
```

f) Print the message `"This is a C program."` on two lines so that the first line ends with `C`.

ANS:
```
printf( "This is a C\nprogram.\n" );
```

g) Print the message `"This is a C program."` with each word on a separate line.

ANS:
```
printf( "This\nis\na\nC\nprogram.\n" );
```

h) Print the message `"This is a C program."` with the words separated by tabs.

ANS:
```
printf( "This\tis\ta\tC\tprogram.\n" );
```

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.

ANS:
```
// Calculate the product of three integers
```

b) Define the variables `x`, `y`, `z` and `result` to be of type `int`.

ANS:
```
int x, y, z, result;
```

c) Prompt the user to enter three integers.

ANS:
```
printf( "Enter three integers: " );
```

d) Read three integers from the keyboard and store them in the variables `x`, `y` and `z`.

ANS:
```
scanf( "%d%d%d", &x, &y, &z );
```

e) Compute the product of the three integers contained in variables `x`, `y` and `z`, and assign the result to the variable `result`.

ANS:
```
result = x * y * z;
```

f) Print `"The product is"` followed by the value of the integer variable `result`.

ANS:
```
printf( "The product is %d\n", result );
```

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

ANS:

```
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 Identify and correct the errors in each of the following statements:

a) `printf( "The value is %d\n", &number );`

ANS: Error: `&number`. Correction: Eliminate the `&`. We discuss exceptions to this later.

b) `scanf( "%d%d", &number1, number2 );`

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

c)
```
if ( c < 7 ); {
   printf( "C is less than 7\n" );
}
```

ANS: 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)
```
if ( c => 7 ) {
   printf( "C is greater than or equal to 7\n" );
}
```

ANS: Error: 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 );`

ANS: `scanf( "%d", &value );`

b) `printf( "The product of %d and %d is %d"\n, x, y );`

ANS: `printf( "The product of %d and %d is %d\n", x, y, x * y );`

c) `firstNumber + secondNumber = sumOfNumbers`

ANS: `sumOfNumbers = firstNumber + secondNumber;`

d)
```
if ( number => largest )
   largest == number;
```

ANS:
```
if ( number >= largest )
   largest = number;
```

e) `*/ Program to determine the largest of three integers /*`

ANS: `/* Program to determine the largest of three integers */`

f) `Scanf( "%d", anInteger );`
ANS: `scanf( "%d", &anInteger );`
g) `printf( "Remainder of %d divided by %d is\n", x, y, x % y );`
ANS: `printf( "Remainder of %d divided by %d is %d\n", x, y, x % y );`
h)
```
if ( x = y );
   printf( %d is equal to %d\n", x, y );
```
ANS:
```
if ( x == y )
   printf( "%d is equal to %d\n", x, y );
```
i) `print( "The sum is %d\n," x + y );`
ANS: `printf( "The sum is %d\n", x + y );`
j) `Printf( "The value you entered is: %d\n, &value );`
ANS: `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.
ANS: comments.
b) The function used to display information on the screen is _____.
ANS: `printf`.
c) A C statement that makes a decision is _____.
ANS: `if`.
d) Calculations are normally performed by _____ statements.
ANS: assignment.
e) The _____ function inputs values from the keyboard.
ANS: `scanf`.

2.9 Write a single C statement or line that accomplishes each of the following:
a) Print the message "`Enter two numbers.`"
ANS: `printf( "Enter two numbers\n" );`
b) Assign the product of variables `b` and `c` to variable `a`.
ANS: `a = b * c;`
c) State that a program performs a sample payroll calculation (i.e., use text that helps to document a program).
ANS: `// Sample payroll calculation program`
d) Input three integer values from the keyboard and place these values in integer variables `a`, `b` and `c`.
ANS: `scanf( "%d%d%d", &a, &b, &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.
ANS: False. Some operators are evaluated left to right and others are evaluated from right to left depending on their associativity (see Appendix A).
b) The following are all valid variable names: `_under_bar_`, `m928134`, `t5`, `j7`, `her_sales`, `his_account_total`, `a`, `b`, `c`, `z`, `z2`.
ANS: True.
c) The statement `printf("a = 5;");` is a typical example of an assignment statement.
ANS: False. The statement prints `a = 5;` on the screen.
d) A valid arithmetic expression containing no parentheses is evaluated from left to right.
ANS: False. Multiplication, division, and modulus are all evaluated first from left to right, then addition and subtraction are evaluated from left to right.
e) The following are all invalid variable names: `3g`, `87`, `67h2`, `h22`, `2h`.
ANS: False. Only those beginning with a number are invalid.

2.11 Fill in the blanks in each of the following:

a) What arithmetic operations are on the same level of precedence as multiplication? _____.

ANS: division, modulus.

b) When parentheses are nested, which set of parentheses is evaluated first in an arithmetic expression? _____.

ANS: The innermost pair of parentheses.

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 _____.

ANS: variable.

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 );`

ANS: `2`

b) `printf( "%d", x + x );`

ANS: `4`

c) `printf( "x=" );`

ANS: `x=`

d) `printf( "x=%d", x );`

ANS: `x=2`

e) `printf( "%d = %d", x + y, y + x );`

ANS: `5 = 5`

f) `z = x + y;`

ANS: Nothing. Value of `x + y` is assigned to `z`.

g) `scanf( "%d%d", &x, &y );`

ANS: Nothing. Two integer values are read into the location of `x` and the location of `y`.

h) `// printf( "x + y = %d", x + y );`

ANS: Nothing. This is a comment.

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

ANS: A newline character is printed, and the cursor is positioned at the beginning of the next line on the screen.

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( "%s", Values are replaced." );`

d) `printf( "a = 5" );`

ANS: a and b.

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 );`

ANS: a, d, and e.

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;`

ANS: `*` is first, `/` is second, `+` is third, `-` is fourth and `=` is last. Value of `x` is `15`.

b) `x = 2 % 2 + 2 * 2 - 2 / 2;`

ANS: `%` is first, `*` is second, `/` is third, `+` is fourth, `-` is fifth and `=` is last. Value of `x` is `3`.

```
c) x = ( 3 * 9 * ( 3 + ( 9 * 3 / ( 3 ) ) ) );
ANS:       5   6     4     2   3   1.  The = evaluates last. Value of x is 324.
```

2.16 Write a program that asks the user to enter two numbers, obtains the two numbers from the user and prints the sum, product, difference, quotient and remainder of the two numbers.

ANS:

```
1  // Exercise 2.16 Solution
2  #include <stdio.h>
3
4  int main( void )
5  {
6     int x; // define first number
7     int y; // define second number
8
9     printf( "%d", "Enter two numbers: " ); // prompt user
10    scanf( "%d%d", &x, &y ); // read values from keyboard
11
12    // output results
13    printf( "The sum is %d\n", x + y );
14    printf( "The product is %d\n", x * y );
15    printf( "The difference is %d\n", x - y );
16    printf( "The quotient is %d\n", x / y );
17    printf( "The remainder is %d\n", x % y );
18 } // end main
```

```
Enter two numbers: 20 5
The sum is 25
The product is 100
The difference is 15
The quotient is 4
The remainder is 0
```

2.17 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.

ANS:

```
1  // Exercise 2.17 Solution
2  #include <stdio.h>
3
4  int main( void )
5  {
6     printf( "1 2 3 4\n\n" ); // part a
7
8     printf( "%d %d %d %d\n\n", 1, 2, 3, 4 ); // part b
9
10    printf( "1 " ); // part c
11    printf( "2 " );
12    printf( "3 " );
13    printf( "4\n" );
```

```
14  } // end main
```

```
1 2 3 4

1 2 3 4

1 2 3 4
```

2.18 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.

ANS:

```
1  // Exercise 2.18 Solution
2  #include <stdio.h>
3
4  int main( void )
5  {
6     int x; // define first number
7     int y; // define second number
8
9     printf( "%s", "Enter two numbers: " ); // prompt
10    scanf( "%d%d", &x, &y ); // read two integers
11
12    // compare the two numbers
13    if ( x > y ) {
14       printf( "%d is larger\n", x );
15    } // end if
16
17    if ( x < y ) {
18       printf( "%d is larger\n", y );
19    } // end if
20
21    if ( x == y ) {
22       puts( "These numbers are equal" );
23    } // end if
24 } // end main
```

```
Enter two numbers: 5 20
20 is larger
```

```
Enter two numbers: 239 92
239 is larger
```

```
Enter two numbers: 17 17
These numbers are equal
```

2.19 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 dialog should appear as follows:

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

ANS:

```c
// Exercise 2.19 Solution
#include <stdio.h>

int main( void )
{
   int a; // define first integer
   int b; // define second integer
   int c; // define third integer
   int smallest; // smallest integer
   int largest; // largest integer

   printf( "%s", "Input three different integers: " ); // prompt user
   scanf( "%d%d%d", &a, &b, &c ); // read three integers

   // output sum, average and product of the three integers
   printf( "Sum is %d\n", a + b + c );
   printf( "Average is %d\n", ( a + b + c ) / 3 );
   printf( "Product is %d\n", a * b * c );

   smallest = a; // assume first number is the smallest

   if ( b < smallest ) { // is b smaller?
      smallest = b;
   } // end if

   if ( c < smallest ) { // is c smaller?
      smallest = c;
   } // end if

   printf( "Smallest is %d\n", smallest );

   largest = a; // assume first number is the largest

   if ( b > largest ) { // is b larger?
      largest = b;
   } // end if

   if ( c > largest ) { // is c larger?
      largest = c;
   } // end if

   printf( "Largest is %d\n", largest );
```

```
43 } // end main
```

2.20 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 have discussed only integer constants and variables. In Chapter 3 we will discuss floating-point numbers, i.e., values that can have decimal points.]

ANS:

```
1  // Exercise 2.20 Solution
2  #include <stdio.h>
3
4  int main( void )
5  {
6     int radius; // circle radius
7
8     printf( "%s", "Input the circle radius: " ); // prompt user
9     scanf( "%d", &radius ); // read integer radius
10
11    // calculate and output diameter, circumference and area
12    printf( "\nThe diameter is %d\n", 2 * radius );
13    printf( "The circumference is %f\n", 2 * 3.14159 * radius );
14    printf( "The area is %f\n", 3.14159 * radius * radius );
15 } // end main
```

```
Input the circle radius: 9

The diameter is 18
The circumference is 56.548620
The area is 254.468790
```

2.21 Write a program that prints a box, an oval, an arrow and a diamond as follows:

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

ANS:

```
1  // Exercise 2.21 Solution
2  #include <stdio.h>
3
4  int main( void )
5  {
6     printf( "%s", "*********        ***         *           *\n" );
7     printf( "%s", "*       *      *     *      ***        * *\n" );
```

```
   printf( "%s", "*         *    *        *   *****       *    *\n" );
   printf( "%s", "*         *    *        *    *         *     *\n" );
   printf( "%s", "*         *    *        *    *        *       *\n" );
   printf( "%s", "*         *    *        *    *         *     *\n" );
   printf( "%s", "*         *    *        *    *          *    *\n" );
   printf( "%s", "*         *     *     *     *           * *\n" );
   printf( "%s", "*********       ***         *            *\n" );
} // end main
```

2.22 What does the following code print?

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

ANS:

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

2.23 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.

ANS:

```
// Exercise 2.23 Solution
#include <stdio.h>

int main( void )
{
   int largest; // largest integer
   int smallest; // smallest integer
   int int1; // define int1 for user input
   int int2; // define int2 for user input
   int int3; // define int3 for user input
   int temp; // temporary integer for swapping

   printf( "%s", "Input 3 integers: " ); // prompt user and read 3 ints
   scanf( "%d%d%d%d%d", &largest, &smallest, &int1, &int2, &int3 );

   if ( smallest > largest ) { // make comparisons
      temp = largest;
      largest = smallest;
      smallest = temp;
   } // end if

   if ( int1 > largest ) {
      largest = int1;
   } // end if

   if ( int1 < smallest ) {
      smallest = int1;
   } // end if
```

```
29
30    if ( int2 > largest ) {
31       largest = int2;
32    } // end if
33
34    if ( int2 < smallest ) {
35       smallest = int2;
36    } // end if
37
38    if ( int3 > largest ) {
39       largest = int3;
40    } // end if
41
42    if ( int3 < smallest ) {
43       smallest = int3;
44    } // end if
45
46    printf( "The largest value is %d\n", largest );
47    printf( "The smallest value is %d\n", smallest );
48 } // end main
```

```
Input 5 integers: 9 4 5 8 7
The largest value is 9
The smallest value is 4
```

2.24 Write a program that reads an integer and determines and prints whether it is 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.]

ANS:

```
1  // Exercise 2.24 Solution
2  #include <stdio.h>
3
4  int main( void )
5  {
6     int integer; // integer input by user
7
8     printf( "%s", "Input an integer: " ); // prompt
9     scanf( "%d", &integer ); // read integer
10
11    // test if integer is even
12    if ( integer % 2 == 0 ) {
13       printf( "%d is an even integer\n", integer );
14    } // end if
15
16    // test if integer is odd
17    if ( integer % 2 != 0 ) {
18       printf( "%d is an odd integer\n", integer );
19    } // end if
20 } // end main
```

```
Input an integer: 78
78 is an even integer
```

```
Input an integer: 79
79 is an odd integer
```

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
```

ANS:

```
1  // Exercise 2.25 Solution
2  #include <stdio.h>
3
4  int main( void )
5  {
6     puts( "PPPPPPPPP" );
7     puts( "    P   P" );
8     puts( "    P   P" );
9     puts( "    P   P" );
10    puts( "     P P\n" );
11    puts( "  JJ" );
12    puts( " J" );
13    puts( "J" );
14    puts( " J" );
15    puts( "  JJJJJJJ\n" );
16    puts( "DDDDDDDDD" );
17    puts( "D        D" );
18    puts( "D        D" );
19    puts( " D      D" );
20    puts( "  DDDDD" );
21 } // end main
```

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

ANS:

```
1  // Exercise 2.26 Solution
```

```
#include <stdio.h>

int main( void )
{
   int integer1; // first integer
   int integer2; // second integer

   printf( "%s", "Input two integers: " ); // prompt user
   scanf( "%d%d", &integer1, &integer2 ); // read two integers

   // use remainder operator
   if ( integer1 % integer2 == 0 ) {
      printf( "%d is a multiple of %d\n", integer1, integer2 );
   } // end if

   if ( integer1 % integer2 != 0 ) {
      printf( "%d is not a multiple of %d\n", integer1, integer2 );
   } // end if
} // end main
```

```
Input two integers: 88 11
88 is a multiple of 11
```

```
Input two integers: 777 5
777 is not a multiple of 5
```

2.27 Display the following checkerboard pattern with eight `printf` statements and then display the same pattern with as few `printf` statements as possible.

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

ANS:

```
// Exercise 2.27 Solution
#include <stdio.h>

int main( void )
{
   puts( "With eight printf() statements:" );

   printf(  "%s", "* * * * * * * *\n" );
   printf(  "%s", " * * * * * * * *\n" );
   printf(  "%s", "* * * * * * * *\n" );
   printf(  "%s", " * * * * * * * *\n" );
   printf(  "%s", "* * * * * * * *\n" );
   printf(  "%s", " * * * * * * * *\n" );
```

```
14      printf( "%s", "* * * * * * * * *\n" );
15      printf( "%s", " * * * * * * * * *\n" );
16
17      puts( "\nNow with one printf() statement:" );
18
19      printf( "%s", "* * * * * * * * *\n * * * * * * * * *\n"
20            "* * * * * * * * *\n * * * * * * * * *\n"
21            "* * * * * * * * *\n * * * * * * * * *\n"
22            "* * * * * * * * *\n * * * * * * * * *\n" );
23  } // end main
```

```
With eight printf() statements:
* * * * * * * * *
 * * * * * * * * *
* * * * * * * * *
 * * * * * * * * *
* * * * * * * * *
 * * * * * * * * *
* * * * * * * * *
 * * * * * * * * *

Now with one printf() statement:
* * * * * * * * *
 * * * * * * * * *
* * * * * * * * *
 * * * * * * * * *
* * * * * * * * *
 * * * * * * * * *
* * * * * * * * *
 * * * * * * * * *
```

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?

ANS: A fatal error causes the program to terminate prematurely. A nonfatal error occurs when the logic of the program is incorrect, and the program does not work properly. A fatal error is preferred for debugging purposes. A fatal error immediately lets you know there is a problem with the program, whereas a nonfatal error can be subtle and possibly go undetected.

2.29 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.

ANS:

```
1  // Exercise 2.29 Solution
2  #include <stdio.h>
3
```

```c
int main( void )
{
   printf( "A's integer equivalent is %d\n", 'A' );
   printf( "B's integer equivalent is %d\n", 'B' );
   printf( "C's integer equivalent is %d\n", 'C' );
   printf( "a's integer equivalent is %d\n", 'a' );
   printf( "b's integer equivalent is %d\n", 'b' );
   printf( "c's integer equivalent is %d\n", 'c' );
   printf( "0's integer equivalent is %d\n", '0' );
   printf( "1's integer equivalent is %d\n", '1' );
   printf( "2's integer equivalent is %d\n", '2' );
   printf( "$'s integer equivalent is %d\n", '$' );
   printf( "*'s integer equivalent is %d\n", '*' );
   printf( "+'s integer equivalent is %d\n", '+' );
   printf( "/'s integer equivalent is %d\n", '/' );
   printf( "The blank character's integer equivalent is %d\n", ' ' );
} // end main
```

```
A's integer equivalent is 65
B's integer equivalent is 66
C's integer equivalent is 67
a's integer equivalent is 97
b's integer equivalent is 98
c's integer equivalent is 99
0's integer equivalent is 48
1's integer equivalent is 49
2's integer equivalent is 50
$'s integer equivalent is 36
*'s integer equivalent is 42
+'s integer equivalent is 43
/'s integer equivalent is 47
The blank character's integer equivalent is 32
```

2.30 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
```

ANS:

```c
// Exercise 2.30 Solution
#include <stdio.h>

int main( void )
{
   int number; // number input by user
   int temp; // temporary integer

   printf(  "%s", "Enter a five-digit number: " ); // prompt user
   scanf( "%d", &number ); // read integer

```

```
12      printf( "%d ", number / 10000 ); // print leftmost digit
13      temp = number % 10000;
14
15      printf( "  %d ", temp / 1000 );
16      temp = temp % 1000;
17
18      printf( "  %d ", temp / 100 );
19      temp = temp % 100;
20
21      printf( "  %d ", temp / 10 );
22      temp = temp % 10;
23
24      printf( "  %d\n", temp ); // print right-most digit
25   } // end main
```

```
Enter a five-digit number: 23456
2   3   4   5   6
```

2.31 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
```

ANS:

```
1   // Exercise 2.31 Solution
2   #include <stdio.h>
3
4   int main( void )
5   {
6      int count = 0; // initialize count to zero
7
8      // calculate the squares and cubes for the numbers 0 to 10
9      puts( "\nnumber\tsquare\tcube" );
10     printf( "%d\t%d\t%d\n", count, count * count,
11        count * count * count );
12
13     count = count + 1; // increment count by 1
14     printf( "%d\t%d\t%d\n", count, count * count,
15        count * count * count );
16
17     count = count + 1;
18     printf( "%d\t%d\t%d\n", count, count * count,
```

```
19          count * count * count );
20
21       count = count + 1;
22       printf( "%d\t%d\t%d\n", count, count * count,
23          count * count * count );
24
25       count = count + 1;
26       printf( "%d\t%d\t%d\n", count, count * count,
27          count * count * count );
28
29       count = count + 1;
30       printf( "%d\t%d\t%d\n", count, count * count,
31          count * count * count );
32
33       count = count + 1;
34       printf( "%d\t%d\t%d\n", count, count * count,
35          count * count * count );
36
37       count = count + 1;
38       printf( "%d\t%d\t%d\n", count, count * count,
39          count * count * count );
40
41       count = count + 1;
42       printf( "%d\t%d\t%d\n", count, count * count,
43          count * count * count );
44
45       count = count + 1;
46       printf( "%d\t%d\t%d\n", count, count * count,
47          count * count * count );
48
49       count = count + 1;
50       printf( "%d\t%d\t%d\n", count, count * count,
51          count * count * count );
52    } // end main
```

Making a Difference

2.32 ***(Body Mass Index Calculator)*** We introduced the body mass index (BMI) calculator in Exercise 1.11. 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 3 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.]

ANS:

```
1  // Exercise 2.32 Solution: BMI.c
2  // Making a Difference: Body Mass Index Calculator
3  #include <stdio.h>
4
5  //function main begins program execution
6  int main ( void )
7  {
8     int weight; // weight of the person
9     int height; // height of the person
10    int BMI; // user's BMI
11
12    // get user's height
13    printf(  "%s", "Please enter your height (in inches): " );
14    scanf( "%d", &height );
15
16    // get user's weight
17    printf( "Please enter your weight (in pounds): " );
18    scanf( "%d", &weight );
19
20    BMI = weight * 703 / ( height * height ); // calculate BMI
21
22    printf( "Your BMI is %d\n\n", BMI ); // output BMI
23
24    // output data to user
25    puts( "BMI VALUES" );
26    puts( "Underweight:\tless than 18.5" );
27    puts( "Normal:\t\tbetween 18.5 and 24.9" );
28    puts( "Overweight:\tbetween 25 and 29.9" );
29    puts( "Obese:\t\t30 or greater" );
30 } // end main
```

```
Please enter your height (in inches): 69
Please enter your weight (in pounds): 155
Your BMI is 22

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

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.

ANS:

```
1  // Exercise 2.33 Solution
2  // Making a Difference: Car-Pool Savings Calculator
3  #include <stdio.h>
4
5  // function main begins program execution
6  int main ( void )
7  {
8     int miles; // total miles driven per day
9     int gasCost; // cost per gallon of gasoline
10    int mpg; // average miles per gallon
11    int parkFee; // parking fees per day
12    int tolls; // tolls per day
13    int total; // total cost
14
15    // get total miles driven
16    printf( "%s", "Please enter the total miles driven per day: " );
17    scanf( "%d", &miles );
18
19    // get cost of gas
20    printf( "%s", "Please enter the cost per gallon of gasoline: " );
21    scanf( "%d", &gasCost );
22
23    // get average miles per gallon
24    printf( "%s", "Please enter average miles per gallon: " );
25    scanf( "%d", &mpg );
26
27    // get parking fees per day
28    printf( "%s", "Please enter the parking fees per day: " );
29    scanf( "%d", &parkFee );
30
31    // get cost of tolls per day
32    printf( "%s", "Please enter the tolls per day: " );
```

```
33      scanf( "%d", &tolls );
34
35      // calculate total cost
36      total = tolls + parkFee + ( miles / mpg ) * gasCost;
37
38      printf( "Your daily cost of driving to work is $%d\n", total );
39   } // end main
```

```
Please enter the total miles driven per day: 100
Please enter the cost per gallon of gasoline: 3
Please enter average miles per gallon: 19
Please enter the parking fees per day: 3
Please enter the tolls per day: 4
Your daily cost of driving to work is $22
```