Computer Programming I

Ming-Feng Tsai (Victor Tsai)

Dept. of Computer Science
National Chengchi University

C Program Control

Objectives

- In this chapter, you'll learn
  - The essential of counter-controlled repetition
  - To use the for and do...while repetition statement
  - To understand multiple selection using the switch statement
  - To use the break and continue statements to alter the flow of control
  - To use the logical operators to form complex conditional expressions in control statements

Introduction

- In this chapter, repetition is considered in greater detail, and additional repetition control statements, namely the **for** and the **do…while**, are presented.
- The **switch** multiple-selection statement is introduced.
- We discuss the **break** statement for exiting immediately from certain control statements, and the **continue** statement for skipping the remainder of the body of a repetition statement and proceeding with the next iteration of the loop.
- The chapter discusses logical operators used for combining conditions, and summarizes the principles of structured programming as presented in Chapter 3 and 4.

Repetition Essentials

- A loop is a group of instructions the computer executes repeatedly while some loop-continuation condition remains true.
- We have discussed two means of repetition:
  - Counter-controlled repetition
  - Sentinel-controlled repetition

Repetition Essentials (Cont.)

- In counter-controlled repetition, a control variable is used to count the number of repetitions.
- When the value of the control variable indicates that the number of repetitions has been performed, the loop terminates and the computer continues executing with the statement after the repetition statement.

Repetition Essentials (Cont.)

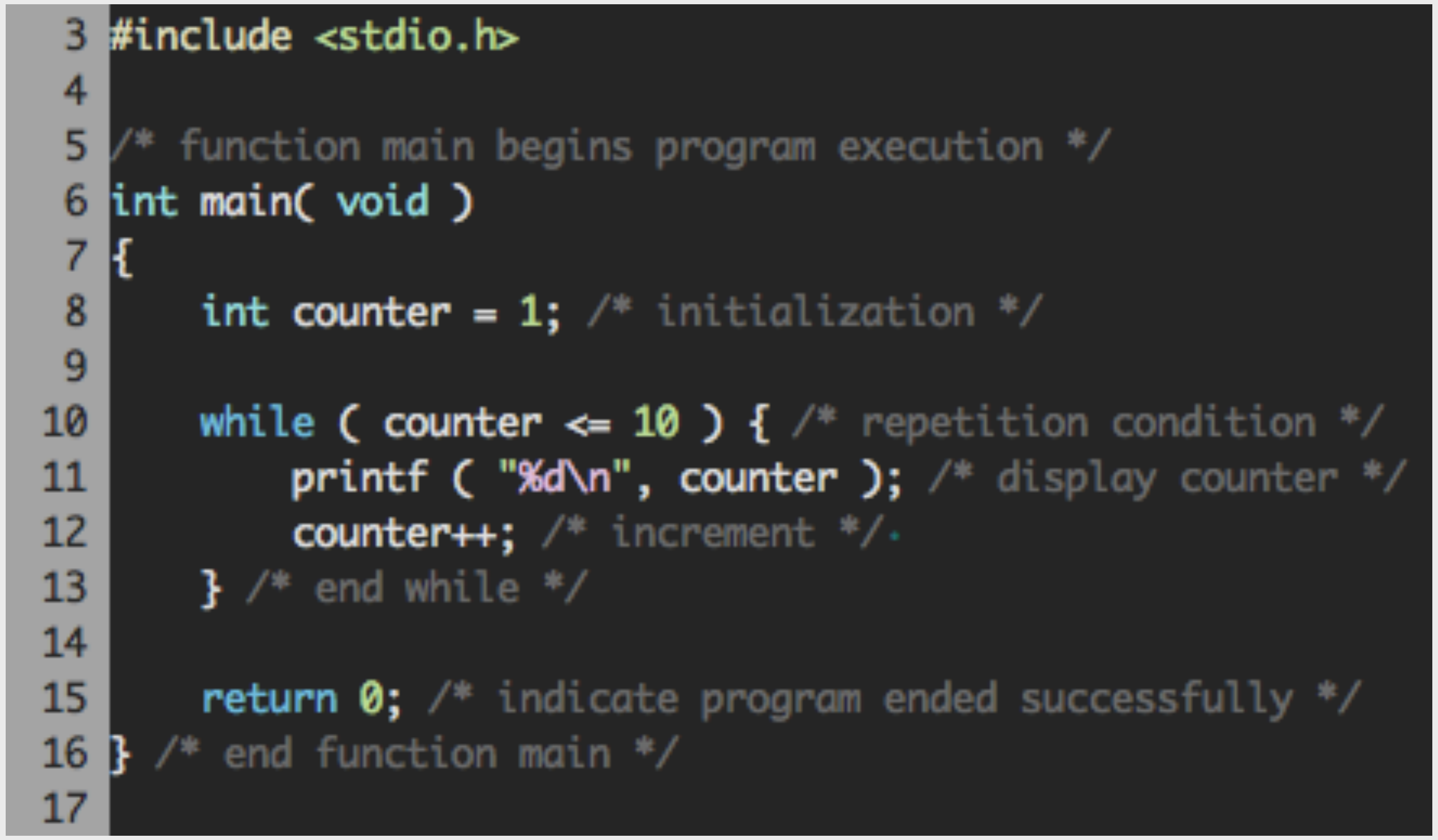
- In sentinel-controlled repetition, sentinel values are used to control repetition when:
    - The precise number of repetitions is not known in advance, and
    - The loop includes statements that obtain data each time the loop is performed.

Counter-Controlled Repetition

- Counter-controlled repetition requires:
    - The name of a control variable (or loop counter).
    - The initial value of the control variable.
    - The increment (or decrement) by which the control variable is modified each time through the loop.
    - The condition that tests for the final value of the control variable (i.e., whether looping should continue).

Counter-Controlled Repetition (Cont.)

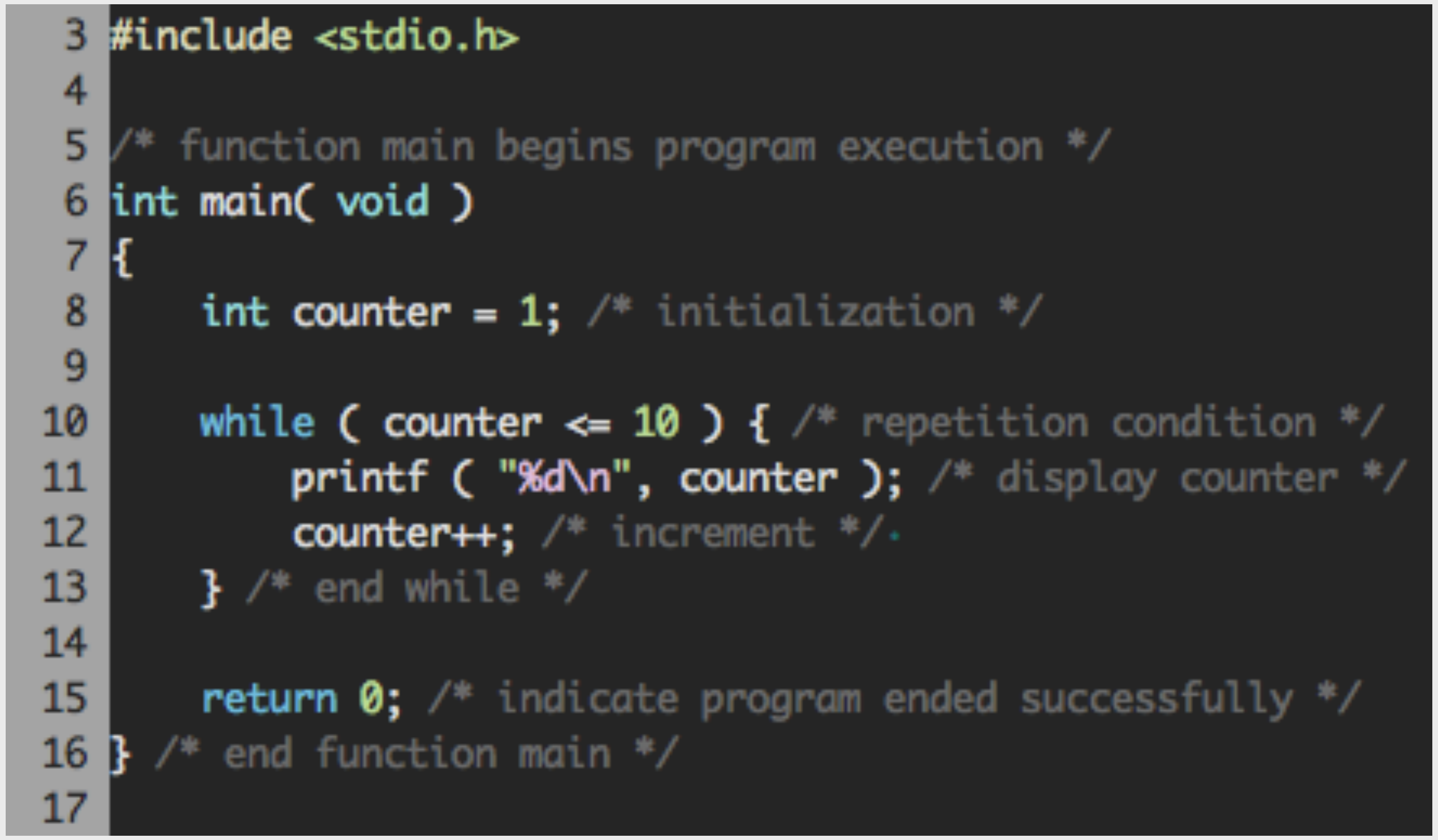
- Example: fig04_01.c



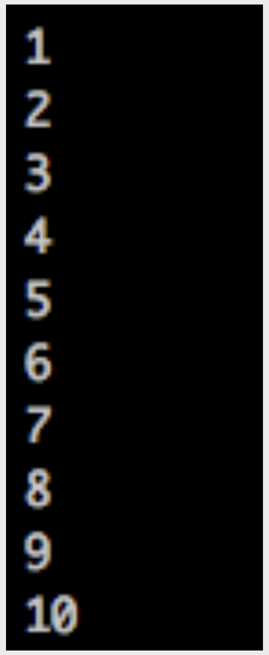
```
#include <stdio.h>

/* function main begins program execution */
int main( void )
{
    int counter = 1; /* initialization */

    while ( counter <= 10 ) { /* repetition condition */
        printf ( "%d\n", counter ); /* display counter */
        counter++; /* increment */
    } /* end while */

    return 0; /* indicate program ended successfully */
} /* end function main */

```

Counter-Controlled Repetition (Cont.)

- Example: fig04_01.c



```c
#include <stdio.h>

/* function main begins program execution */
int main( void )
{
    int counter = 1; /* initialization */

    while ( counter <= 10 ) { /* repetition condition */
        printf ( "%d\n", counter ); /* display counter */
        counter++; /* increment */
    } /* end while */

    return 0; /* indicate program ended successfully */
} /* end function main */
```



```
1
2
3
4
5
6
7
8
9
10
```

Counter-Controlled Repetition (Cont.)

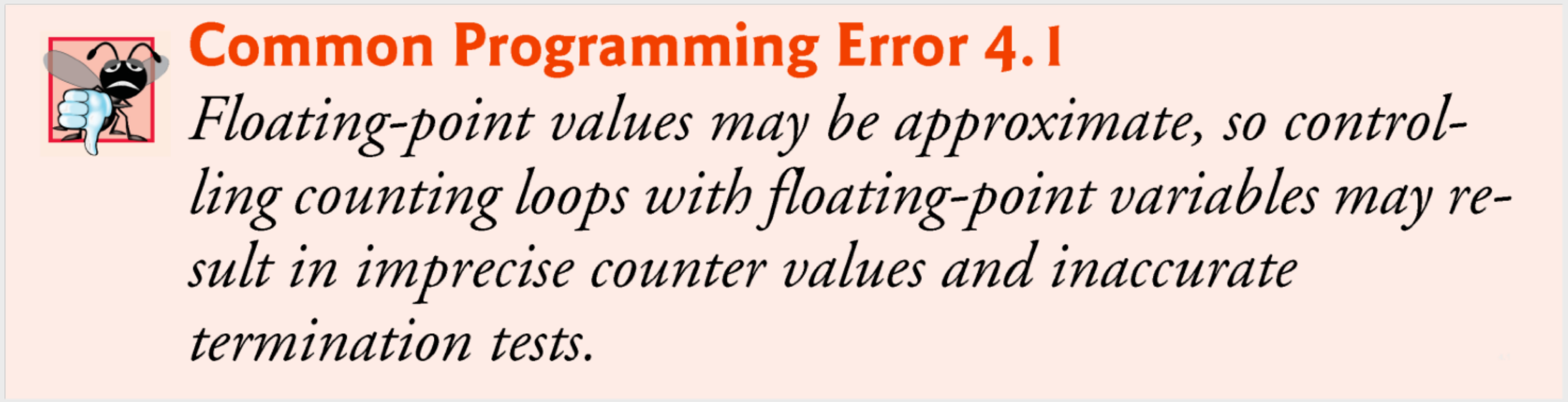
- You could make the program in fig04_01.c more concise by initializing counter to 0 and by replacing the while statement with

```
while ( ++counter <= 10 )
    printf( "%d\n", counter );
```

- This code eliminates the need for the braces around the body of the while because the while now contains only one statement.

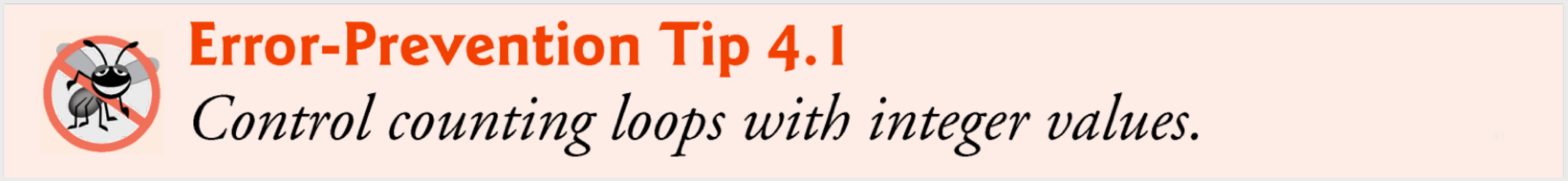
Common Programming Error 4.1

Floating-point values may be approximate, so controlling counting loops with floating-point variables may result in imprecise counter values and inaccurate termination tests.



Common Programming Error 4.1

Floating-point values may be approximate, so controlling counting loops with floating-point variables may result in imprecise counter values and inaccurate termination tests.



Error-Prevention Tip 4.1

Control counting loops with integer values.

Good Programming Practice 4.1

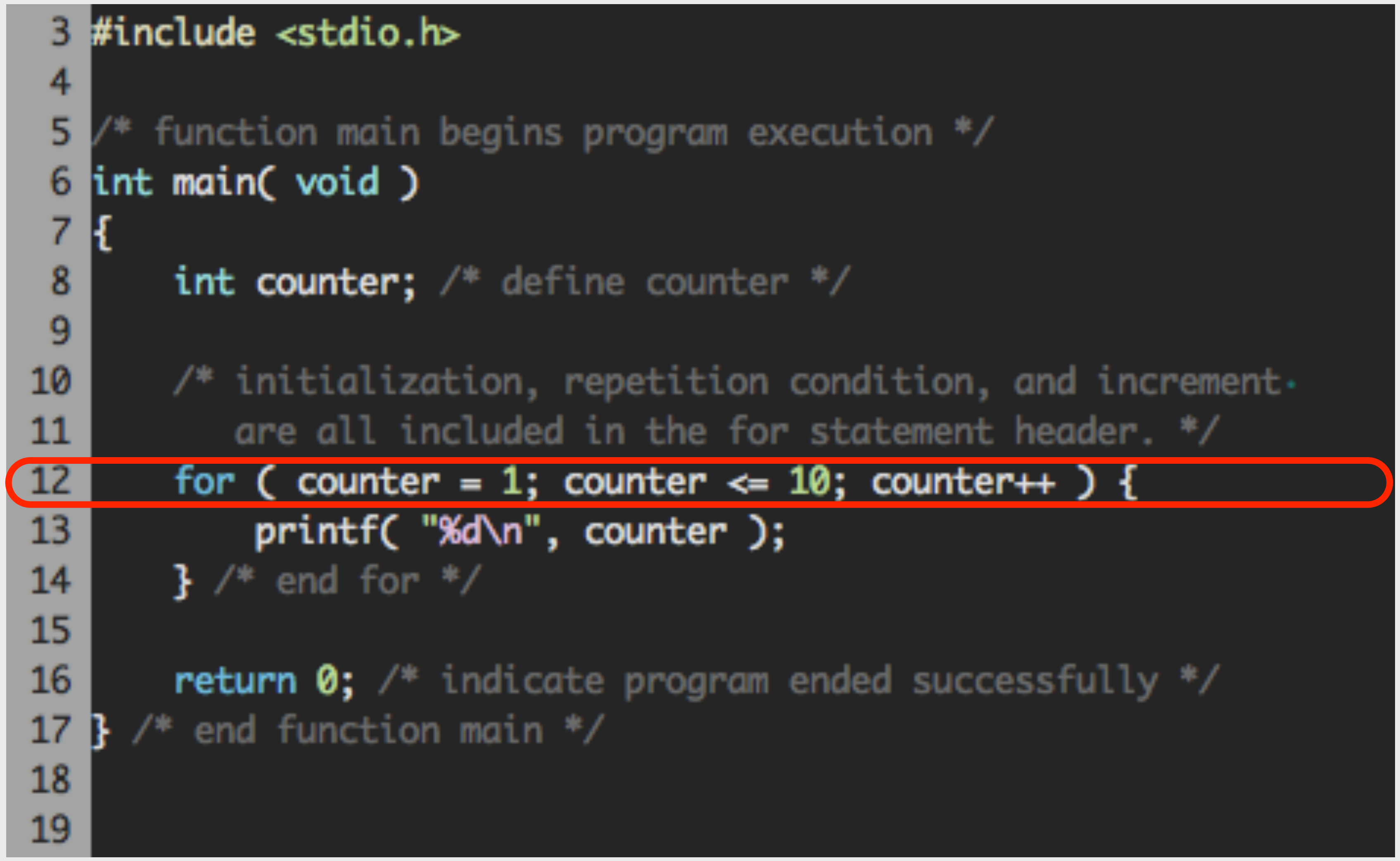
Too many levels of nesting can make a program difficult to understand. As a rule, try to avoid using more than three levels of nesting.

`for` Repetition Statement

- The for repetition statement handles all the details of counter-controlled repetition.
- To illustrate its power, let’s rewrite the program of fig04_01.c

`for` Repetition Statement (Cont.)

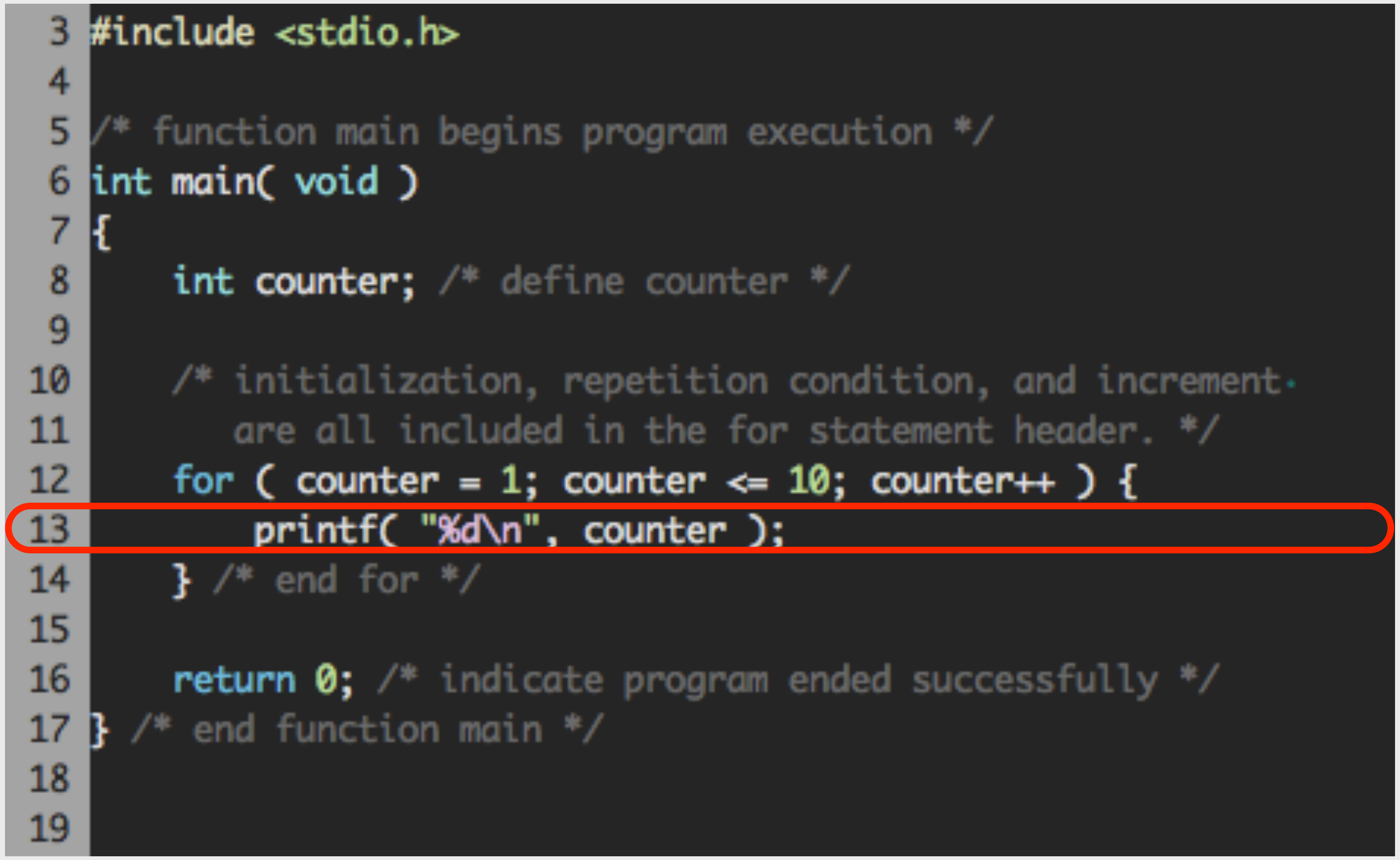
- Example: fig04_02.c



```
#include <stdio.h>

/* function main begins program execution */
int main( void )
{
    int counter; /* define counter */

    /* initialization, repetition condition, and increment
       are all included in the for statement header. */
    for ( counter = 1; counter <= 10; counter++ ) {
        printf( "%d\n", counter );
    } /* end for */

    return 0; /* indicate program ended successfully */
} /* end function main */

```

for Repetition Statement (Cont.)

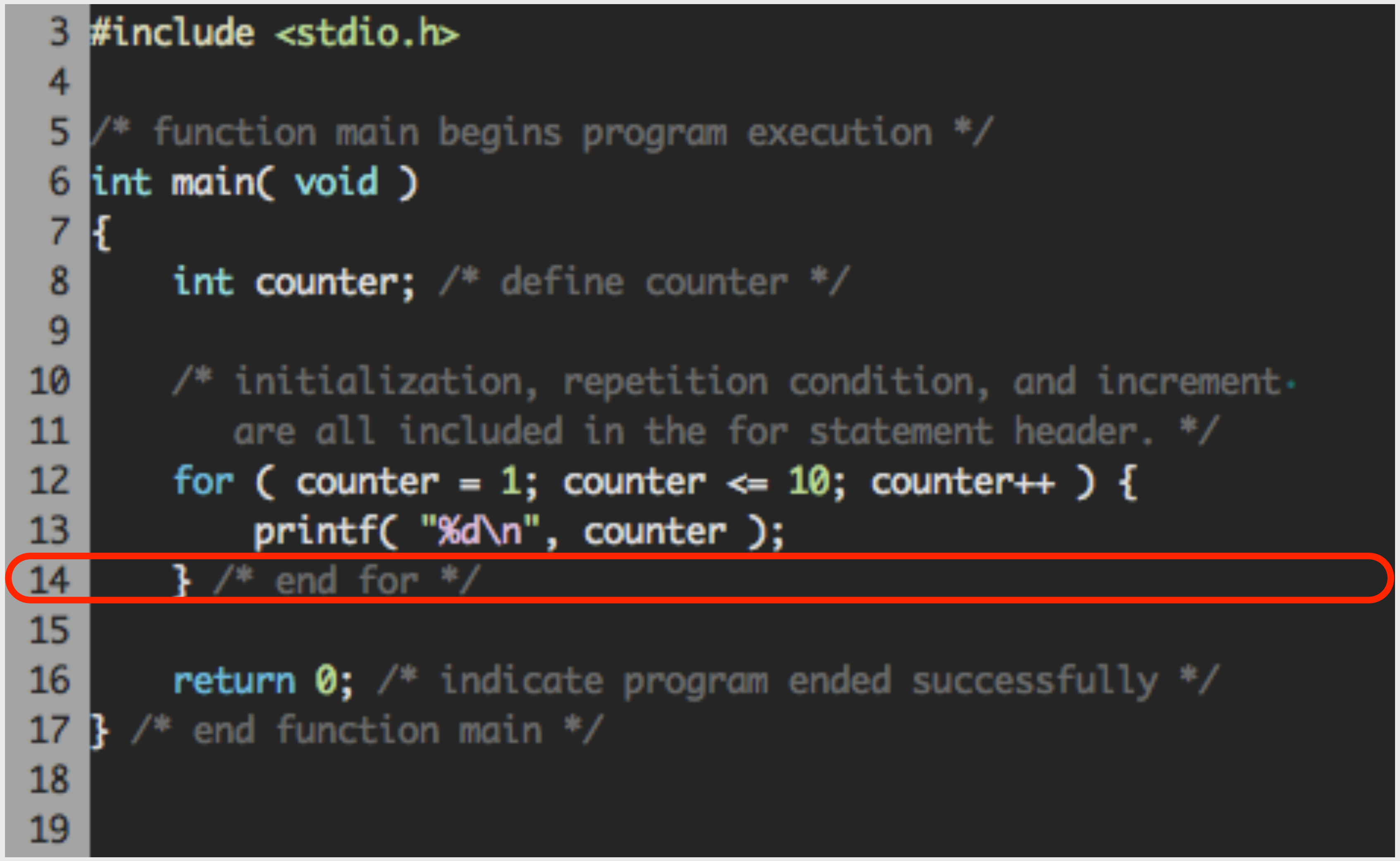
- Example: fig04_02.c



```c
#include <stdio.h>

/* function main begins program execution */
int main( void )
{
    int counter; /* define counter */

    /* initialization, repetition condition, and increment
       are all included in the for statement header. */
    for ( counter = 1; counter <= 10; counter++ ) {
        printf( "%d\n", counter );
    } /* end for */

    return 0; /* indicate program ended successfully */
} /* end function main */
```

`for` Repetition Statement (Cont.)

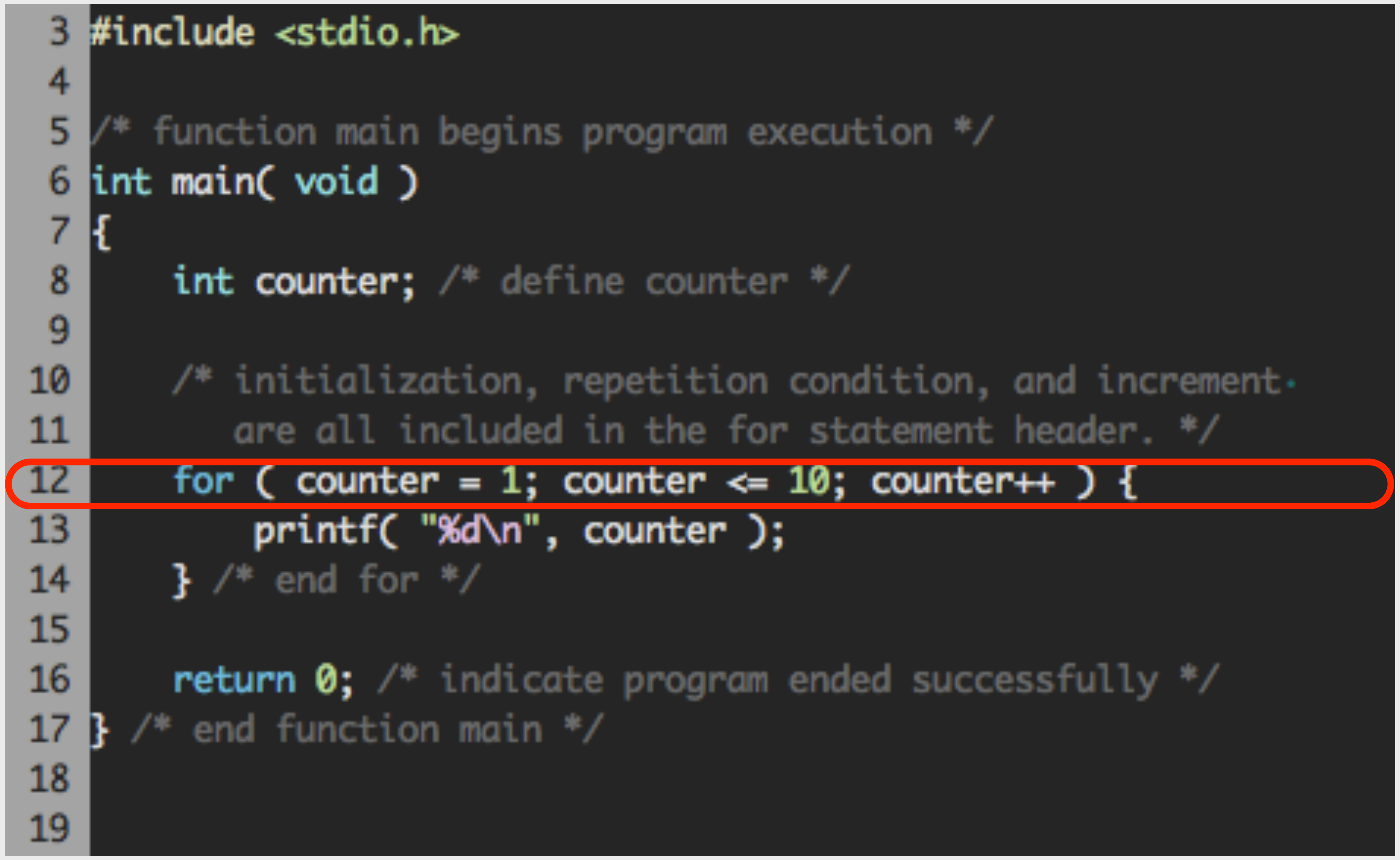
- Example: fig04_02.c



```c
#include <stdio.h>

/* function main begins program execution */
int main( void )
{
    int counter; /* define counter */

    /* initialization, repetition condition, and increment
       are all included in the for statement header. */
    for ( counter = 1; counter <= 10; counter++ ) {
        printf( "%d\n", counter );
    } /* end for */

    return 0; /* indicate program ended successfully */
} /* end function main */


```

`for` Repetition Statement (Cont.)

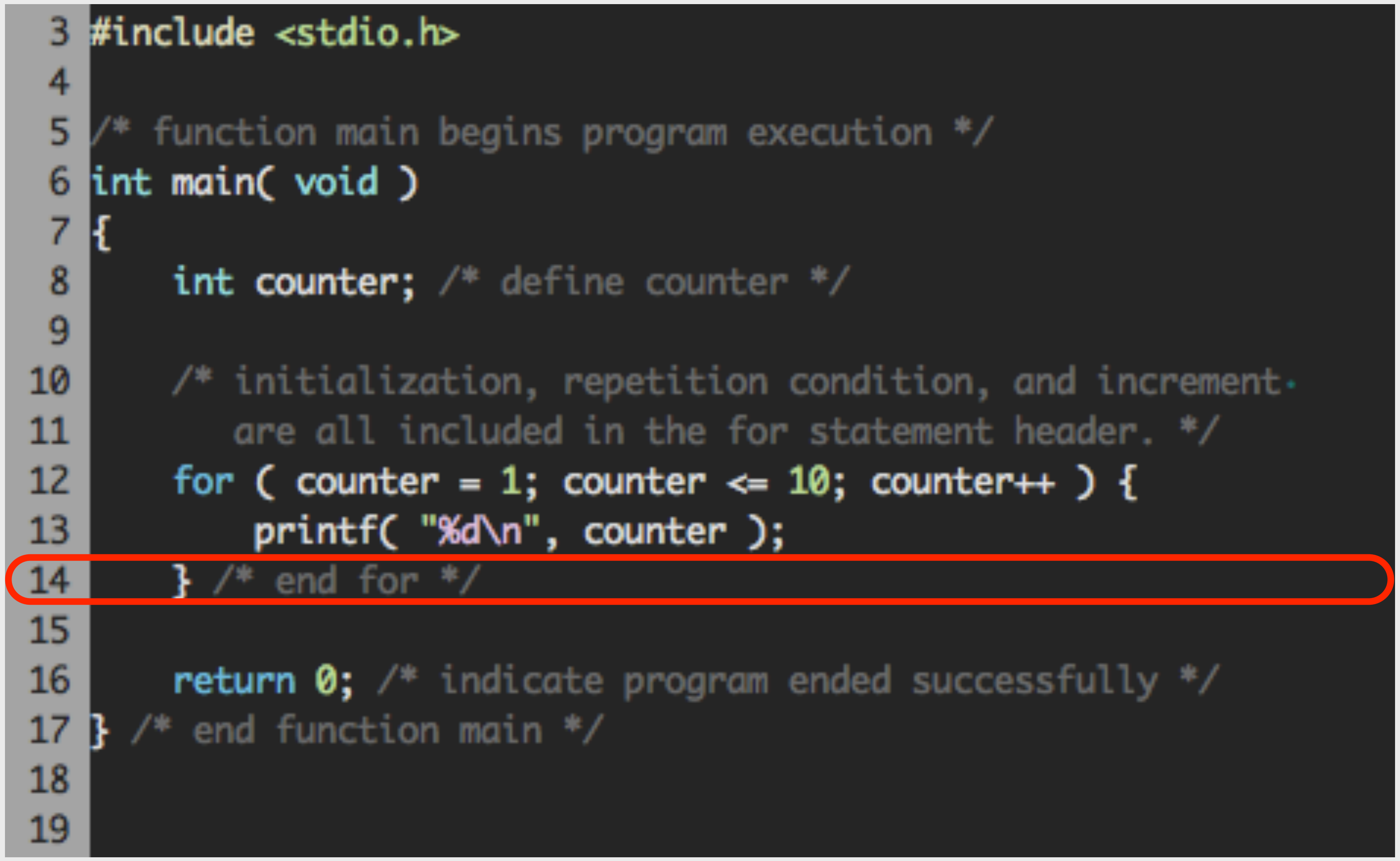
- Example: fig04_02.c



```c
#include <stdio.h>

/* function main begins program execution */
int main( void )
{
    int counter; /* define counter */

    /* initialization, repetition condition, and increment
       are all included in the for statement header. */
    for ( counter = 1; counter <= 10; counter++ ) {
        printf( "%d\n", counter );
    } /* end for */

    return 0; /* indicate program ended successfully */
} /* end function main */
```

for Repetition Statement (Cont.)

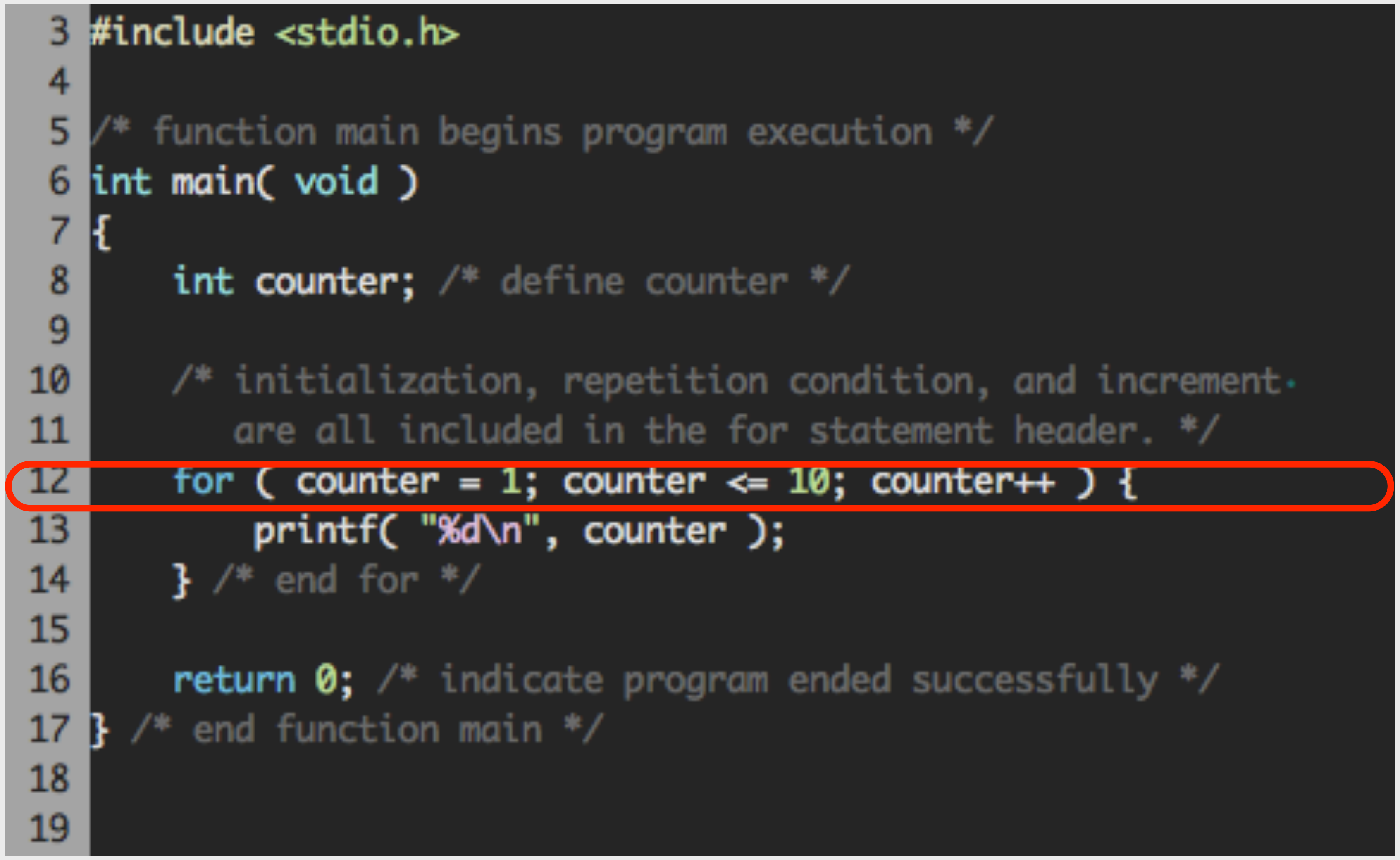
- Example: fig04_02.c



```
#include <stdio.h>

/* function main begins program execution */
int main( void )
{
    int counter; /* define counter */

    /* initialization, repetition condition, and increment
       are all included in the for statement header. */
    for ( counter = 1; counter <= 10; counter++ ) {
        printf( "%d\n", counter );
    } /* end for */

    return 0; /* indicate program ended successfully */
} /* end function main */


```

for Repetition Statement (Cont.)

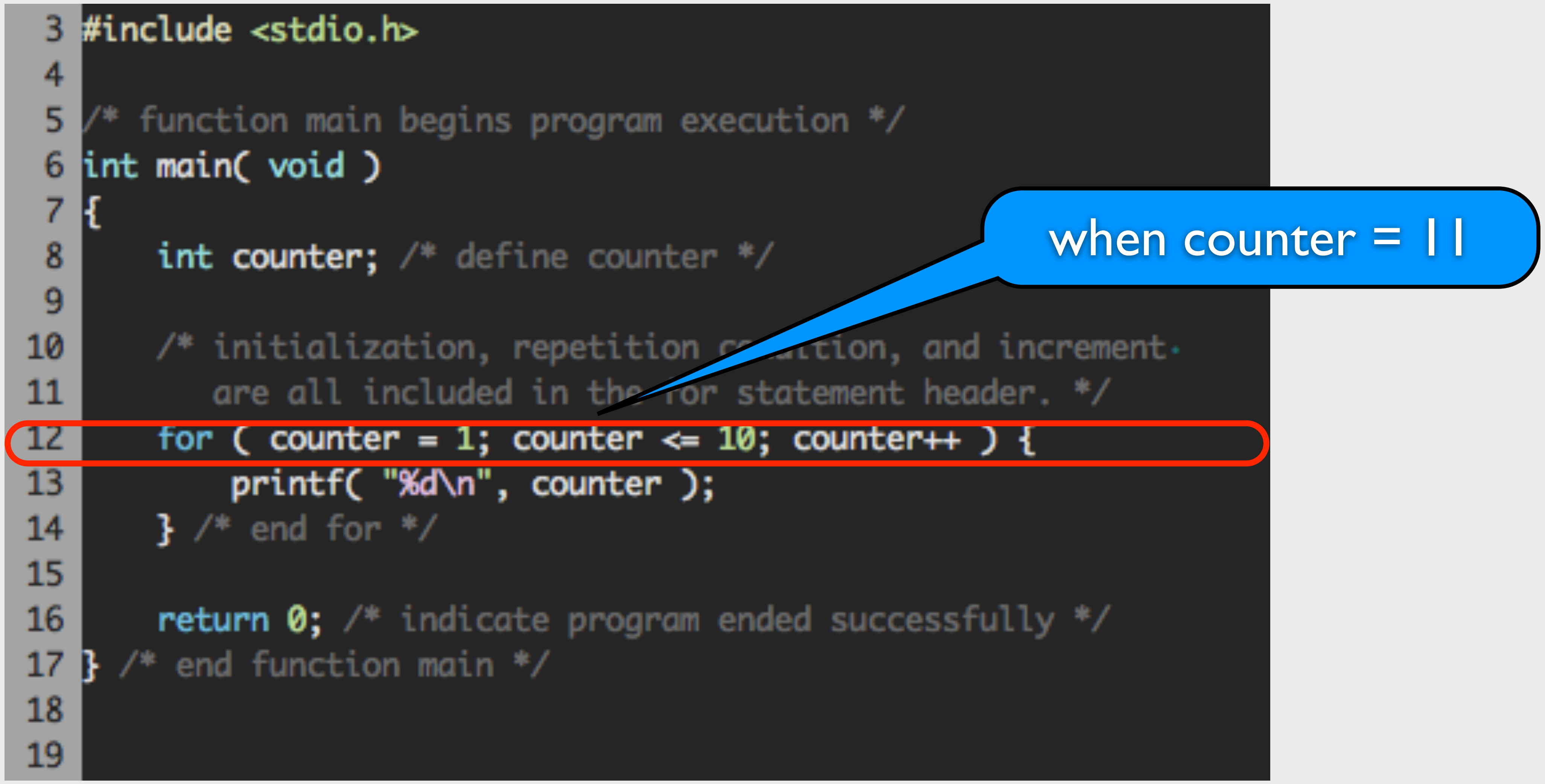
- Example: fig04_02.c



```c
#include <stdio.h>

/* function main begins program execution */
int main( void )
{
    int counter; /* define counter */

    /* initialization, repetition condition, and increment
       are all included in the for statement header. */
    for ( counter = 1; counter <= 10; counter++ ) {
        printf( "%d\n", counter );
    } /* end for */

    return 0; /* indicate program ended successfully */
} /* end function main */


```

for Repetition Statement (Cont.)

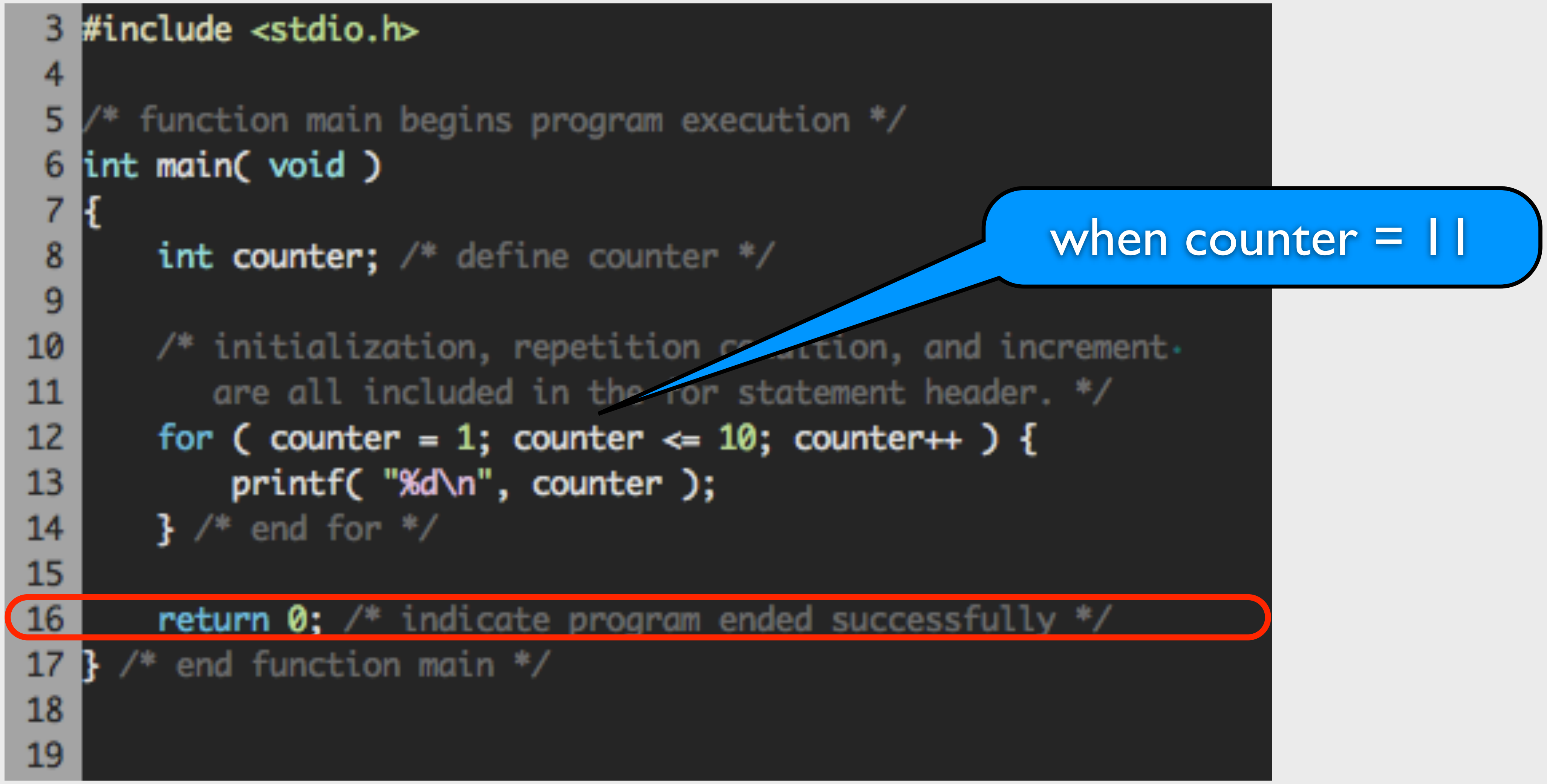
- Example: fig04_02.c



```
#include <stdio.h>

/* function main begins program execution */
int main( void )
{
    int counter; /* define counter */

    /* initialization, repetition condition, and increment
       are all included in the for statement header. */
    for ( counter = 1; counter <= 10; counter++ ) {
        printf( "%d\n", counter );
    } /* end for */

    return 0; /* indicate program ended successfully */
} /* end function main */
```

`for` Repetition Statement (Cont.)

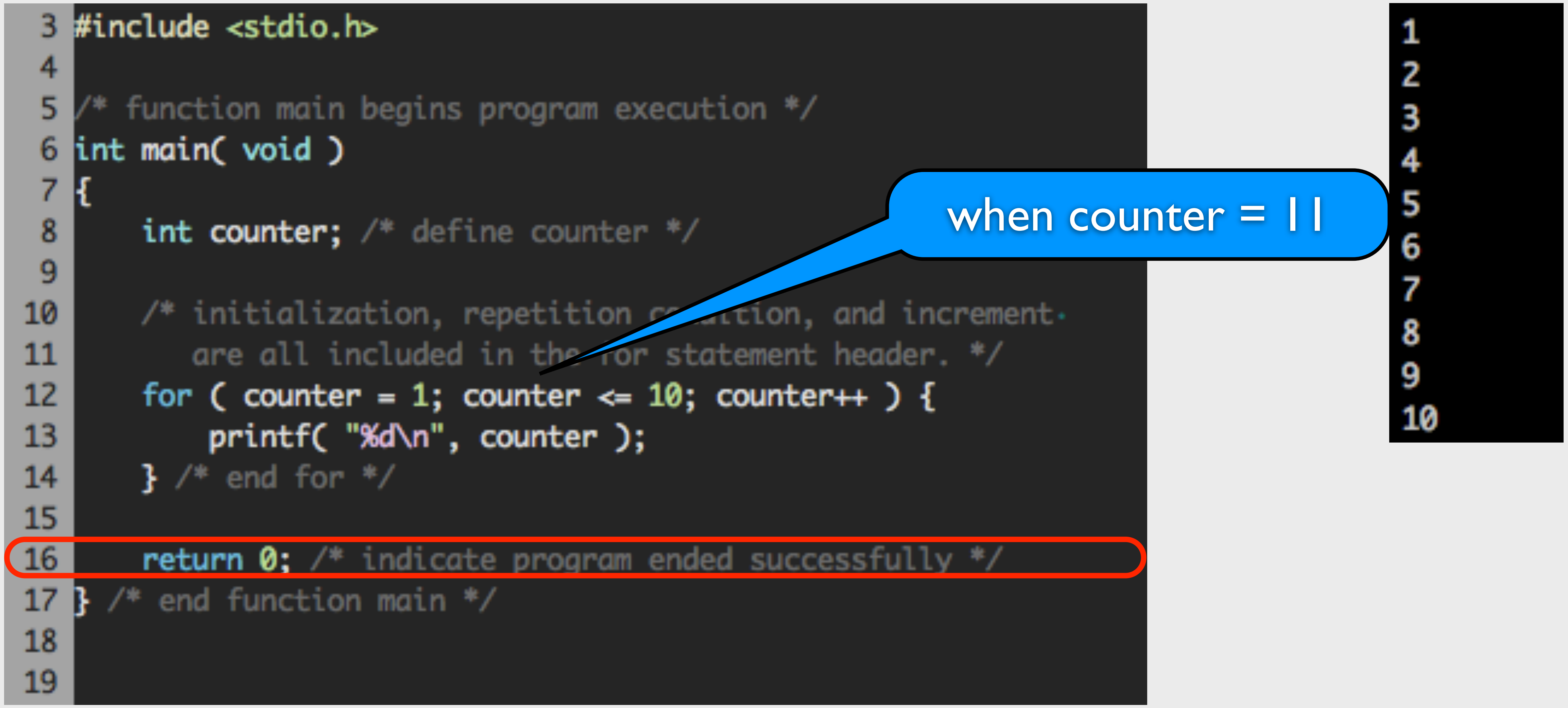
- Example: fig04_02.c



```
#include <stdio.h>

/* function main begins program execution */
int main( void )
{
    int counter; /* define counter */

    /* initialization, repetition condition, and increment
       are all included in the for statement header. */
    for ( counter = 1; counter <= 10; counter++ ) {
        printf( "%d\n", counter );
    } /* end for */

    return 0; /* indicate program ended successfully */
} /* end function main */
```

for Repetition Statement (Cont.)

- Example: fig04_02.c



```c
#include <stdio.h>

/* function main begins program execution */
int main( void )
{
    int counter; /* define counter */

    /* initialization, repetition condition, and increment
       are all included in the for statement header. */
    for ( counter = 1; counter <= 10; counter++ ) {
        printf( "%d\n", counter );
    } /* end for */

    return 0; /* indicate program ended successfully */
} /* end function main */
```

```
1
2
3
4
5
6
7
8
9
10
```

`for` Repetition Statement (Cont.)

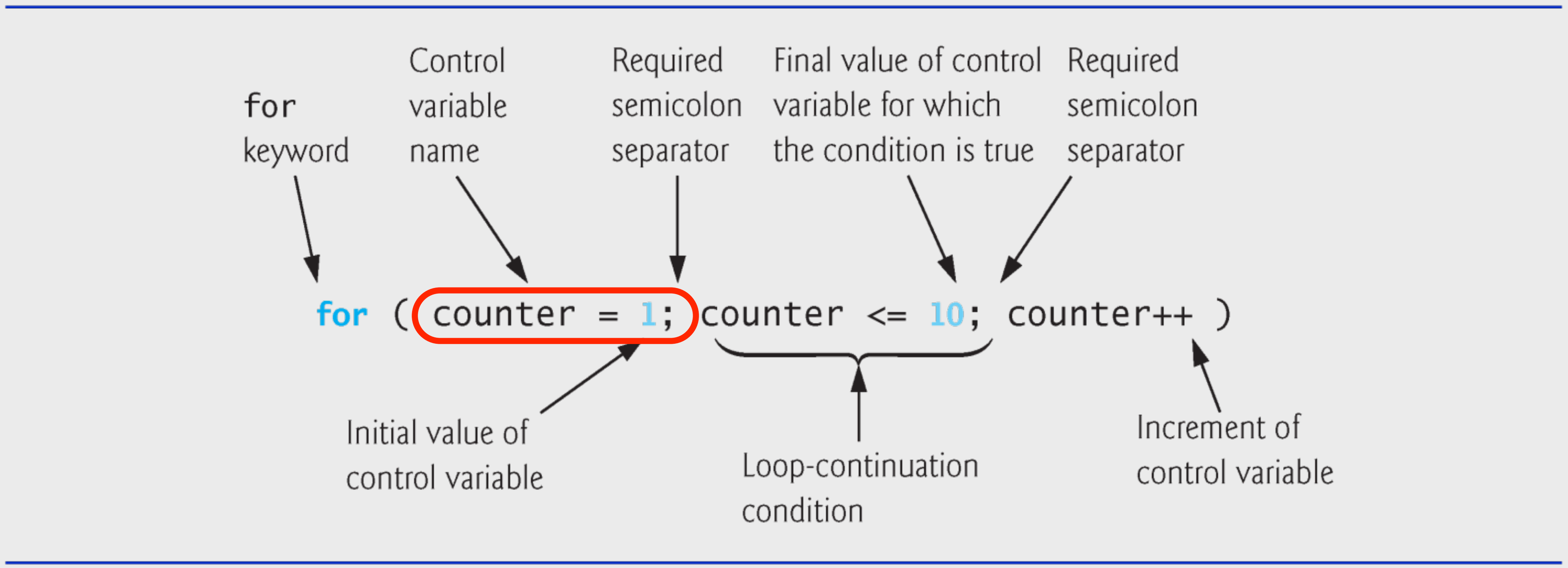

for keyword
Control variable name
Required semicolon separator
Final value of control variable for which the condition is true
Required semicolon separator
for (counter = 1; counter <= 10; counter++)
Initial value of control variable
Loop-continuation condition
Increment of control variable

Fig. 4.3 | for statement header components.

`for` Repetition Statement (Cont.)

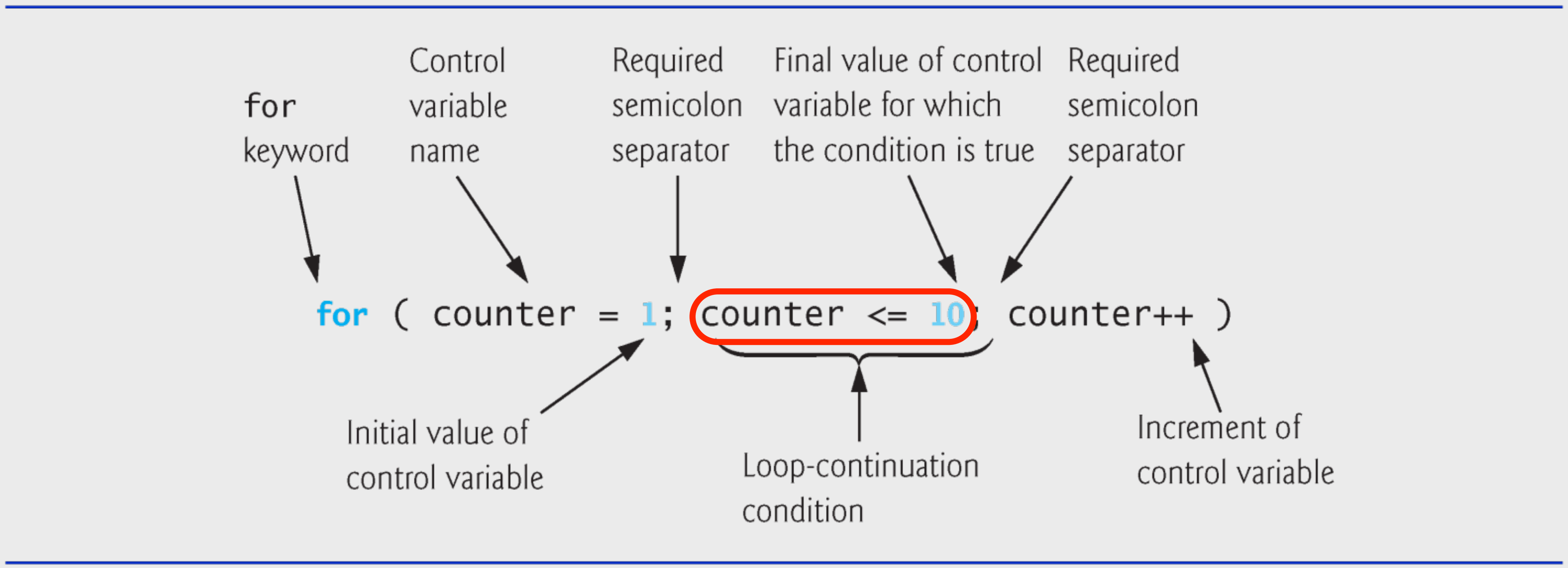

for keyword
Control variable name
Required semicolon separator
Final value of control variable for which the condition is true
Required semicolon separator
for (counter = 1; counter <= 10; counter++)
Initial value of control variable
Loop-continuation condition
Increment of control variable

Fig. 4.3 | `for` statement header components.

`for` Repetition Statement (Cont.)

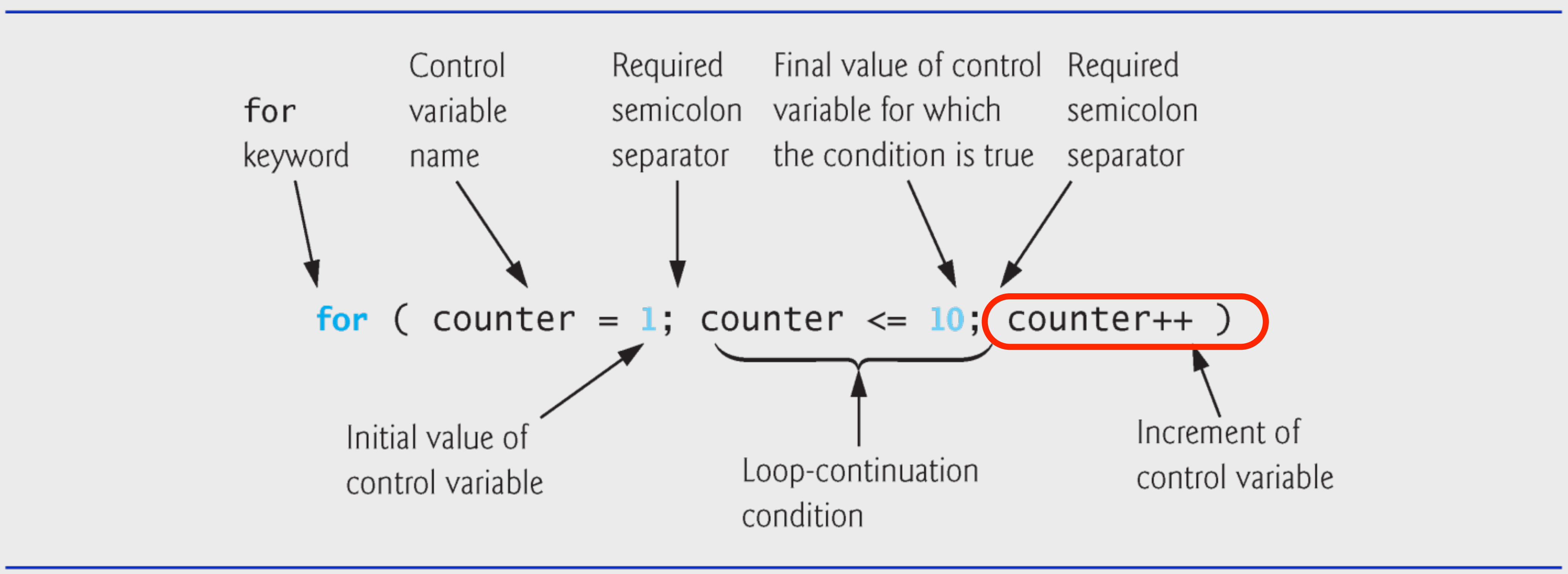

for keyword
Control variable name
Required semicolon separator
Final value of control variable for which the condition is true
Required semicolon separator
for (counter = 1; counter <= 10; counter++)
Initial value of control variable
Loop-continuation condition
Increment of control variable

Fig. 4.3 | `for` statement header components.

`for` Repetition Statement (Cont.)

for Repetition Statement (Cont.)

- The general format of the for statement is

```
for ( expression1; expression2; expression3 )
{
                     statements
}
```

for Repetition Statement (Cont.)

- The general format of the for statement is

```
for ( expression1; expression2; expression3 )
{
                        statements
}
```

  - **expression1** initializes the loop-control variable

`for` Repetition Statement (Cont.)

- The general format of the for statement is

```
for ( expression1; expression2; expression3 )
{
                          statements
}
```

  - **`expression1`** initializes the loop-control variable
  - **`expression2`** is the loop-continuation condition

`for` Repetition Statement (Cont.)

- The general format of the for statement is

```
for ( expression1; expression2; expression3 )
{
                        statements
}
```

  - **`expression1`** initializes the loop-control variable
  - **`expression2`** is the loop-continuation condition
  - **`expression3`** increments the control variable

`for` Repetition Statement (Cont.)

- In most cases, the for statement can be represented with an equivalent while statement as follows:

```
expression1;
while ( expression2 ) {
      statements
      expression3;
}
```

`for` Repetition Statement (Cont.)

- Often, **expression1** and **expression3** are comma-separated lists of expressions.
- Its primary use is to enable you to use multiple initialization and/or multiple increment expressions.
- For example, there may be two control variables in a single for statement that must be initialized and incremented.

`for` Repetition Statement (Cont.)

- The three expressions in the for statement are **optional**.
- If **expression2** is omitted, C assumes that the condition is true, thus creating an **infinite loop**.
- One may omit **expression1** if the control variable is initialized elsewhere in the program.
- **expression3** may be omitted if the increment is calculated by statements in the body of the for statement or if no increment is needed.

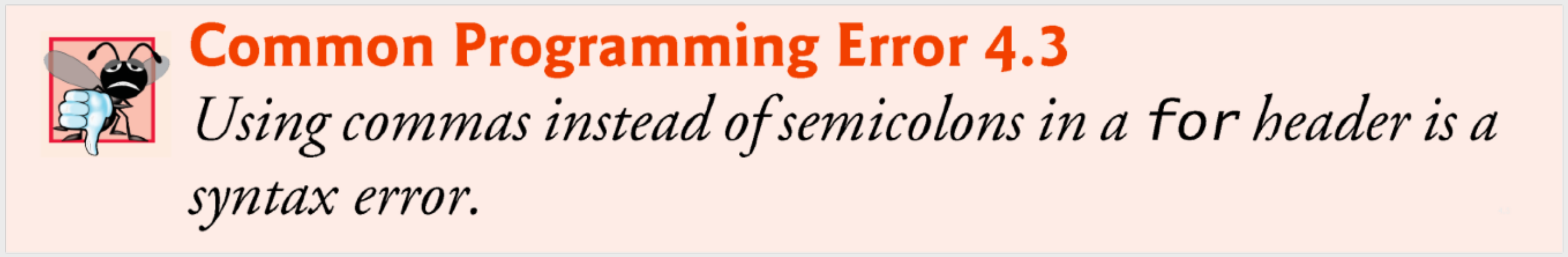
`for` Repetition Statement (Cont.)

- Therefore, the expressions

  ```
  counter = counter + 1
  counter += 1
  ++counter
  counter++
  ```

  are all equivalent in the increment part of the for statement.
- The two semicolons in the **`for`** statement are required.

`for` Repetition Statement (Cont.)



Common Programming Error 4.3

Using commas instead of semicolons in a `for` *header is a syntax error.*

`for` statement: Notes and Observations

- The initialization, loop-continuation condition and increment can contain arithmetic expressions. For example, if x = 2 and y = 10, the statement

```
for(j = x; j <= 4 * x * y; j += y / x)
```

is equivalent to the statement

```
for ( j = 2; j <= 80; j += 5 )
```

`for` statement: Notes and Observations (Cont.)

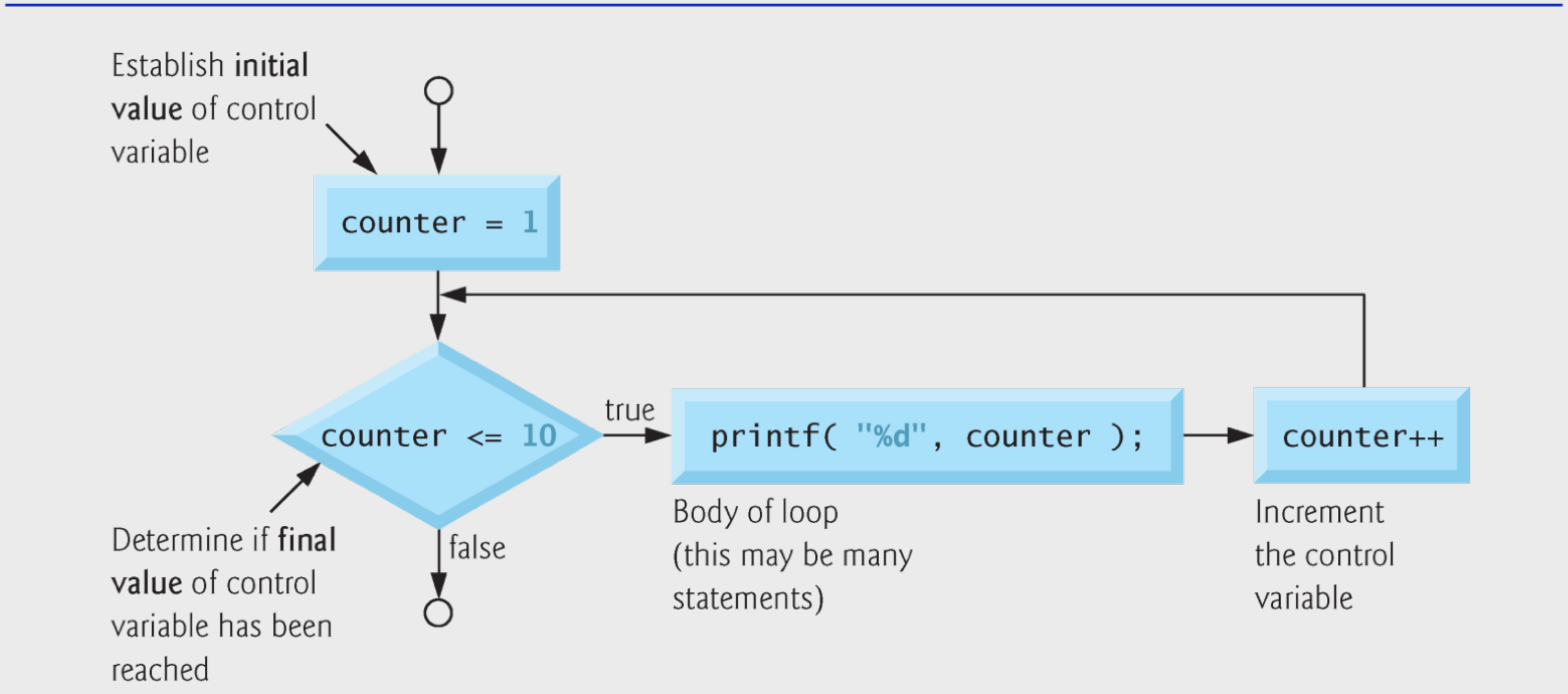
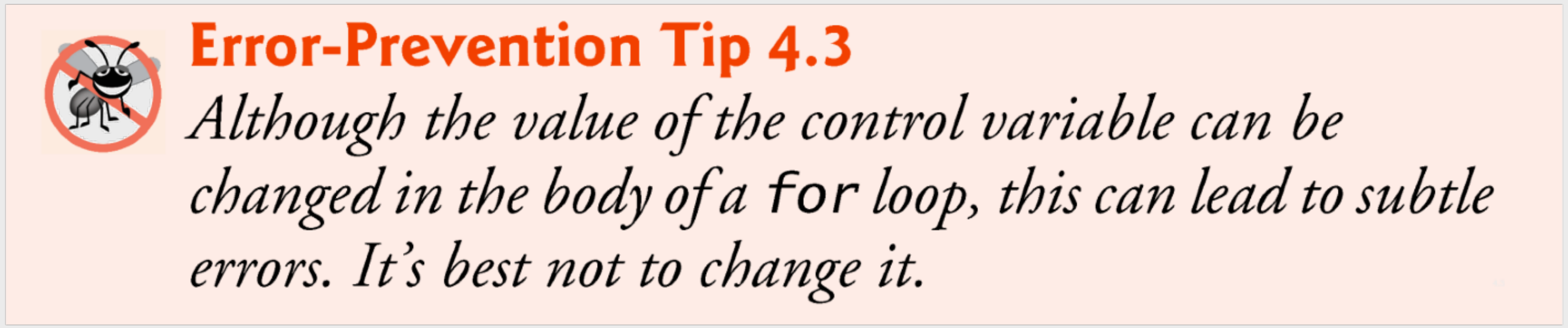

Establish initial value of control variable
counter = 1
counter <= 10
true
printf("%d", counter);
counter++
false
Determine if final value of control variable has been reached
Body of loop (this may be many statements)
Increment the control variable

Fig. 4.4 | Flowcharting a typical `for` repetition statement.

`for` statement: Notes and Observations (Cont.)



Error-Prevention Tip 4.3

Although the value of the control variable can be changed in the body of a `for` *loop, this can lead to subtle errors. It's best not to change it.*

Examples of Using the **`for`** Statement

Examples of Using the **`for`** Statement

- The following examples show methods of varying the control variable in a for statement.

Examples of Using the **`for`** Statement

- The following examples show methods of varying the control variable in a for statement.
  - **`for ( i = 1; i <= 100; i++ )`**

Examples of Using the `for` Statement

- The following examples show methods of varying the control variable in a for statement.
  - **`for ( i = 1; i <= 100; i++ )`**
    - Vary the control variable from 1 to 100 in increments of 1.

Examples of Using the `for` Statement

- The following examples show methods of varying the control variable in a for statement.
  - **`for ( i = 1; i <= 100; i++ )`**
    - Vary the control variable from 1 to 100 in increments of 1.
  - **`for ( i = 100; i >= 1; i-- )`**

Examples of Using the `for` Statement

- The following examples show methods of varying the control variable in a for statement.
  - **`for ( i = 1; i <= 100; i++ )`**
    - Vary the control variable from 1 to 100 in increments of 1.
  - **`for ( i = 100; i >= 1; i-- )`**
    - Vary the control variable from 100 to 1 in increments of -1 (decrements of 1).

Examples of Using the `for` Statement

- The following examples show methods of varying the control variable in a for statement.
- `for ( i = 1; i <= 100; i++ )`
  - Vary the control variable from 1 to 100 in increments of 1.
- `for ( i = 100; i >= 1; i-- )`
  - Vary the control variable from 100 to 1 in increments of -1 (decrements of 1).
- `for ( i = 7; i <= 77; i += 7 )`

Examples of Using the `for` Statement

- The following examples show methods of varying the control variable in a for statement.
  - **`for ( i = 1; i <= 100; i++ )`**
    - Vary the control variable from 1 to 100 in increments of 1.
  - **`for ( i = 100; i >= 1; i-- )`**
    - Vary the control variable from 100 to 1 in increments of -1 (decrements of 1).
  - **`for ( i = 7; i <= 77; i += 7 )`**
    - Vary the control variable from 7 to 77 in steps of 7.

Examples of Using the **`for`** Statement (Cont.)

Examples of Using the `for` Statement (Cont.)

- `for ( i = 20; i >= 2; i -= 2 )`

Examples of Using the `for` Statement (Cont.)

- `for ( i = 20; i >= 2; i -= 2 )`
    - Vary the control variable from 20 to 2 in steps of -2.

Examples of Using the `for` Statement (Cont.)

- `for ( i = 20; i >= 2; i -= 2 )`
  - Vary the control variable from 20 to 2 in steps of -2.
- `for ( j = 2; j <= 17; j += 3 )`

Examples of Using the `for` Statement (Cont.)

- `for ( i = 20; i >= 2; i -= 2 )`
  - Vary the control variable from 20 to 2 in steps of -2.
- `for ( j = 2; j <= 17; j += 3 )`
  - Vary the control variable over the following sequence of values: 2, 5, 8, 11, 14, 17.

Examples of Using the `for` Statement (Cont.)

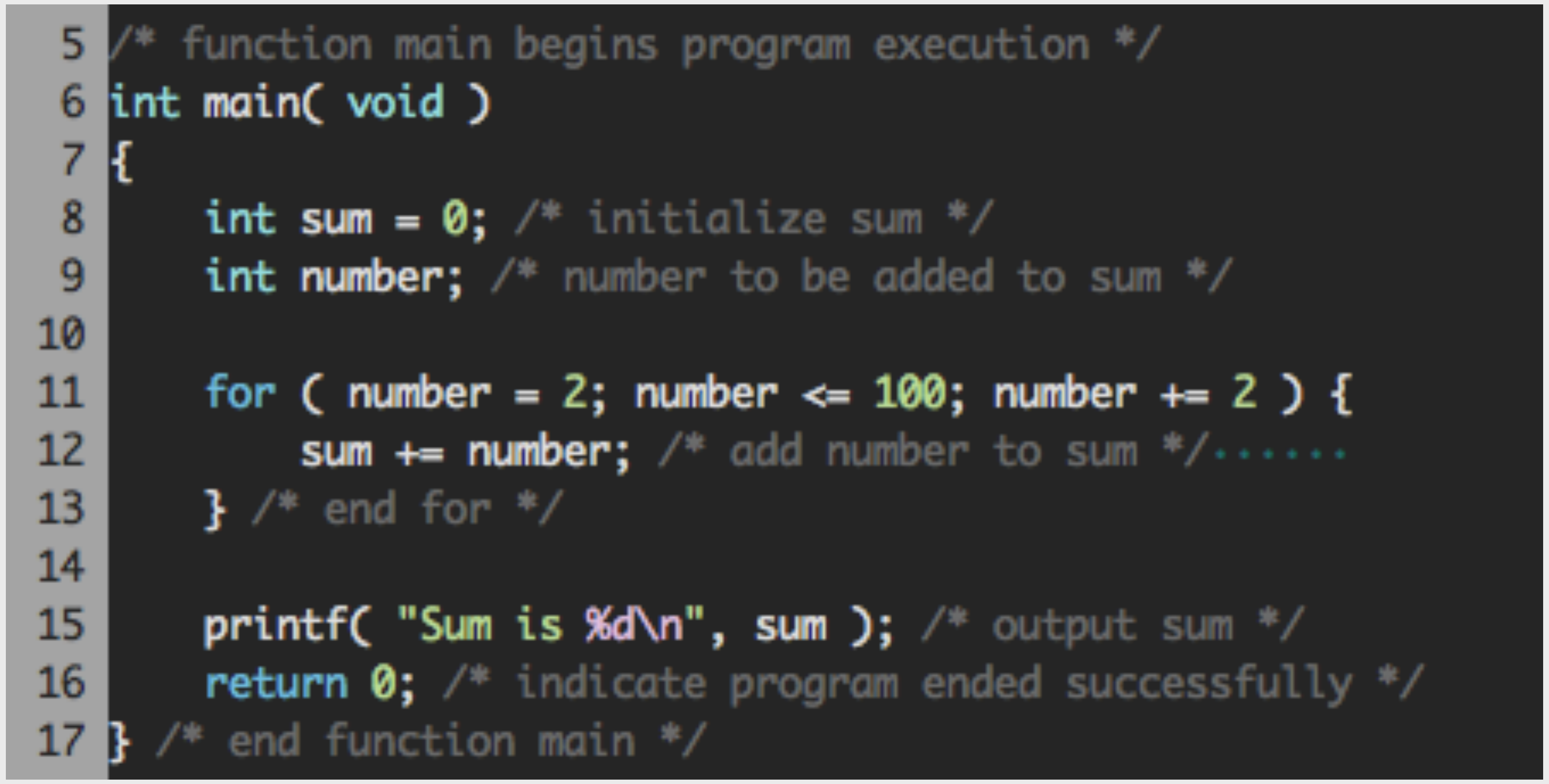
- `for ( i = 20; i >= 2; i -= 2 )`
  - Vary the control variable from 20 to 2 in steps of -2.
- `for ( j = 2; j <= 17; j += 3 )`
  - Vary the control variable over the following sequence of values: 2, 5, 8, 11, 14, 17.
- `for ( j = 44; j >= 0; j -= 11 )`

Examples of Using the `for` Statement (Cont.)

- `for ( i = 20; i >= 2; i -= 2 )`
  - Vary the control variable from 20 to 2 in steps of -2.
- `for ( j = 2; j <= 17; j += 3 )`
  - Vary the control variable over the following sequence of values: 2, 5, 8, 11, 14, 17.
- `for ( j = 44; j >= 0; j -= 11 )`
  - Vary the control variable over the following sequence of values: 44, 33, 22, 11, 0.

`for` Repetition Statement (Cont.)

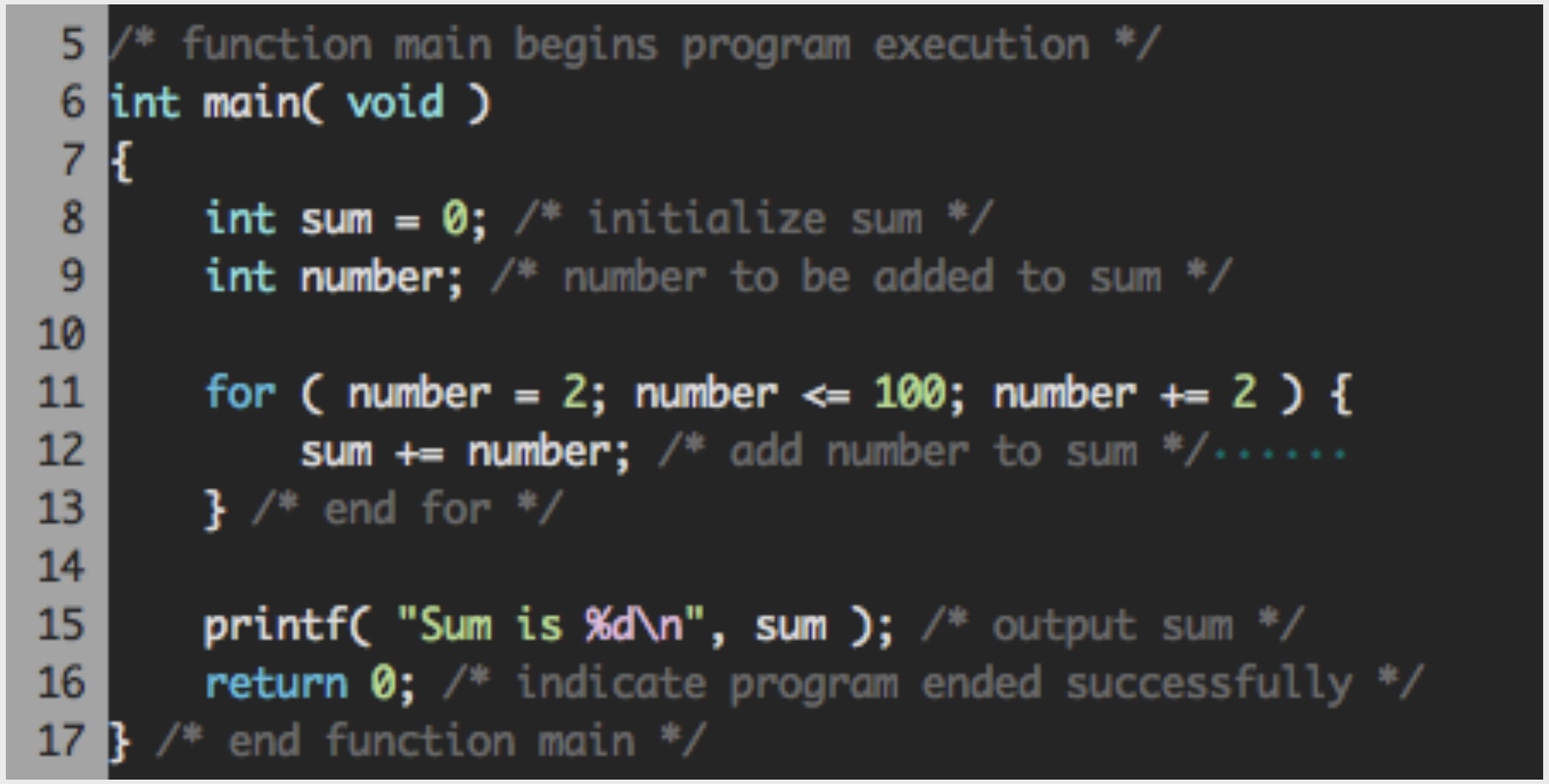
- Example: fig04_05.c



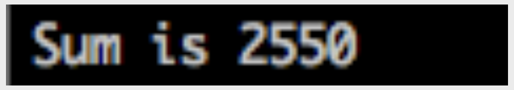
```
/* function main begins program execution */
int main( void )
{
    int sum = 0; /* initialize sum */
    int number; /* number to be added to sum */

    for ( number = 2; number <= 100; number += 2 ) {
        sum += number; /* add number to sum */
    } /* end for */

    printf( "Sum is %d\n", sum ); /* output sum */
    return 0; /* indicate program ended successfully */
} /* end function main */
```

`for` Repetition Statement (Cont.)

- Example: fig04_05.c



```
/* function main begins program execution */
int main( void )
{
    int sum = 0; /* initialize sum */
    int number; /* number to be added to sum */

    for ( number = 2; number <= 100; number += 2 ) {
        sum += number; /* add number to sum */
    } /* end for */

    printf( "Sum is %d\n", sum ); /* output sum */
    return 0; /* indicate program ended successfully */
} /* end function main */
```



```
Sum is 2550
```

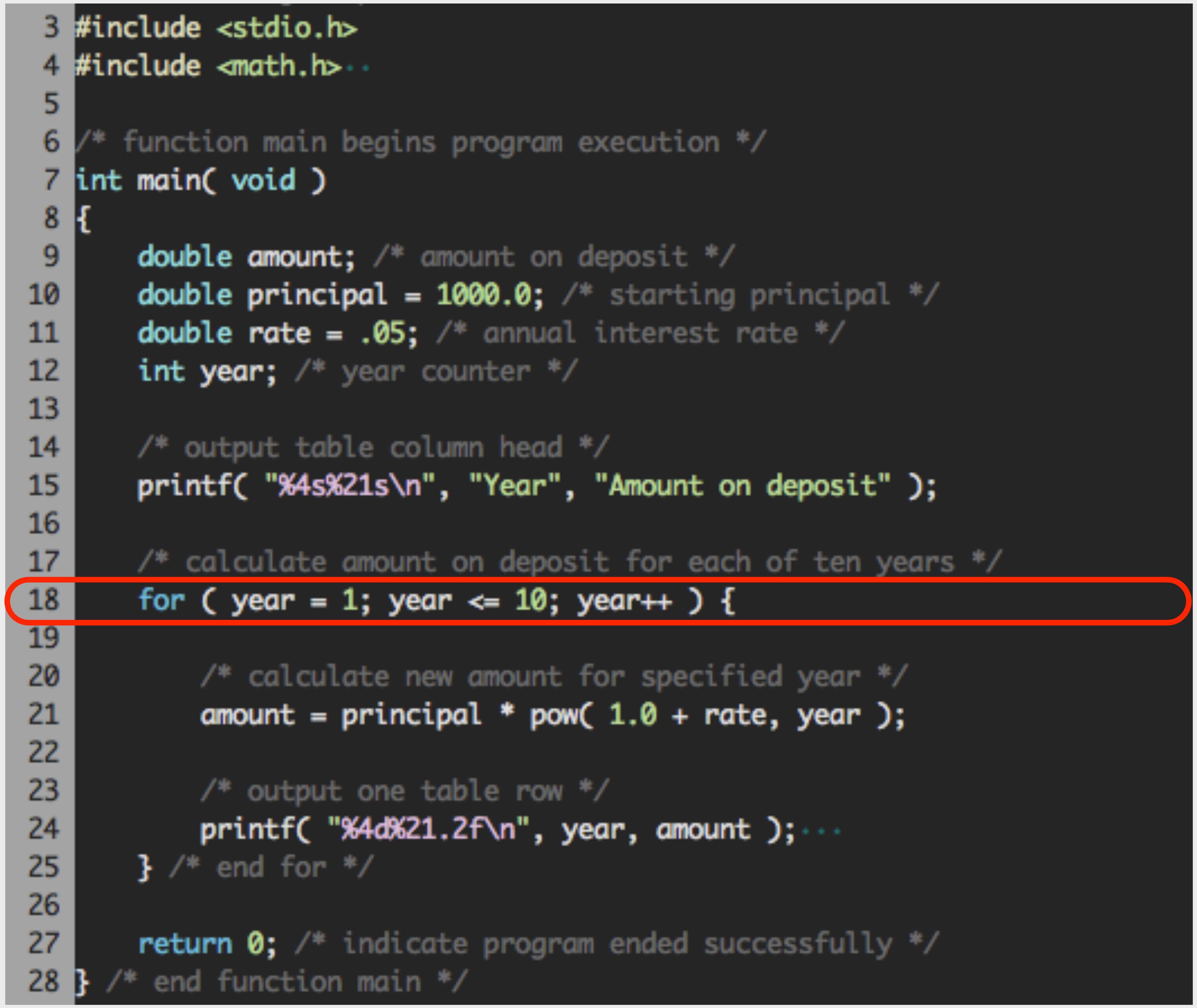
Examples of Using the **`for`** Statement (Cont.)

- Consider the following problem statement:
  - A person invests $1000.00 in a savings account yielding 5% interest. Assuming that all interest is left on deposit in the account, calculate and print the amount of money in the account at the end of each year for 10 years. Use the following formula for determining these amounts:

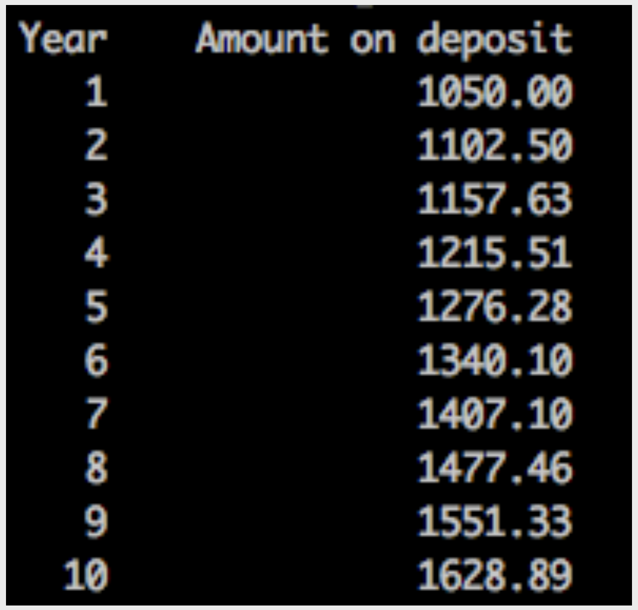
    a = p(1 + r)^n
    where
    p is the original amount invested (i.e., the principal)

    r is the annual interest rate
    n is the number of years
    a is the amount on deposit at the end of the nth year.
- This problem involves a loop that performs the indicated calculation for each of the 10 years the money remains on deposit.

Example: fig04_06.c

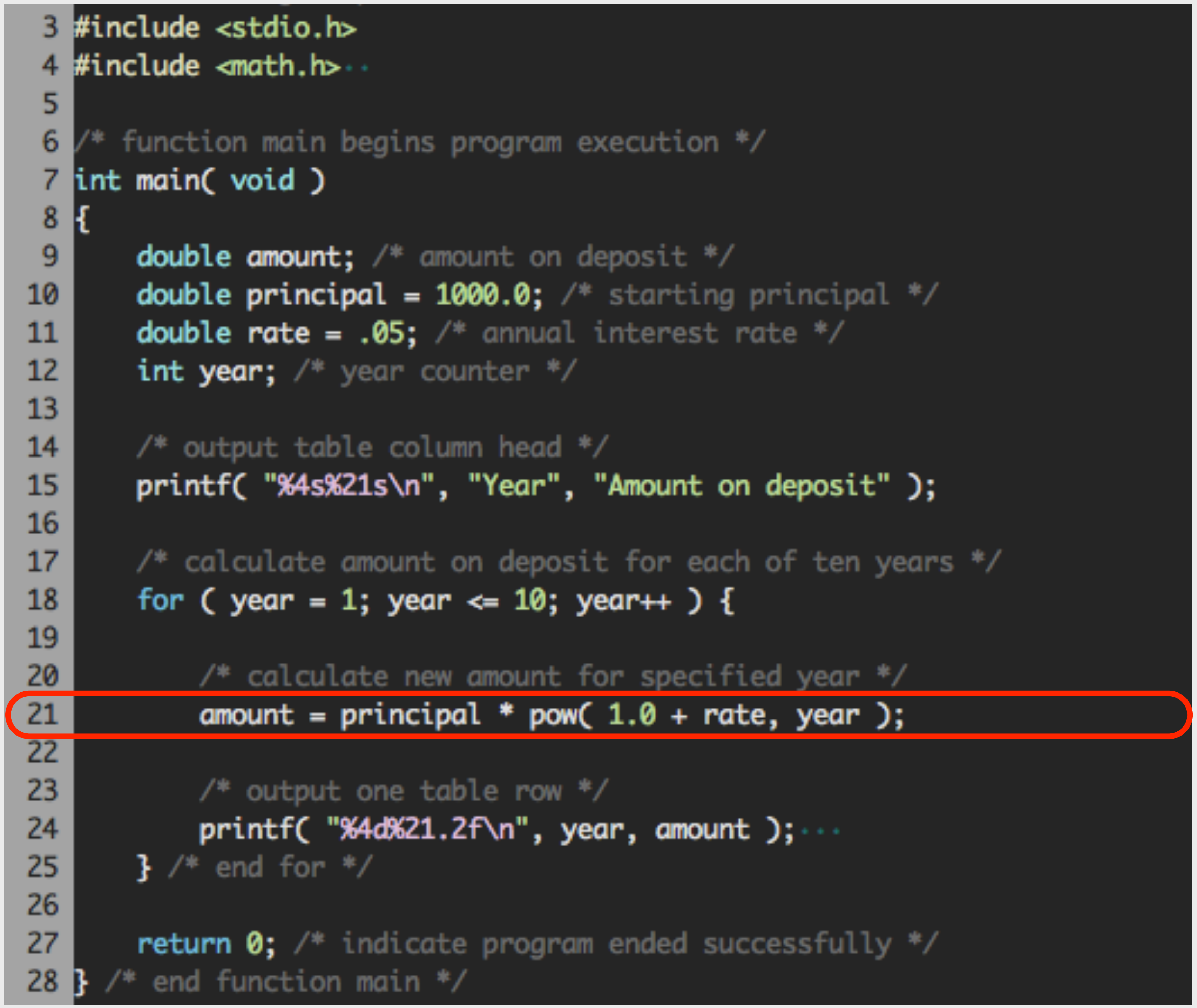


```c
#include <stdio.h>
#include <math.h>

/* function main begins program execution */
int main( void )
{
    double amount; /* amount on deposit */
    double principal = 1000.0; /* starting principal */
    double rate = .05; /* annual interest rate */
    int year; /* year counter */

    /* output table column head */
    printf( "%4s%21s\n", "Year", "Amount on deposit" );

    /* calculate amount on deposit for each of ten years */
    for ( year = 1; year <= 10; year++ ) {

        /* calculate new amount for specified year */
        amount = principal * pow( 1.0 + rate, year );

        /* output one table row */
        printf( "%4d%21.2f\n", year, amount );
    } /* end for */

    return 0; /* indicate program ended successfully */
} /* end function main */
```

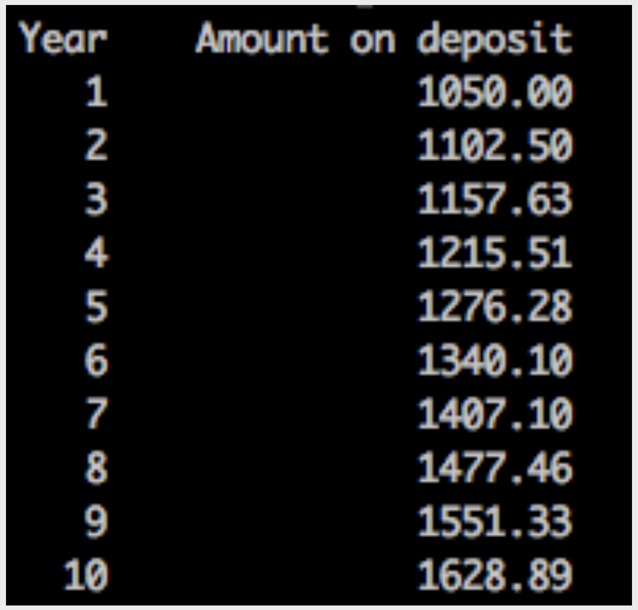


```
Year    Amount on deposit
   1              1050.00
   2              1102.50
   3              1157.63
   4              1215.51
   5              1276.28
   6              1340.10
   7              1407.10
   8              1477.46
   9              1551.33
  10              1628.89
```

Example: fig04_06.c

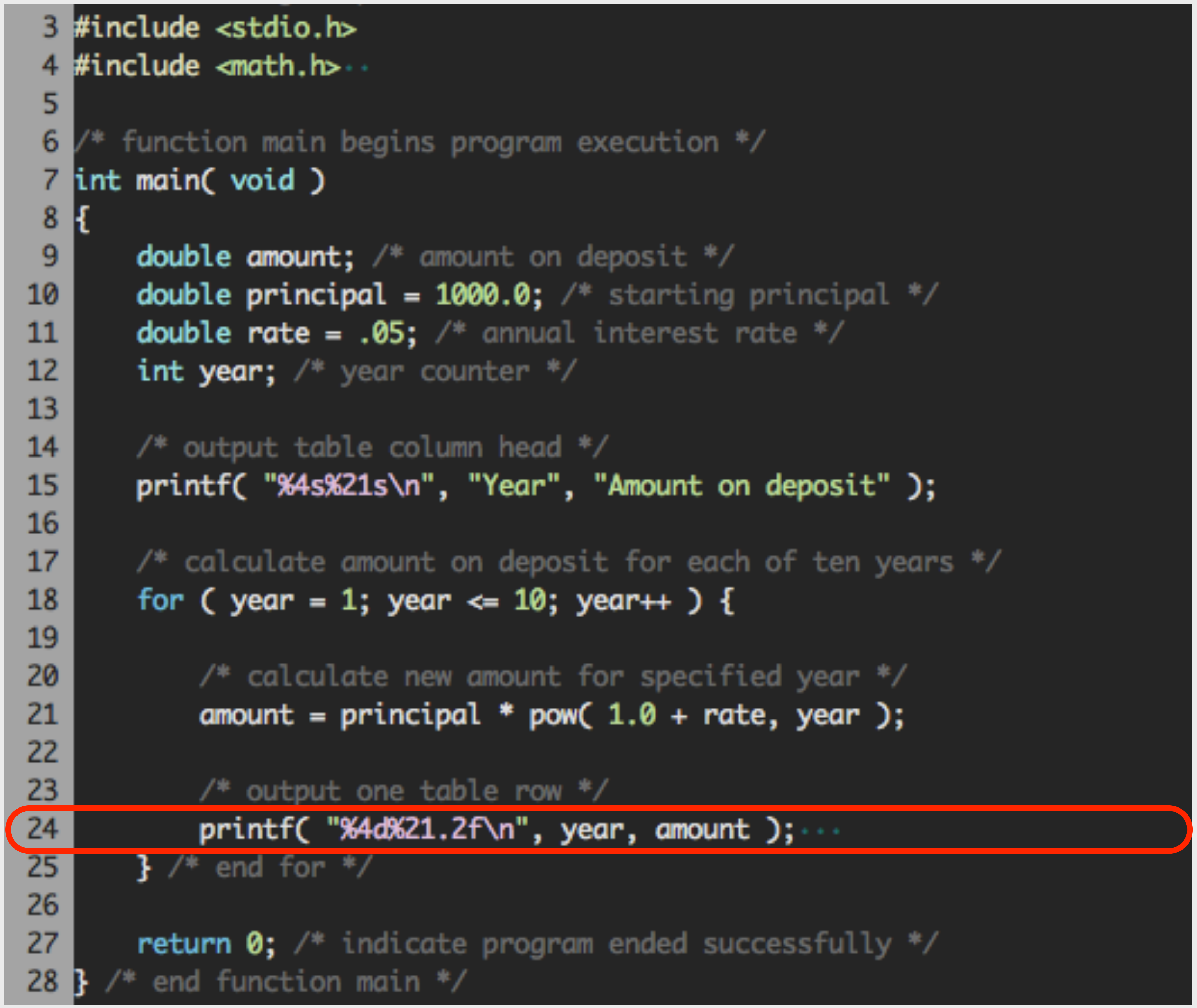


```c
#include <stdio.h>
#include <math.h>

/* function main begins program execution */
int main( void )
{
    double amount; /* amount on deposit */
    double principal = 1000.0; /* starting principal */
    double rate = .05; /* annual interest rate */
    int year; /* year counter */

    /* output table column head */
    printf( "%4s%21s\n", "Year", "Amount on deposit" );

    /* calculate amount on deposit for each of ten years */
    for ( year = 1; year <= 10; year++ ) {

        /* calculate new amount for specified year */
        amount = principal * pow( 1.0 + rate, year );

        /* output one table row */
        printf( "%4d%21.2f\n", year, amount );
    } /* end for */

    return 0; /* indicate program ended successfully */
} /* end function main */
```

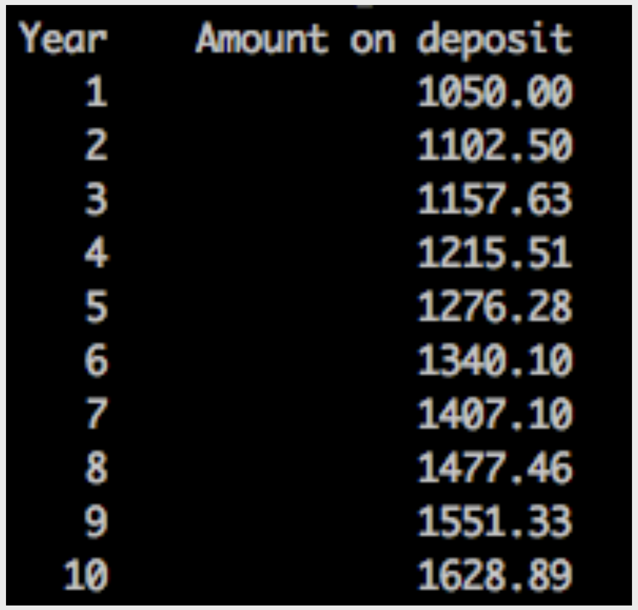


```
Year    Amount on deposit
   1              1050.00
   2              1102.50
   3              1157.63
   4              1215.51
   5              1276.28
   6              1340.10
   7              1407.10
   8              1477.46
   9              1551.33
  10              1628.89
```

Example: fig04_06.c

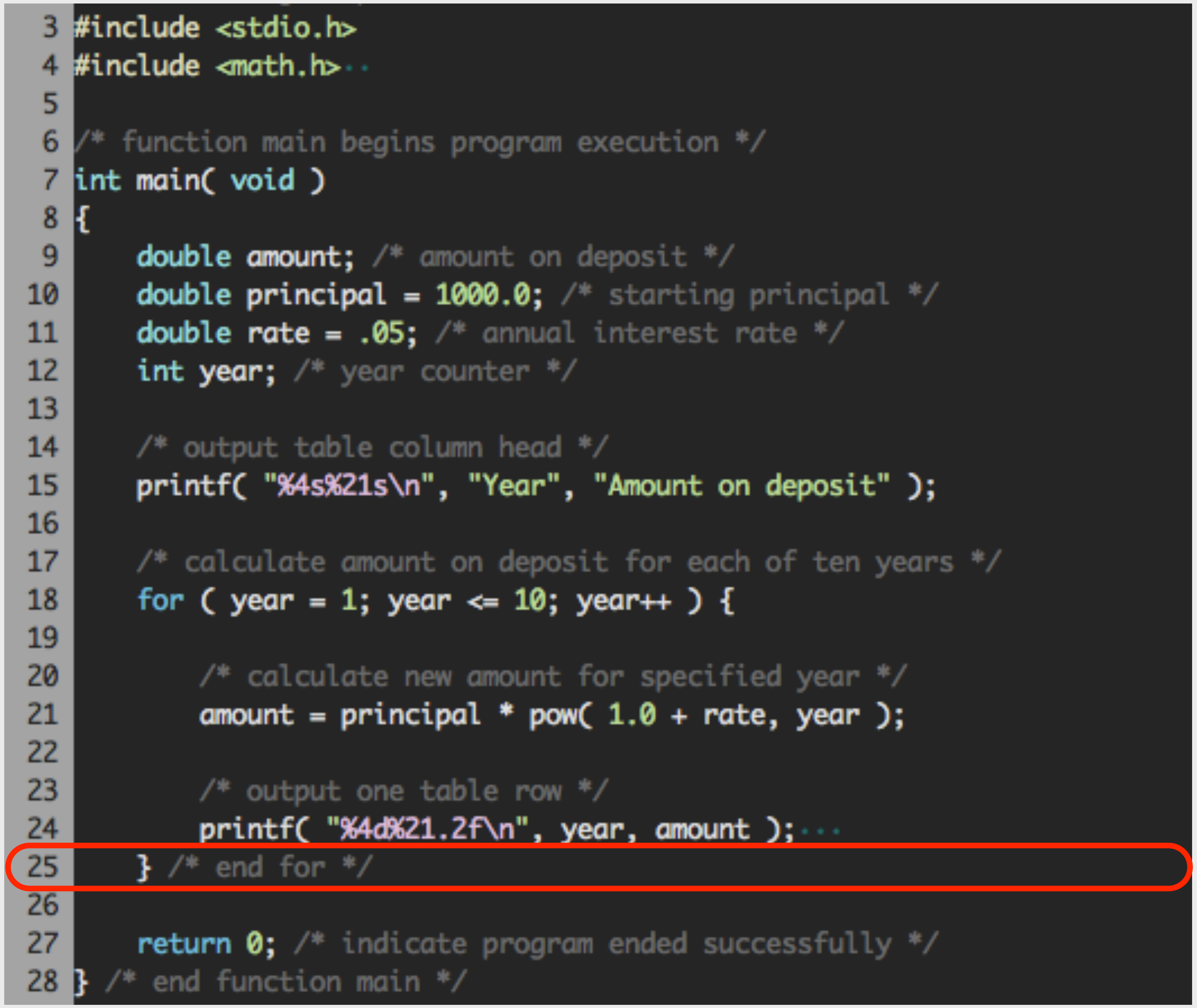


```c
#include <stdio.h>
#include <math.h>

/* function main begins program execution */
int main( void )
{
    double amount; /* amount on deposit */
    double principal = 1000.0; /* starting principal */
    double rate = .05; /* annual interest rate */
    int year; /* year counter */

    /* output table column head */
    printf( "%4s%21s\n", "Year", "Amount on deposit" );

    /* calculate amount on deposit for each of ten years */
    for ( year = 1; year <= 10; year++ ) {

        /* calculate new amount for specified year */
        amount = principal * pow( 1.0 + rate, year );

        /* output one table row */
        printf( "%4d%21.2f\n", year, amount );
    } /* end for */

    return 0; /* indicate program ended successfully */
} /* end function main */
```

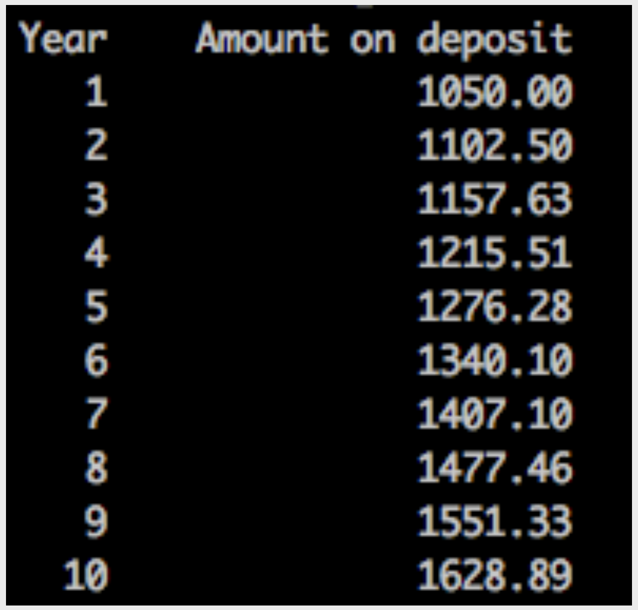


```
Year    Amount on deposit
   1              1050.00
   2              1102.50
   3              1157.63
   4              1215.51
   5              1276.28
   6              1340.10
   7              1407.10
   8              1477.46
   9              1551.33
  10              1628.89
```

Example: fig04_06.c

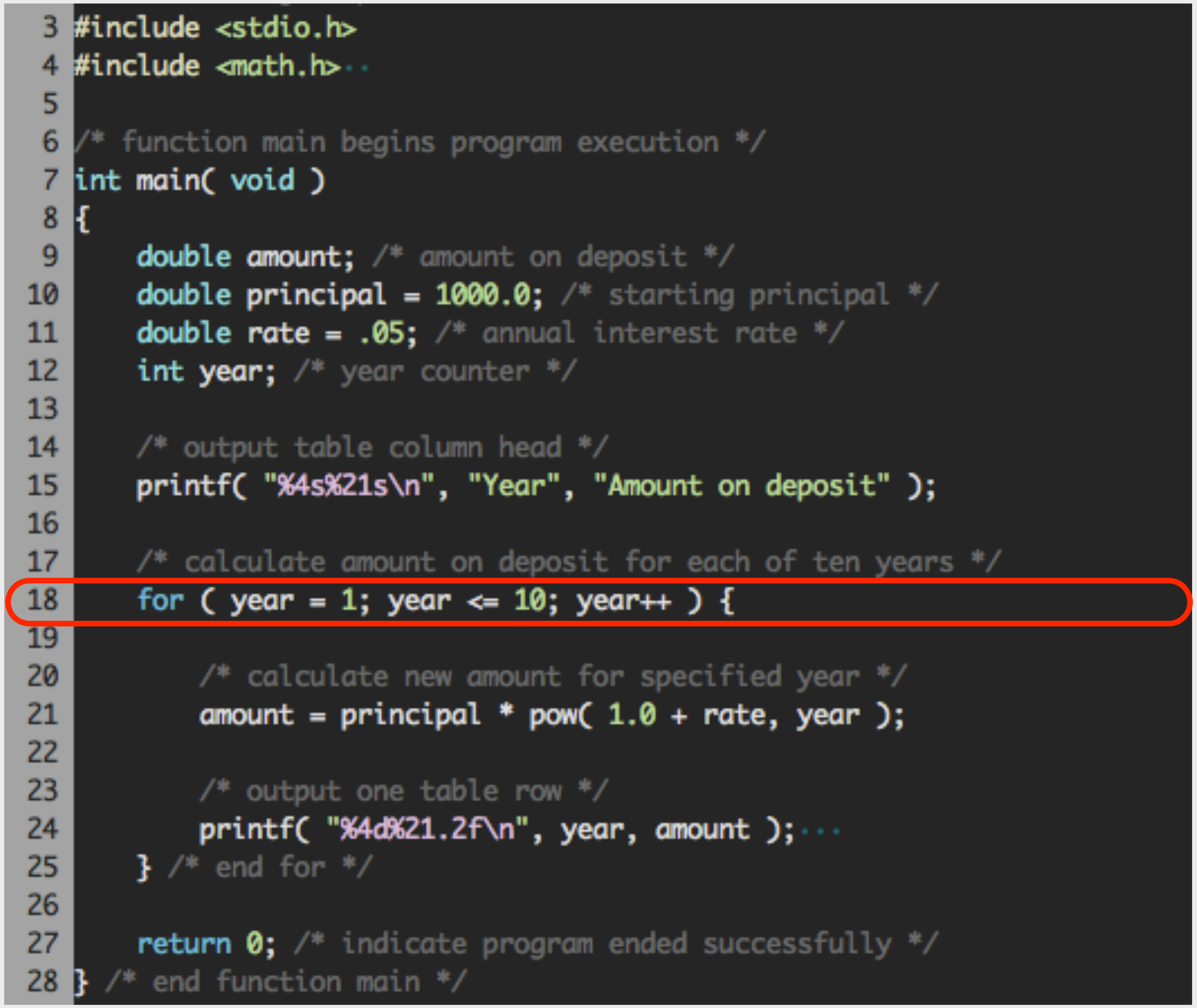


```c
#include <stdio.h>
#include <math.h>

/* function main begins program execution */
int main( void )
{
    double amount; /* amount on deposit */
    double principal = 1000.0; /* starting principal */
    double rate = .05; /* annual interest rate */
    int year; /* year counter */

    /* output table column head */
    printf( "%4s%21s\n", "Year", "Amount on deposit" );

    /* calculate amount on deposit for each of ten years */
    for ( year = 1; year <= 10; year++ ) {

        /* calculate new amount for specified year */
        amount = principal * pow( 1.0 + rate, year );

        /* output one table row */
        printf( "%4d%21.2f\n", year, amount );
    } /* end for */

    return 0; /* indicate program ended successfully */
} /* end function main */
```

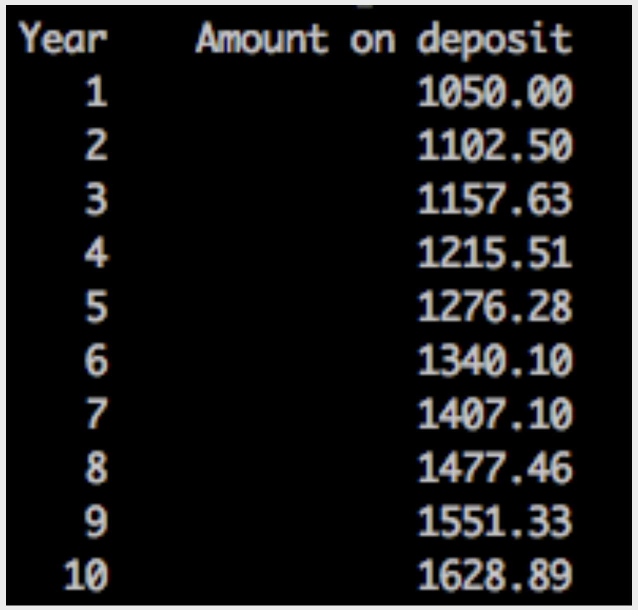


```
Year    Amount on deposit
   1              1050.00
   2              1102.50
   3              1157.63
   4              1215.51
   5              1276.28
   6              1340.10
   7              1407.10
   8              1477.46
   9              1551.33
  10              1628.89
```

Example: fig04_06.c

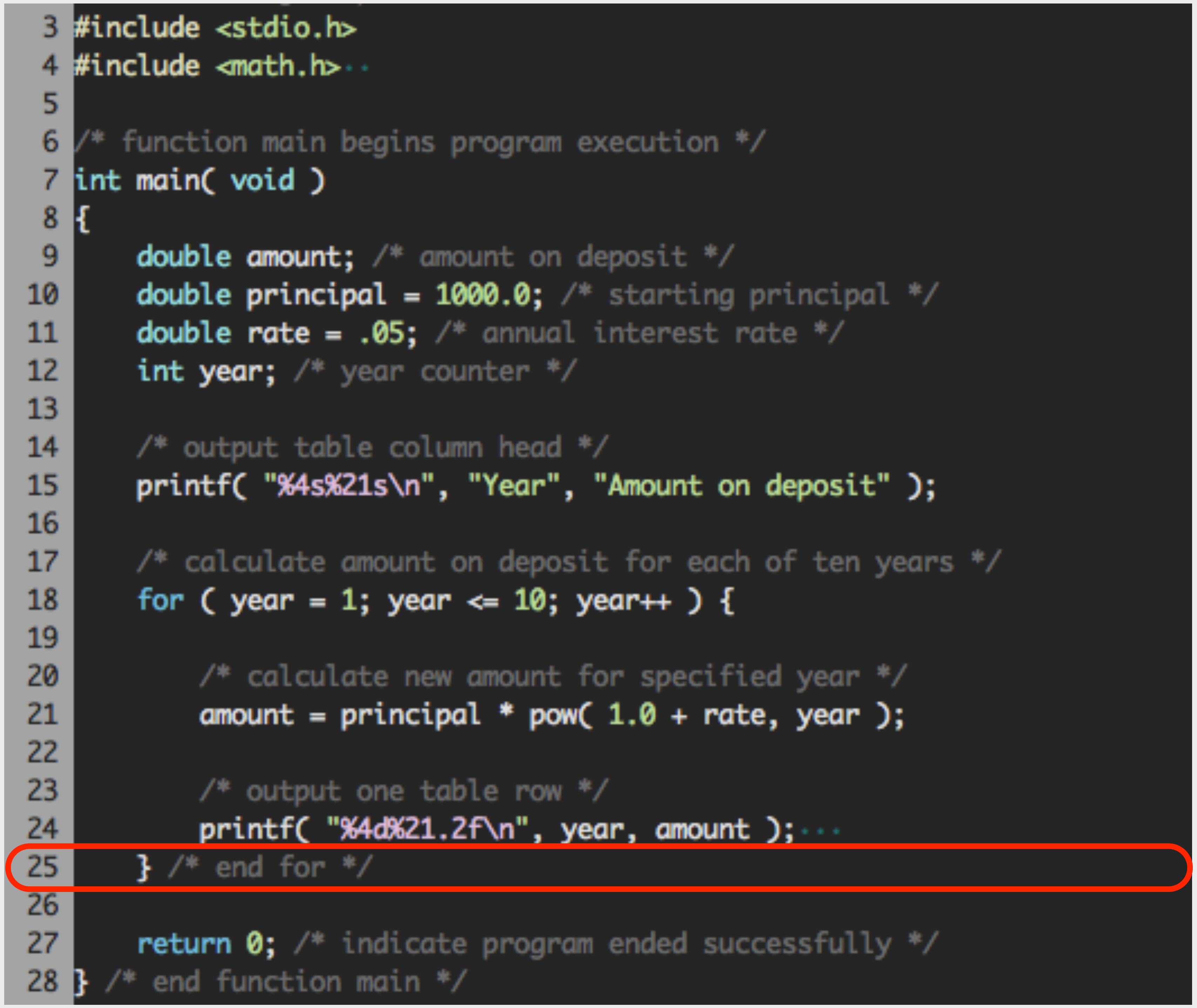


```c
#include <stdio.h>
#include <math.h>

/* function main begins program execution */
int main( void )
{
    double amount; /* amount on deposit */
    double principal = 1000.0; /* starting principal */
    double rate = .05; /* annual interest rate */
    int year; /* year counter */

    /* output table column head */
    printf( "%4s%21s\n", "Year", "Amount on deposit" );

    /* calculate amount on deposit for each of ten years */
    for ( year = 1; year <= 10; year++ ) {

        /* calculate new amount for specified year */
        amount = principal * pow( 1.0 + rate, year );

        /* output one table row */
        printf( "%4d%21.2f\n", year, amount );
    } /* end for */

    return 0; /* indicate program ended successfully */
} /* end function main */
```

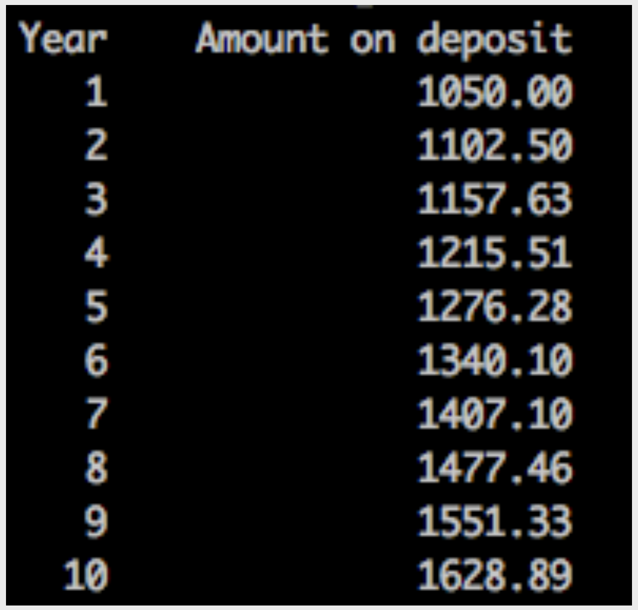


```
Year    Amount on deposit
   1              1050.00
   2              1102.50
   3              1157.63
   4              1215.51
   5              1276.28
   6              1340.10
   7              1407.10
   8              1477.46
   9              1551.33
  10              1628.89
```

Example: fig04_06.c

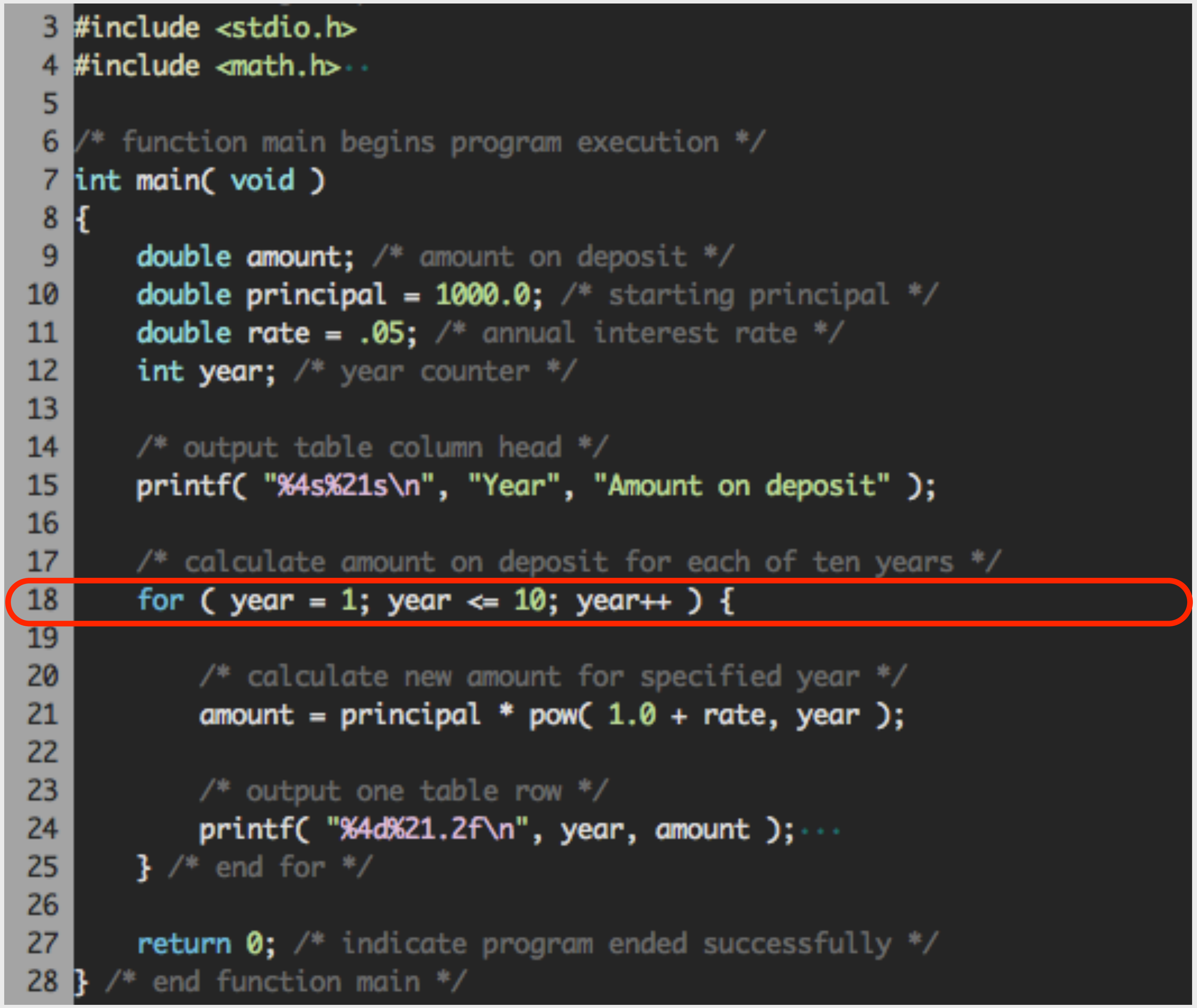


```c
#include <stdio.h>
#include <math.h>

/* function main begins program execution */
int main( void )
{
    double amount; /* amount on deposit */
    double principal = 1000.0; /* starting principal */
    double rate = .05; /* annual interest rate */
    int year; /* year counter */

    /* output table column head */
    printf( "%4s%21s\n", "Year", "Amount on deposit" );

    /* calculate amount on deposit for each of ten years */
    for ( year = 1; year <= 10; year++ ) {

        /* calculate new amount for specified year */
        amount = principal * pow( 1.0 + rate, year );

        /* output one table row */
        printf( "%4d%21.2f\n", year, amount );
    } /* end for */

    return 0; /* indicate program ended successfully */
} /* end function main */
```

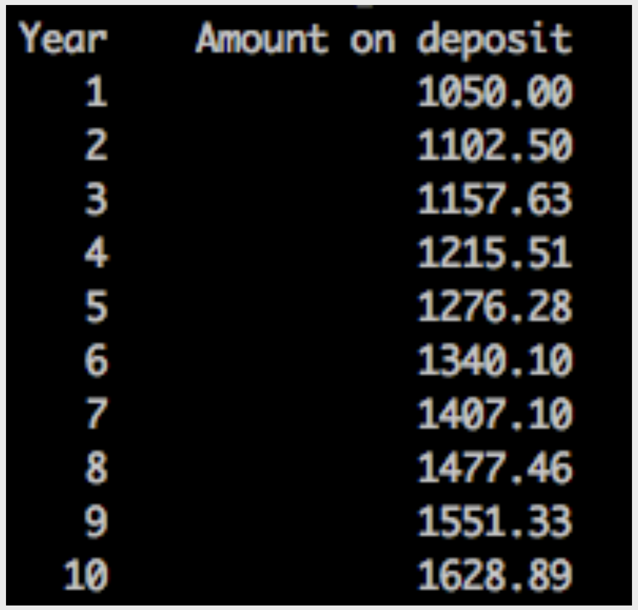


```
Year    Amount on deposit
   1              1050.00
   2              1102.50
   3              1157.63
   4              1215.51
   5              1276.28
   6              1340.10
   7              1407.10
   8              1477.46
   9              1551.33
  10              1628.89
```

Example: fig04_06.c

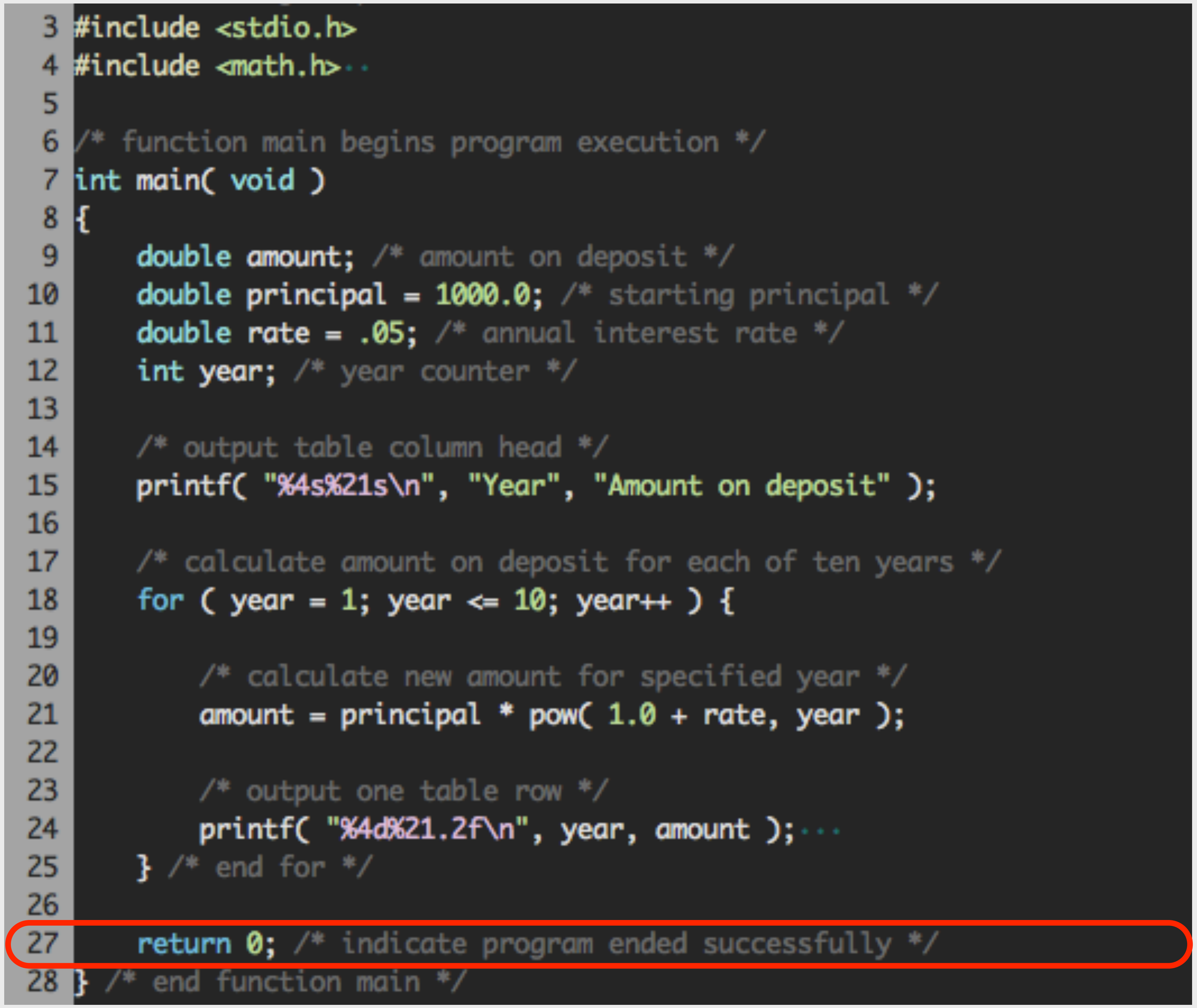


```c
#include <stdio.h>
#include <math.h>

/* function main begins program execution */
int main( void )
{
    double amount; /* amount on deposit */
    double principal = 1000.0; /* starting principal */
    double rate = .05; /* annual interest rate */
    int year; /* year counter */

    /* output table column head */
    printf( "%4s%21s\n", "Year", "Amount on deposit" );

    /* calculate amount on deposit for each of ten years */
    for ( year = 1; year <= 10; year++ ) {

        /* calculate new amount for specified year */
        amount = principal * pow( 1.0 + rate, year );

        /* output one table row */
        printf( "%4d%21.2f\n", year, amount );
    } /* end for */

    return 0; /* indicate program ended successfully */
} /* end function main */
```

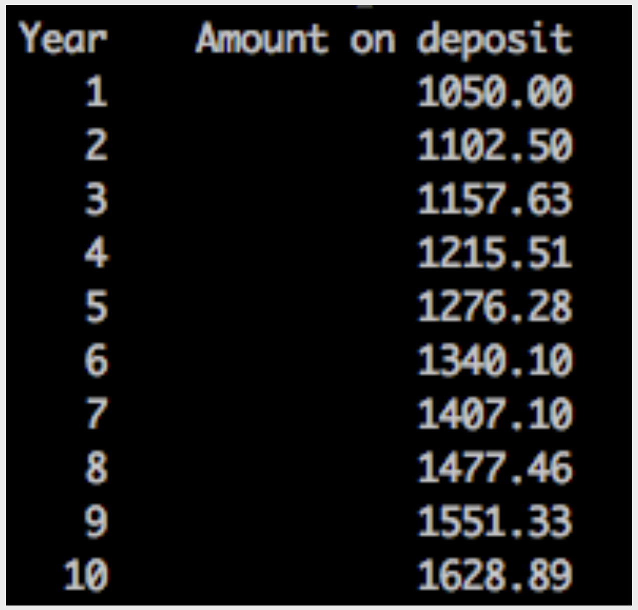


```
Year    Amount on deposit
   1              1050.00
   2              1102.50
   3              1157.63
   4              1215.51
   5              1276.28
   6              1340.10
   7              1407.10
   8              1477.46
   9              1551.33
  10              1628.89
```

Example: fig04_06.c



```c
#include <stdio.h>
#include <math.h>

/* function main begins program execution */
int main( void )
{
    double amount; /* amount on deposit */
    double principal = 1000.0; /* starting principal */
    double rate = .05; /* annual interest rate */
    int year; /* year counter */

    /* output table column head */
    printf( "%4s%21s\n", "Year", "Amount on deposit" );

    /* calculate amount on deposit for each of ten years */
    for ( year = 1; year <= 10; year++ ) {

        /* calculate new amount for specified year */
        amount = principal * pow( 1.0 + rate, year );

        /* output one table row */
        printf( "%4d%21.2f\n", year, amount );
    } /* end for */

    return 0; /* indicate program ended successfully */
} /* end function main */
```

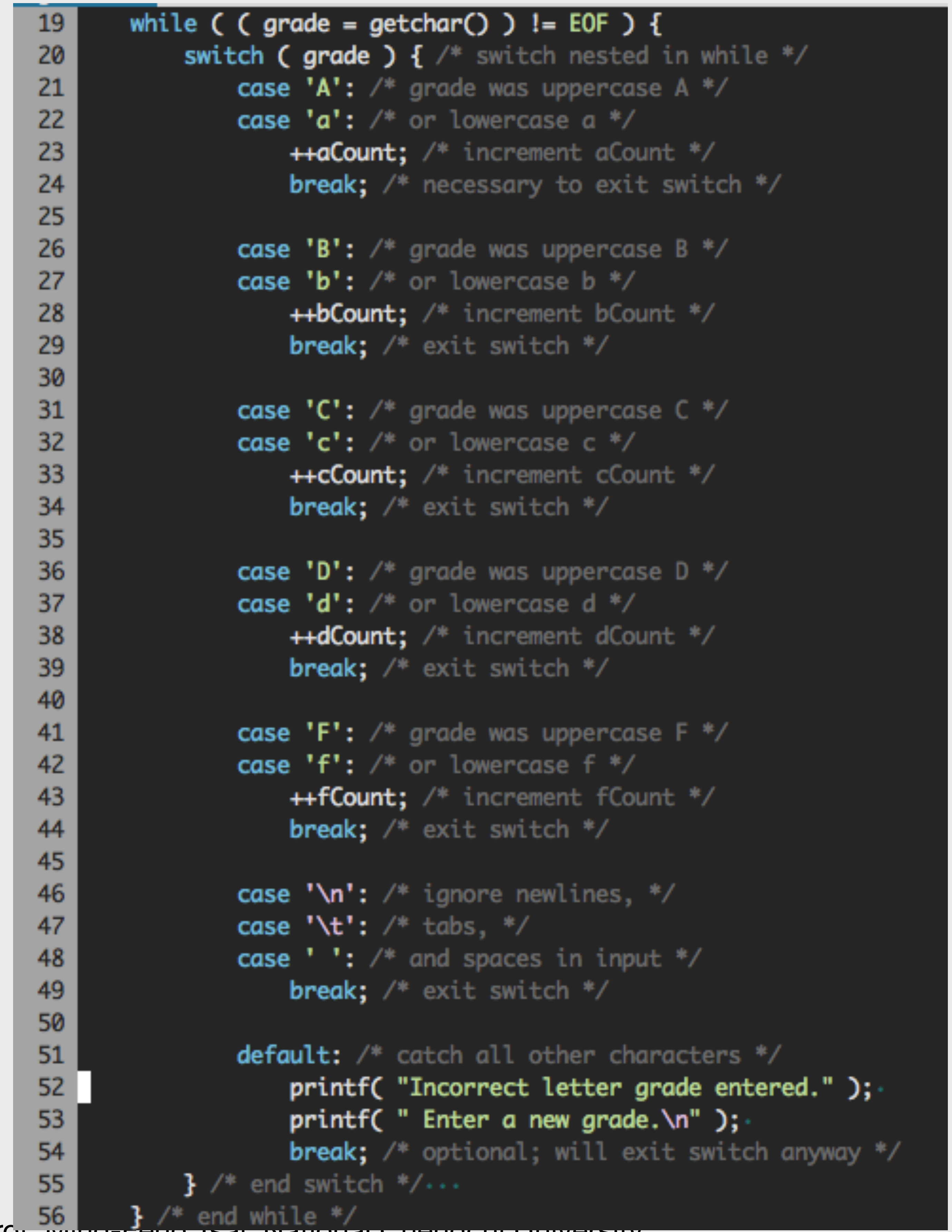


```
Year    Amount on deposit
   1              1050.00
   2              1102.50
   3              1157.63
   4              1215.51
   5              1276.28
   6              1340.10
   7              1407.10
   8              1477.46
   9              1551.33
  10              1628.89
```

`switch` Multiple-Selection Statement

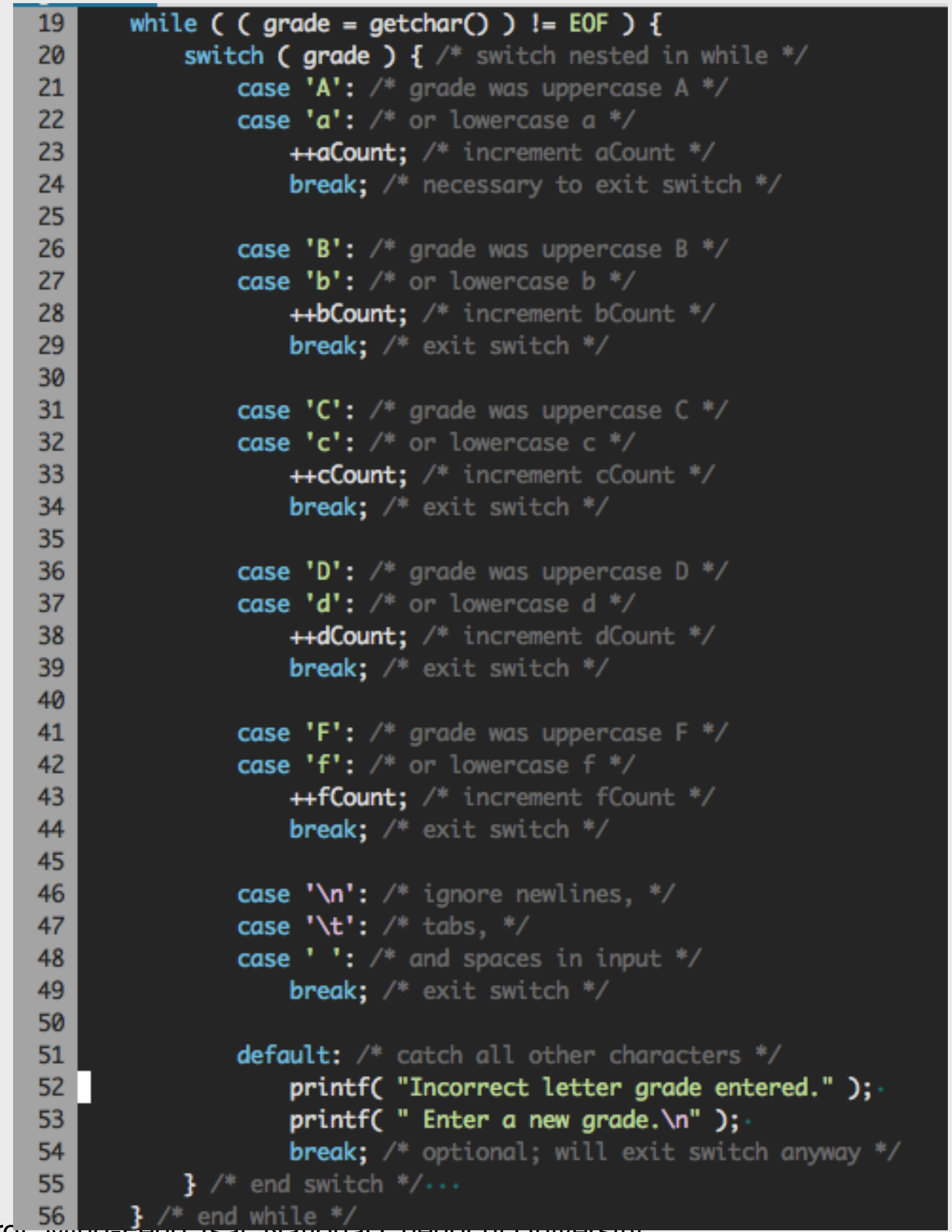
- Occasionally, an algorithm will contain a series of decisions in which a variable or expression is tested separately for each of the constant integral values it may assume, and different actions are taken.
- This is called multiple selection.
- C provides the switch multiple-selection statement to handle such decision making.
- The switch statement consists of a series of case labels, an optional default case and statements to execute for each case.

Example: fig04_07.c

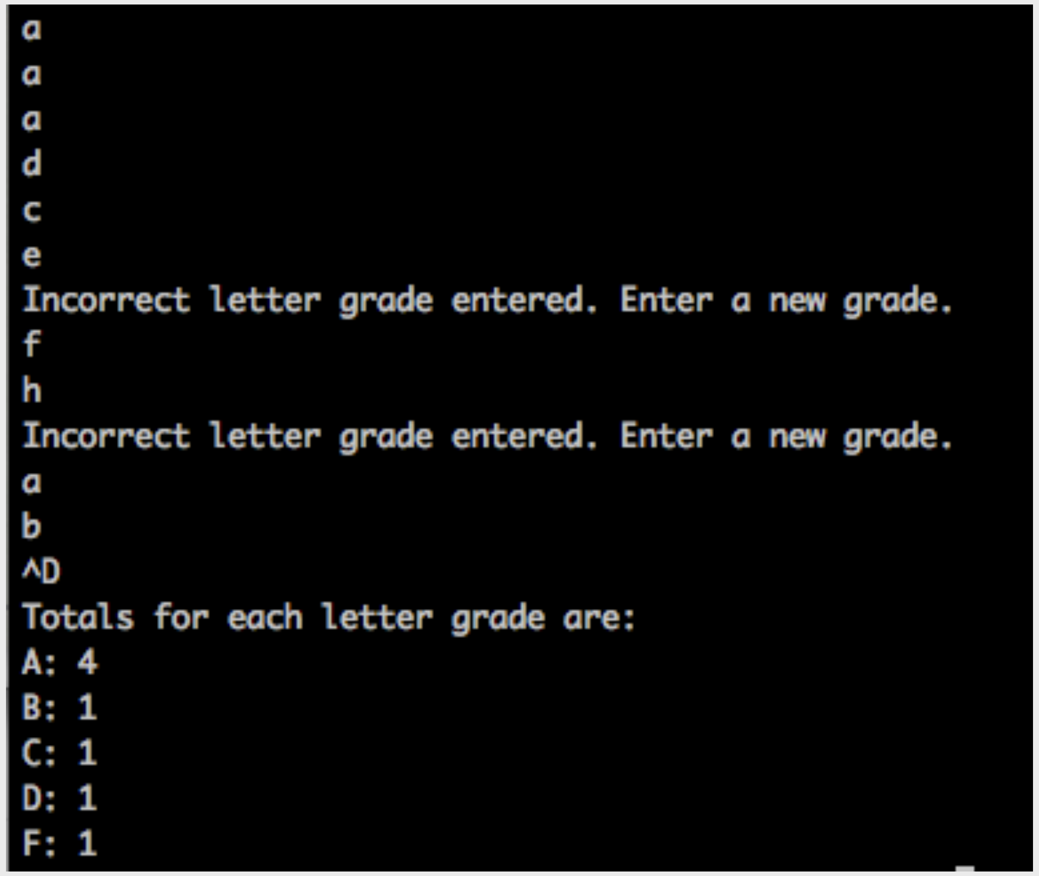


```
while ( ( grade = getchar() ) != EOF ) {
    switch ( grade ) { /* switch nested in while */
        case 'A': /* grade was uppercase A */
        case 'a': /* or lowercase a */
            ++aCount; /* increment aCount */
            break; /* necessary to exit switch */

        case 'B': /* grade was uppercase B */
        case 'b': /* or lowercase b */
            ++bCount; /* increment bCount */
            break; /* exit switch */

        case 'C': /* grade was uppercase C */
        case 'c': /* or lowercase c */
            ++cCount; /* increment cCount */
            break; /* exit switch */

        case 'D': /* grade was uppercase D */
        case 'd': /* or lowercase d */
            ++dCount; /* increment dCount */
            break; /* exit switch */

        case 'F': /* grade was uppercase F */
        case 'f': /* or lowercase f */
            ++fCount; /* increment fCount */
            break; /* exit switch */

        case '\n': /* ignore newlines, */
        case '\t': /* tabs, */
        case ' ': /* and spaces in input */
            break; /* exit switch */

        default: /* catch all other characters */
            printf( "Incorrect letter grade entered." );
            printf( " Enter a new grade.\n" );
            break; /* optional; will exit switch anyway */
    } /* end switch */
} /* end while */
```

Example: fig04_07.c

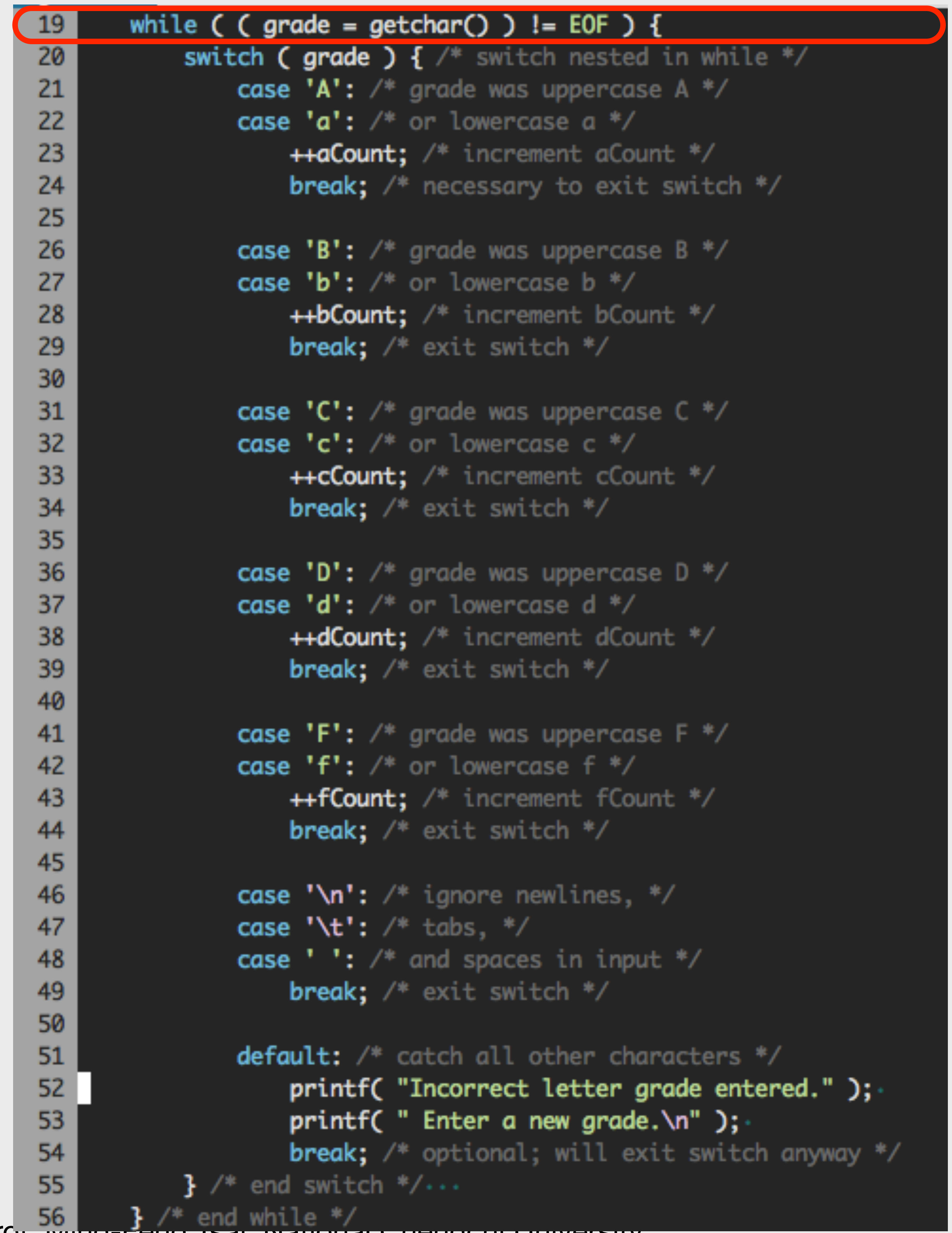


```c
    while ( ( grade = getchar() ) != EOF ) {
        switch ( grade ) { /* switch nested in while */
            case 'A': /* grade was uppercase A */
            case 'a': /* or lowercase a */
                ++aCount; /* increment aCount */
                break; /* necessary to exit switch */

            case 'B': /* grade was uppercase B */
            case 'b': /* or lowercase b */
                ++bCount; /* increment bCount */
                break; /* exit switch */

            case 'C': /* grade was uppercase C */
            case 'c': /* or lowercase c */
                ++cCount; /* increment cCount */
                break; /* exit switch */

            case 'D': /* grade was uppercase D */
            case 'd': /* or lowercase d */
                ++dCount; /* increment dCount */
                break; /* exit switch */

            case 'F': /* grade was uppercase F */
            case 'f': /* or lowercase f */
                ++fCount; /* increment fCount */
                break; /* exit switch */

            case '\n': /* ignore newlines, */
            case '\t': /* tabs, */
            case ' ': /* and spaces in input */
                break; /* exit switch */

            default: /* catch all other characters */
                printf( "Incorrect letter grade entered." );
                printf( " Enter a new grade.\n" );
                break; /* optional; will exit switch anyway */
        } /* end switch */
    } /* end while */
```

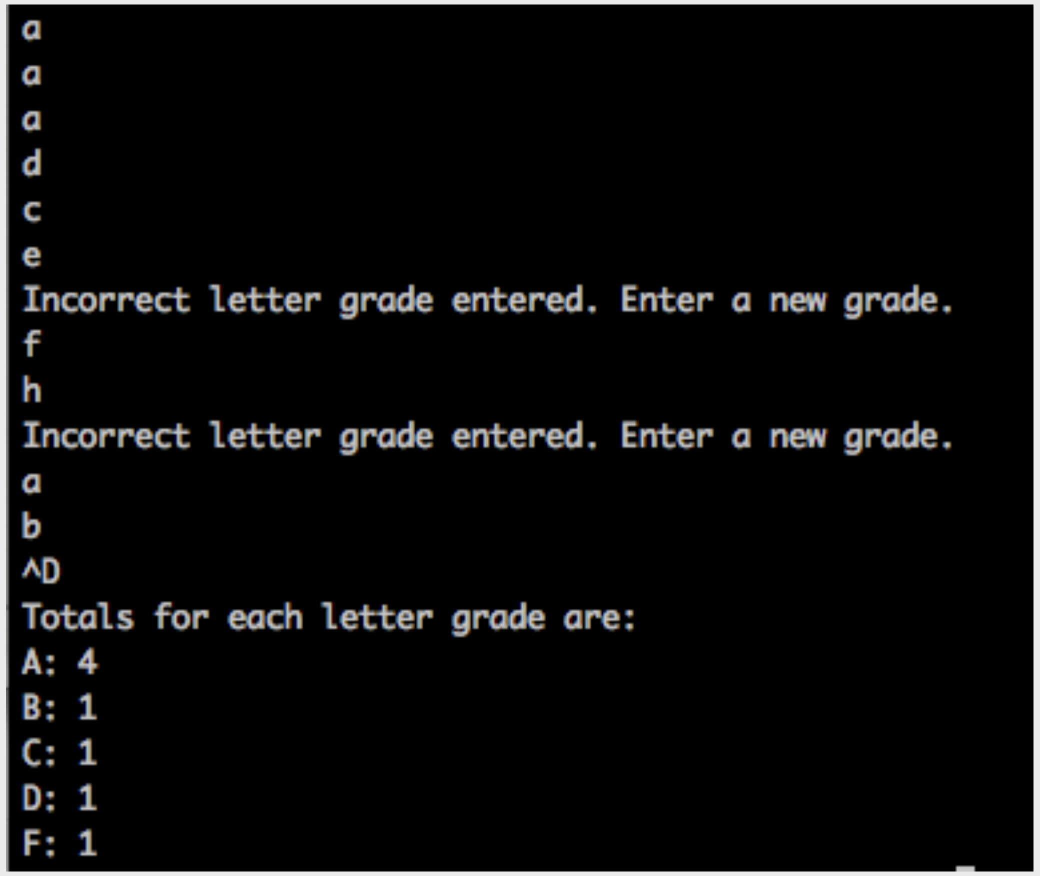


```
a
a
a
d
c
e
Incorrect letter grade entered. Enter a new grade.
f
h
Incorrect letter grade entered. Enter a new grade.
a
b
^D
Totals for each letter grade are:
A: 4
B: 1
C: 1
D: 1
F: 1
```

Example: fig04_07.c

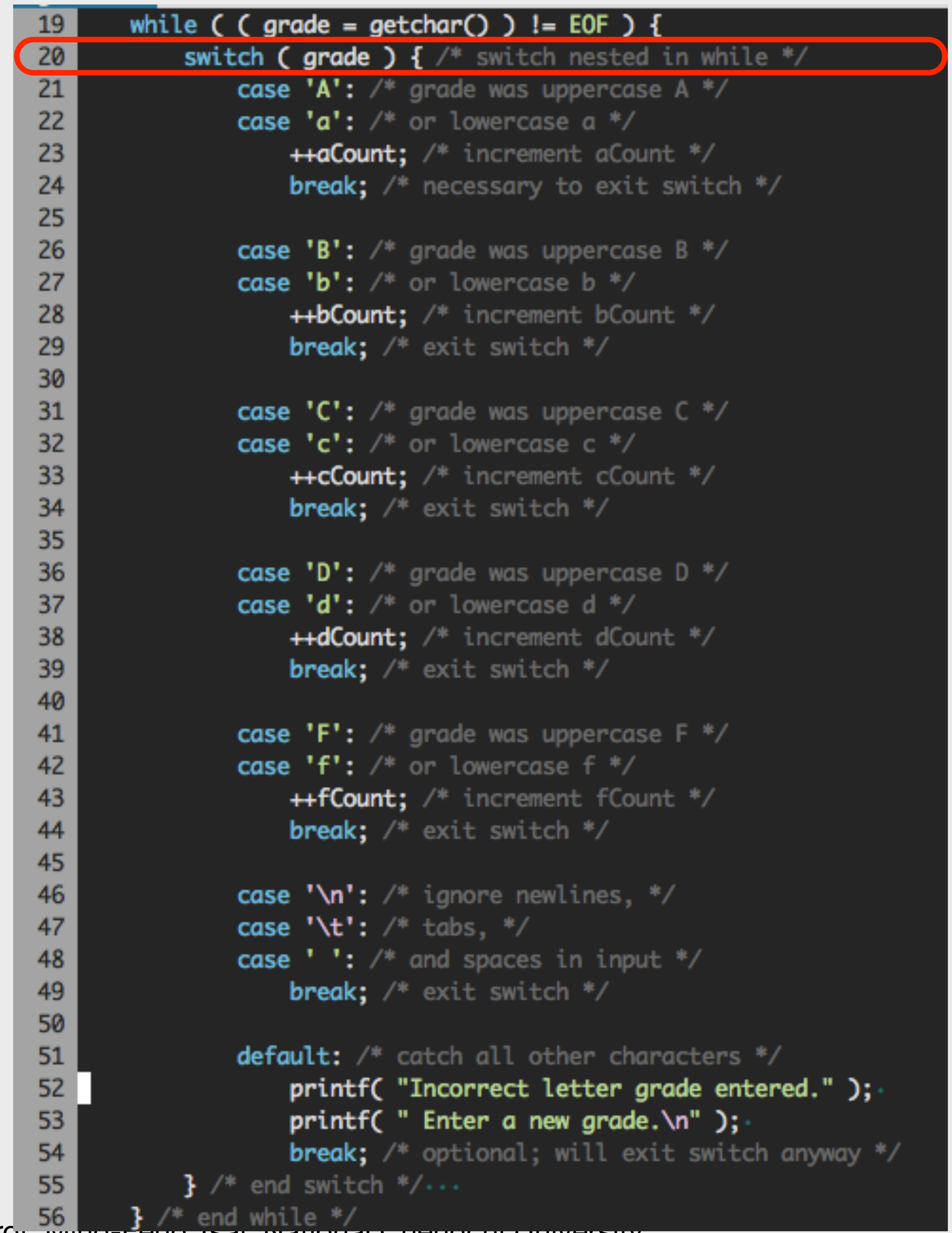


```c
    while ( ( grade = getchar() ) != EOF ) {
        switch ( grade ) { /* switch nested in while */
            case 'A': /* grade was uppercase A */
            case 'a': /* or lowercase a */
                ++aCount; /* increment aCount */
                break; /* necessary to exit switch */

            case 'B': /* grade was uppercase B */
            case 'b': /* or lowercase b */
                ++bCount; /* increment bCount */
                break; /* exit switch */

            case 'C': /* grade was uppercase C */
            case 'c': /* or lowercase c */
                ++cCount; /* increment cCount */
                break; /* exit switch */

            case 'D': /* grade was uppercase D */
            case 'd': /* or lowercase d */
                ++dCount; /* increment dCount */
                break; /* exit switch */

            case 'F': /* grade was uppercase F */
            case 'f': /* or lowercase f */
                ++fCount; /* increment fCount */
                break; /* exit switch */

            case '\n': /* ignore newlines, */
            case '\t': /* tabs, */
            case ' ': /* and spaces in input */
                break; /* exit switch */

            default: /* catch all other characters */
                printf( "Incorrect letter grade entered." );
                printf( " Enter a new grade.\n" );
                break; /* optional; will exit switch anyway */
        } /* end switch */
    } /* end while */
```

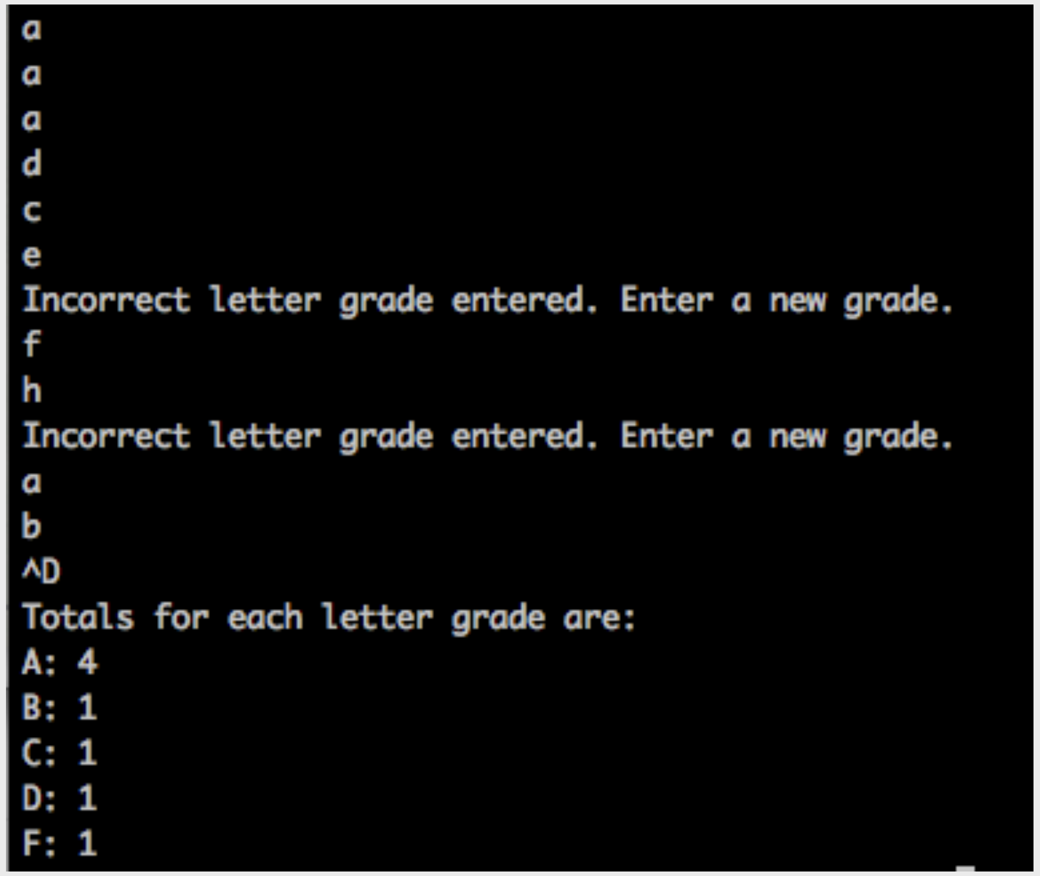


```
a
a
a
d
c
e
Incorrect letter grade entered. Enter a new grade.
f
h
Incorrect letter grade entered. Enter a new grade.
a
b
^D
Totals for each letter grade are:
A: 4
B: 1
C: 1
D: 1
F: 1
```

Example: fig04_07.c

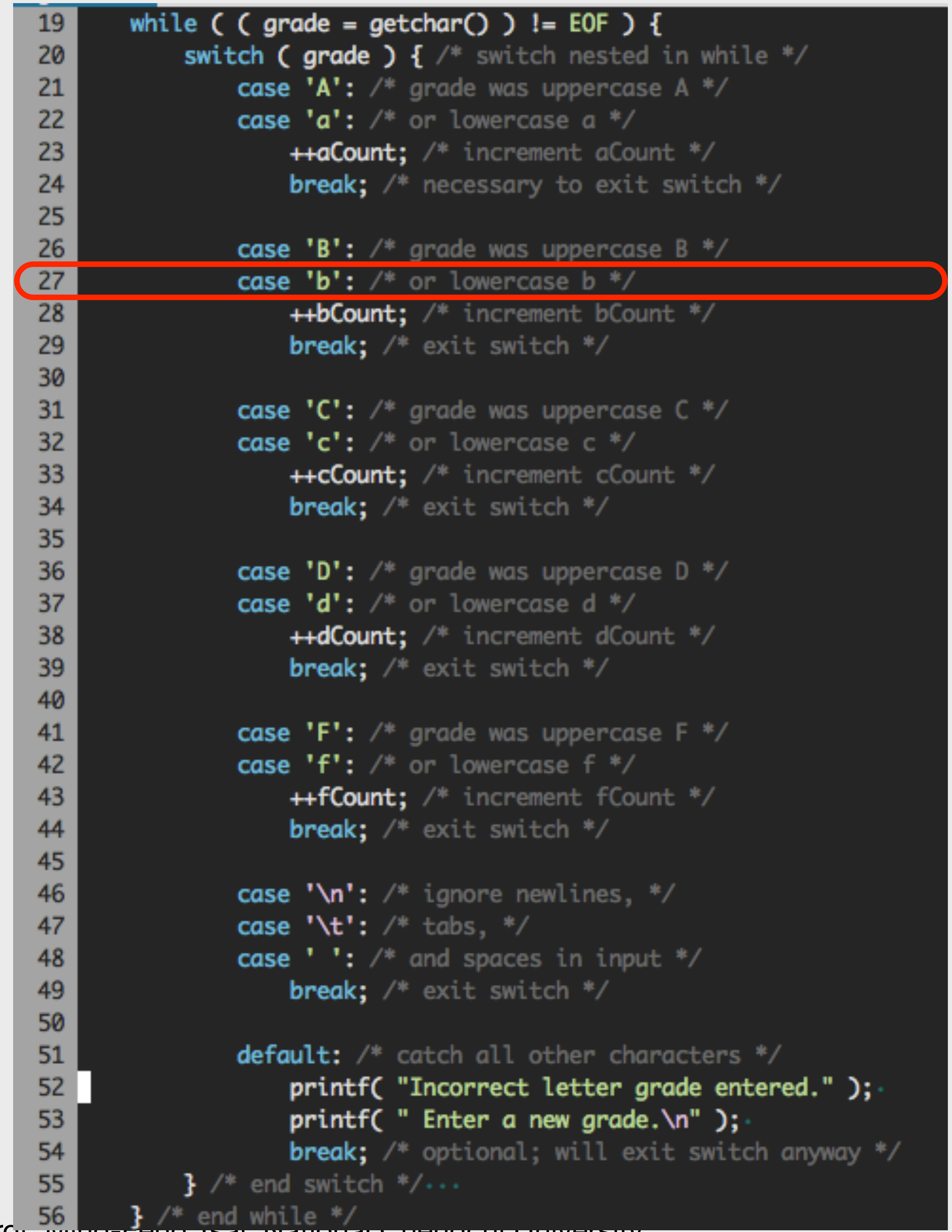


```c
    while ( ( grade = getchar() ) != EOF ) {
        switch ( grade ) { /* switch nested in while */
            case 'A': /* grade was uppercase A */
            case 'a': /* or lowercase a */
                ++aCount; /* increment aCount */
                break; /* necessary to exit switch */

            case 'B': /* grade was uppercase B */
            case 'b': /* or lowercase b */
                ++bCount; /* increment bCount */
                break; /* exit switch */

            case 'C': /* grade was uppercase C */
            case 'c': /* or lowercase c */
                ++cCount; /* increment cCount */
                break; /* exit switch */

            case 'D': /* grade was uppercase D */
            case 'd': /* or lowercase d */
                ++dCount; /* increment dCount */
                break; /* exit switch */

            case 'F': /* grade was uppercase F */
            case 'f': /* or lowercase f */
                ++fCount; /* increment fCount */
                break; /* exit switch */

            case '\n': /* ignore newlines, */
            case '\t': /* tabs, */
            case ' ': /* and spaces in input */
                break; /* exit switch */

            default: /* catch all other characters */
                printf( "Incorrect letter grade entered." );
                printf( " Enter a new grade.\n" );
                break; /* optional; will exit switch anyway */
        } /* end switch */
    } /* end while */
```

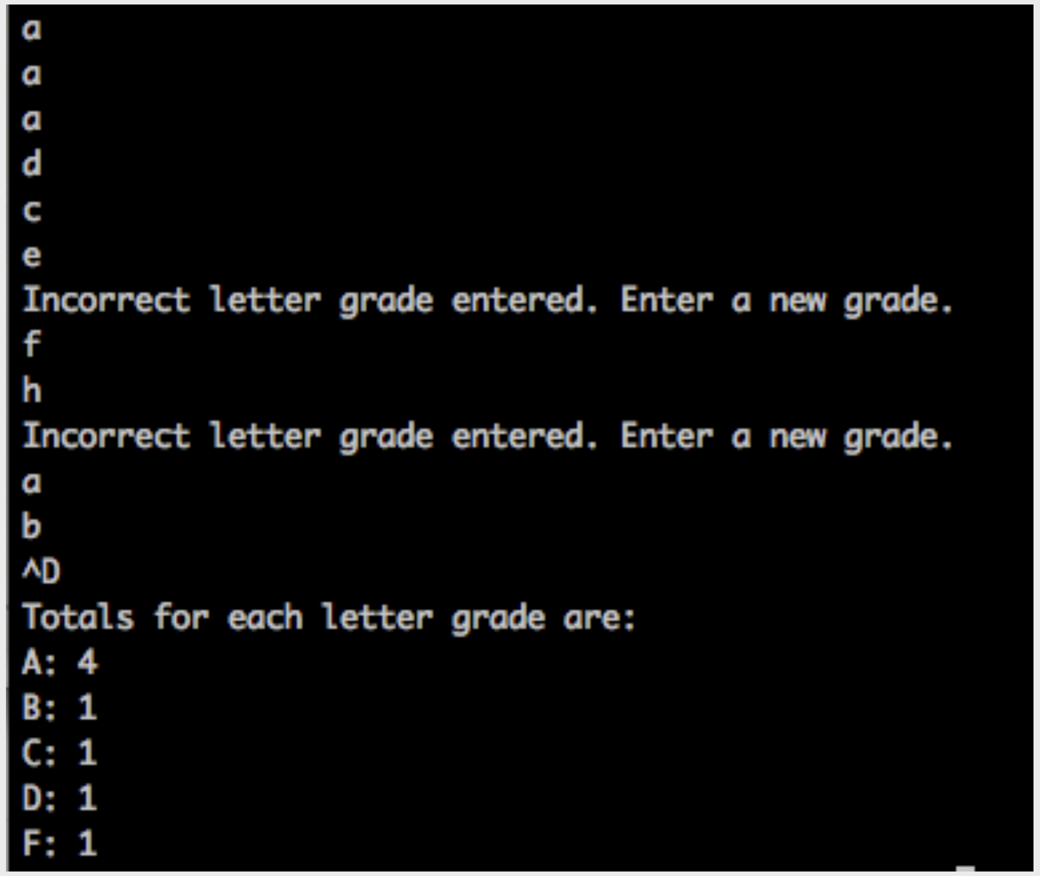


```
a
a
a
d
c
e
Incorrect letter grade entered. Enter a new grade.
f
h
Incorrect letter grade entered. Enter a new grade.
a
b
^D
Totals for each letter grade are:
A: 4
B: 1
C: 1
D: 1
F: 1
```

Example: fig04_07.c

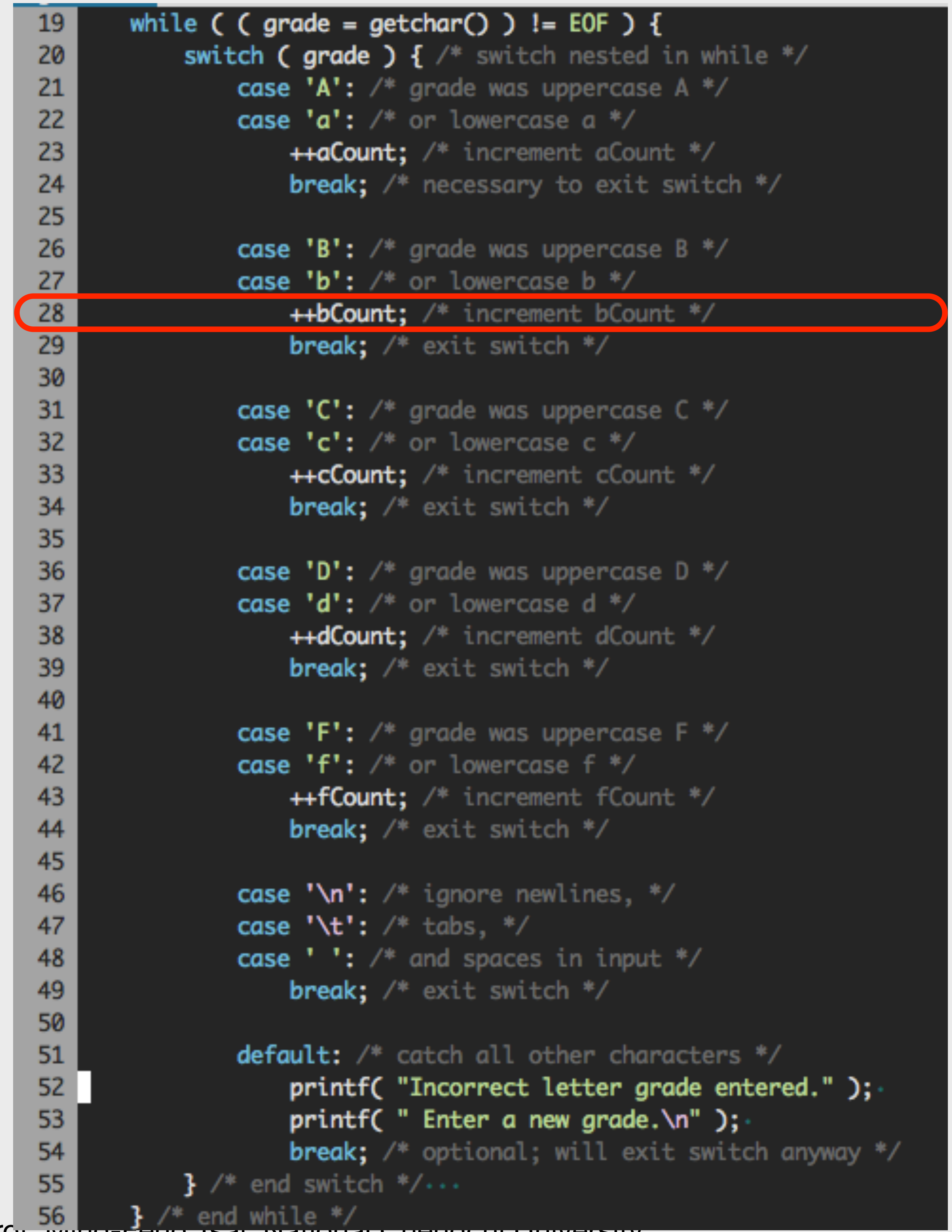


```
    while ( ( grade = getchar() ) != EOF ) {
        switch ( grade ) { /* switch nested in while */
            case 'A': /* grade was uppercase A */
            case 'a': /* or lowercase a */
                ++aCount; /* increment aCount */
                break; /* necessary to exit switch */

            case 'B': /* grade was uppercase B */
            case 'b': /* or lowercase b */
                ++bCount; /* increment bCount */
                break; /* exit switch */

            case 'C': /* grade was uppercase C */
            case 'c': /* or lowercase c */
                ++cCount; /* increment cCount */
                break; /* exit switch */

            case 'D': /* grade was uppercase D */
            case 'd': /* or lowercase d */
                ++dCount; /* increment dCount */
                break; /* exit switch */

            case 'F': /* grade was uppercase F */
            case 'f': /* or lowercase f */
                ++fCount; /* increment fCount */
                break; /* exit switch */

            case '\n': /* ignore newlines, */
            case '\t': /* tabs, */
            case ' ': /* and spaces in input */
                break; /* exit switch */

            default: /* catch all other characters */
                printf( "Incorrect letter grade entered." );
                printf( " Enter a new grade.\n" );
                break; /* optional; will exit switch anyway */
        } /* end switch */
    } /* end while */
```

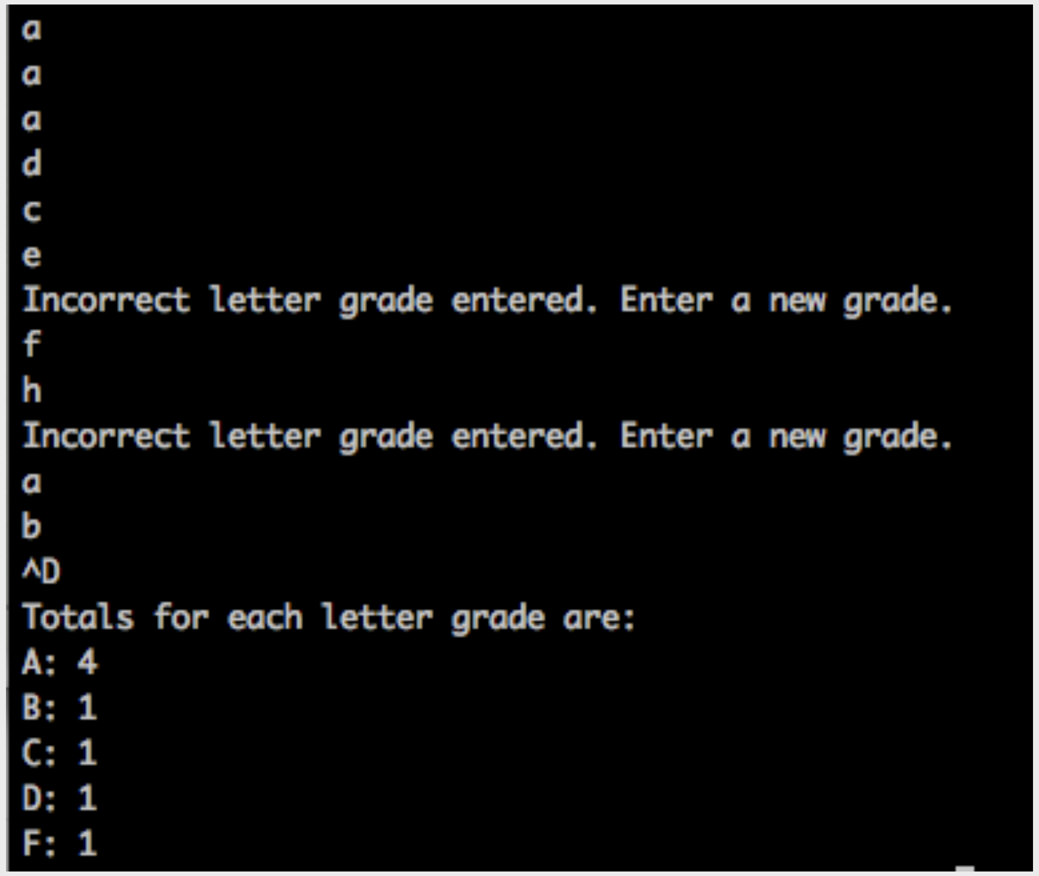


```
a
a
a
d
c
e
Incorrect letter grade entered. Enter a new grade.
f
h
Incorrect letter grade entered. Enter a new grade.
a
b
^D
Totals for each letter grade are:
A: 4
B: 1
C: 1
D: 1
F: 1
```

Example: fig04_07.c

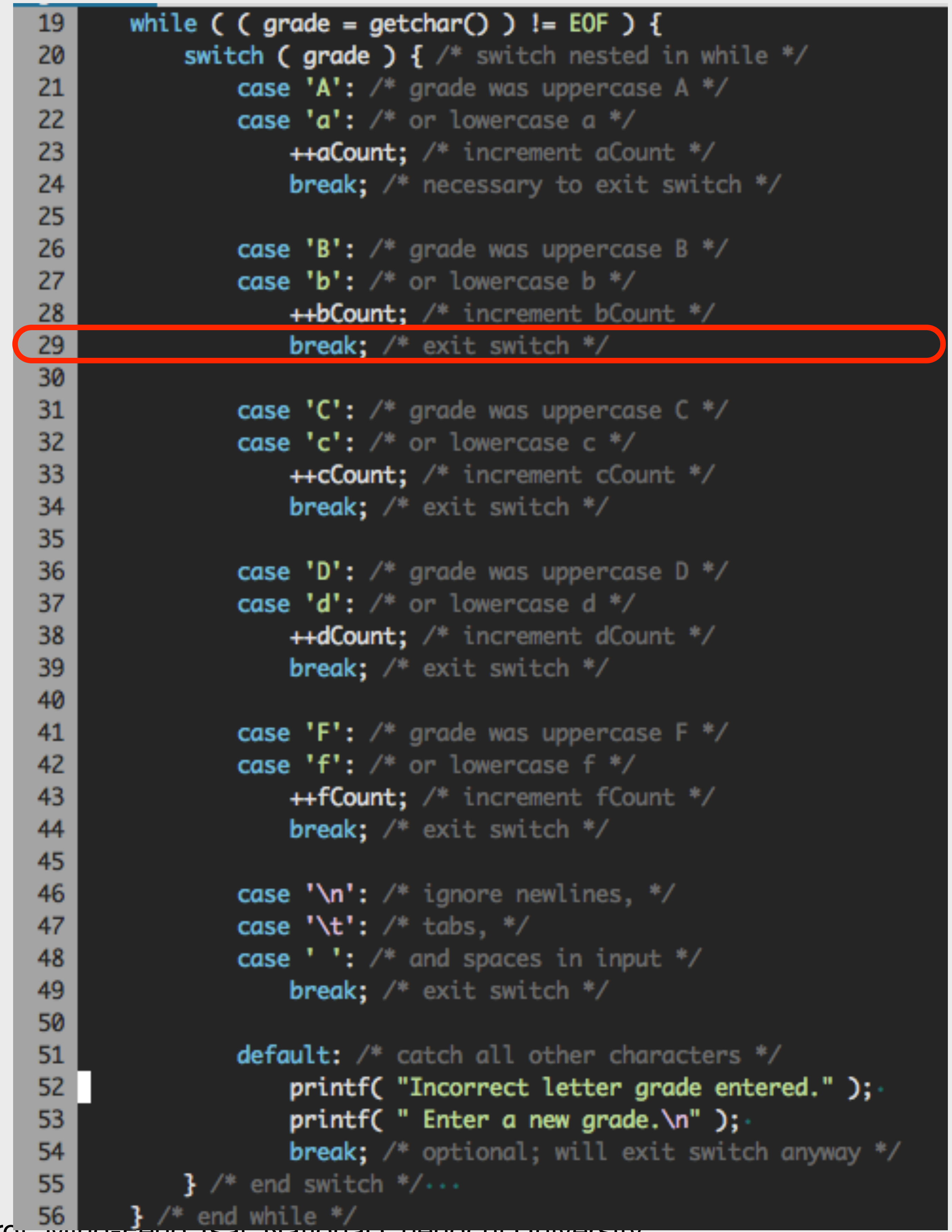


```c
    while ( ( grade = getchar() ) != EOF ) {
        switch ( grade ) { /* switch nested in while */
            case 'A': /* grade was uppercase A */
            case 'a': /* or lowercase a */
                ++aCount; /* increment aCount */
                break; /* necessary to exit switch */

            case 'B': /* grade was uppercase B */
            case 'b': /* or lowercase b */
                ++bCount; /* increment bCount */
                break; /* exit switch */

            case 'C': /* grade was uppercase C */
            case 'c': /* or lowercase c */
                ++cCount; /* increment cCount */
                break; /* exit switch */

            case 'D': /* grade was uppercase D */
            case 'd': /* or lowercase d */
                ++dCount; /* increment dCount */
                break; /* exit switch */

            case 'F': /* grade was uppercase F */
            case 'f': /* or lowercase f */
                ++fCount; /* increment fCount */
                break; /* exit switch */

            case '\n': /* ignore newlines, */
            case '\t': /* tabs, */
            case ' ': /* and spaces in input */
                break; /* exit switch */

            default: /* catch all other characters */
                printf( "Incorrect letter grade entered." );
                printf( " Enter a new grade.\n" );
                break; /* optional; will exit switch anyway */
        } /* end switch */
    } /* end while */
```

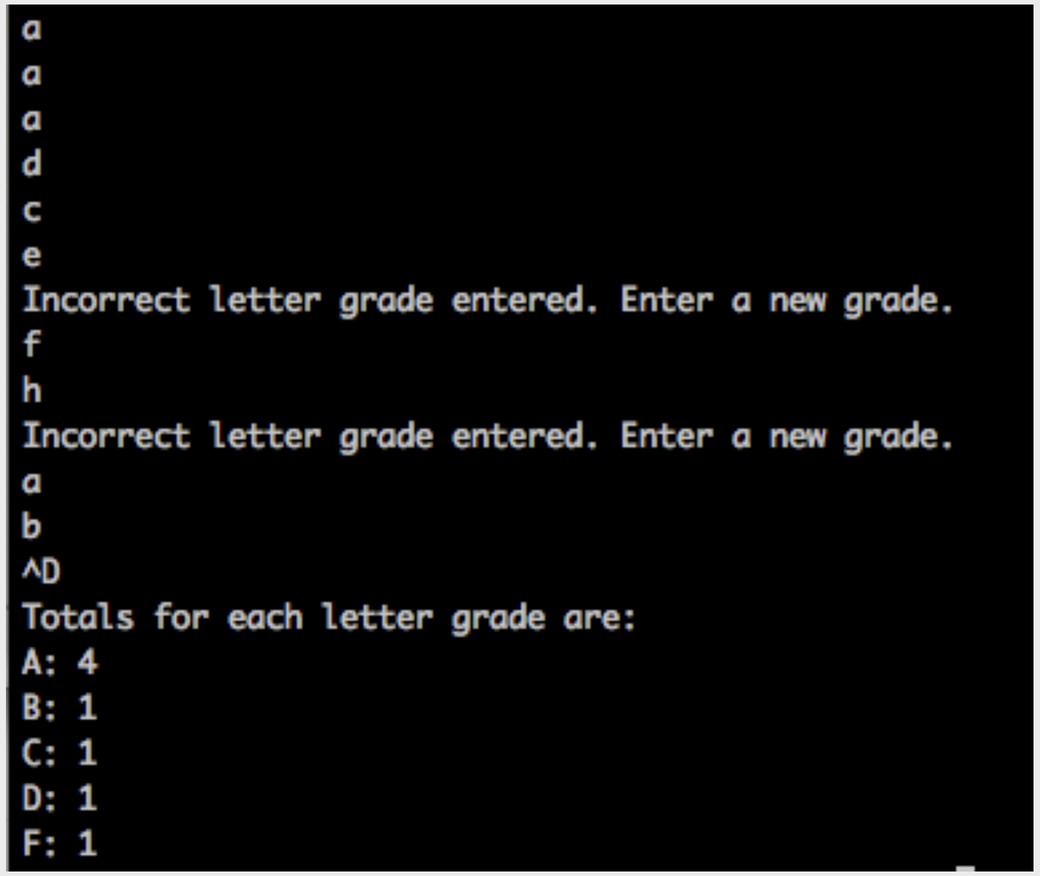


```
a
a
a
d
c
e
Incorrect letter grade entered. Enter a new grade.
f
h
Incorrect letter grade entered. Enter a new grade.
a
b
^D
Totals for each letter grade are:
A: 4
B: 1
C: 1
D: 1
F: 1
```

Example: fig04_07.c

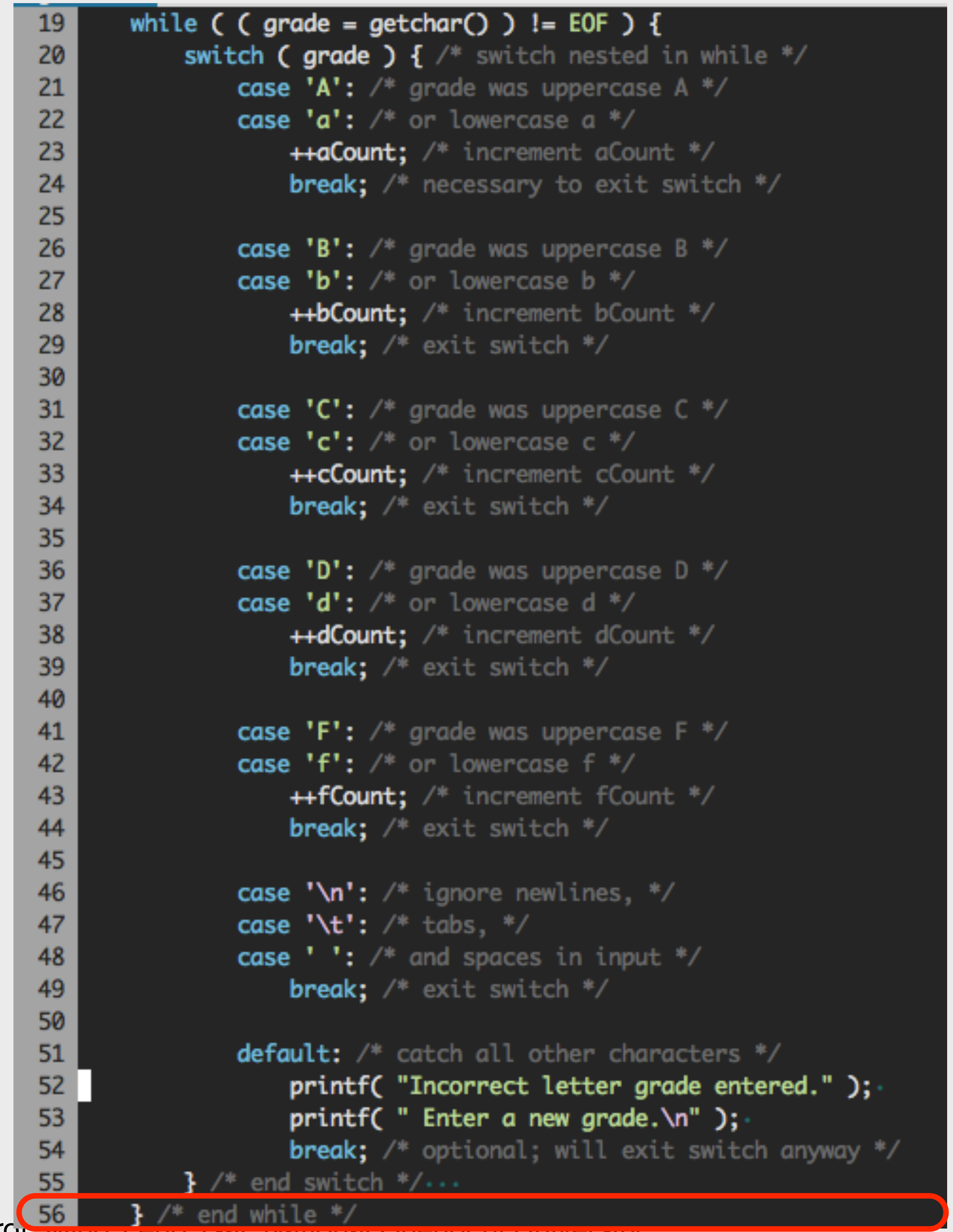


```c
    while ( ( grade = getchar() ) != EOF ) {
        switch ( grade ) { /* switch nested in while */
            case 'A': /* grade was uppercase A */
            case 'a': /* or lowercase a */
                ++aCount; /* increment aCount */
                break; /* necessary to exit switch */

            case 'B': /* grade was uppercase B */
            case 'b': /* or lowercase b */
                ++bCount; /* increment bCount */
                break; /* exit switch */

            case 'C': /* grade was uppercase C */
            case 'c': /* or lowercase c */
                ++cCount; /* increment cCount */
                break; /* exit switch */

            case 'D': /* grade was uppercase D */
            case 'd': /* or lowercase d */
                ++dCount; /* increment dCount */
                break; /* exit switch */

            case 'F': /* grade was uppercase F */
            case 'f': /* or lowercase f */
                ++fCount; /* increment fCount */
                break; /* exit switch */

            case '\n': /* ignore newlines, */
            case '\t': /* tabs, */
            case ' ': /* and spaces in input */
                break; /* exit switch */

            default: /* catch all other characters */
                printf( "Incorrect letter grade entered." );
                printf( " Enter a new grade.\n" );
                break; /* optional; will exit switch anyway */
        } /* end switch */
    } /* end while */
```

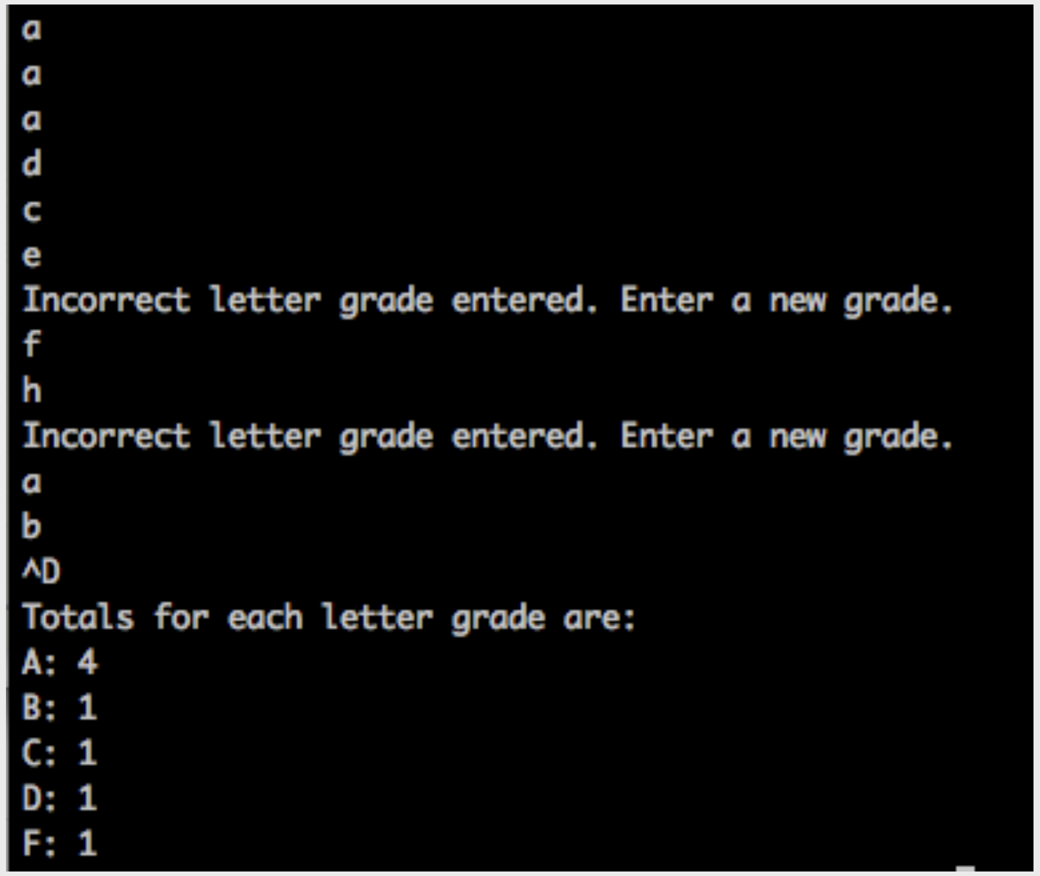


```
a
a
a
d
c
e
Incorrect letter grade entered. Enter a new grade.
f
h
Incorrect letter grade entered. Enter a new grade.
a
b
^D
Totals for each letter grade are:
A: 4
B: 1
C: 1
D: 1
F: 1
```

Example: fig04_07.c

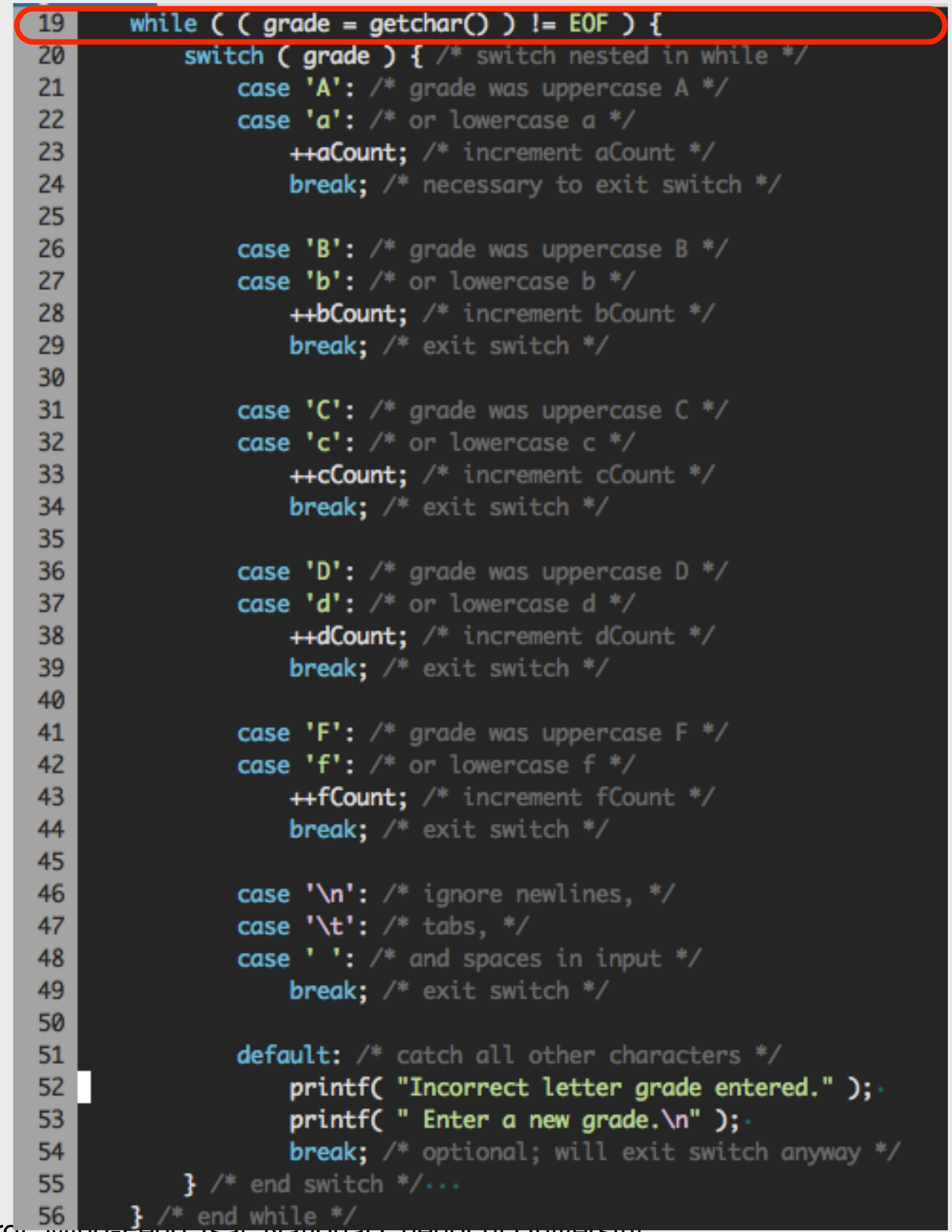


```c
    while ( ( grade = getchar() ) != EOF ) {
        switch ( grade ) { /* switch nested in while */
            case 'A': /* grade was uppercase A */
            case 'a': /* or lowercase a */
                ++aCount; /* increment aCount */
                break; /* necessary to exit switch */

            case 'B': /* grade was uppercase B */
            case 'b': /* or lowercase b */
                ++bCount; /* increment bCount */
                break; /* exit switch */

            case 'C': /* grade was uppercase C */
            case 'c': /* or lowercase c */
                ++cCount; /* increment cCount */
                break; /* exit switch */

            case 'D': /* grade was uppercase D */
            case 'd': /* or lowercase d */
                ++dCount; /* increment dCount */
                break; /* exit switch */

            case 'F': /* grade was uppercase F */
            case 'f': /* or lowercase f */
                ++fCount; /* increment fCount */
                break; /* exit switch */

            case '\n': /* ignore newlines, */
            case '\t': /* tabs, */
            case ' ': /* and spaces in input */
                break; /* exit switch */

            default: /* catch all other characters */
                printf( "Incorrect letter grade entered." );
                printf( " Enter a new grade.\n" );
                break; /* optional; will exit switch anyway */
        } /* end switch */
    } /* end while */
```

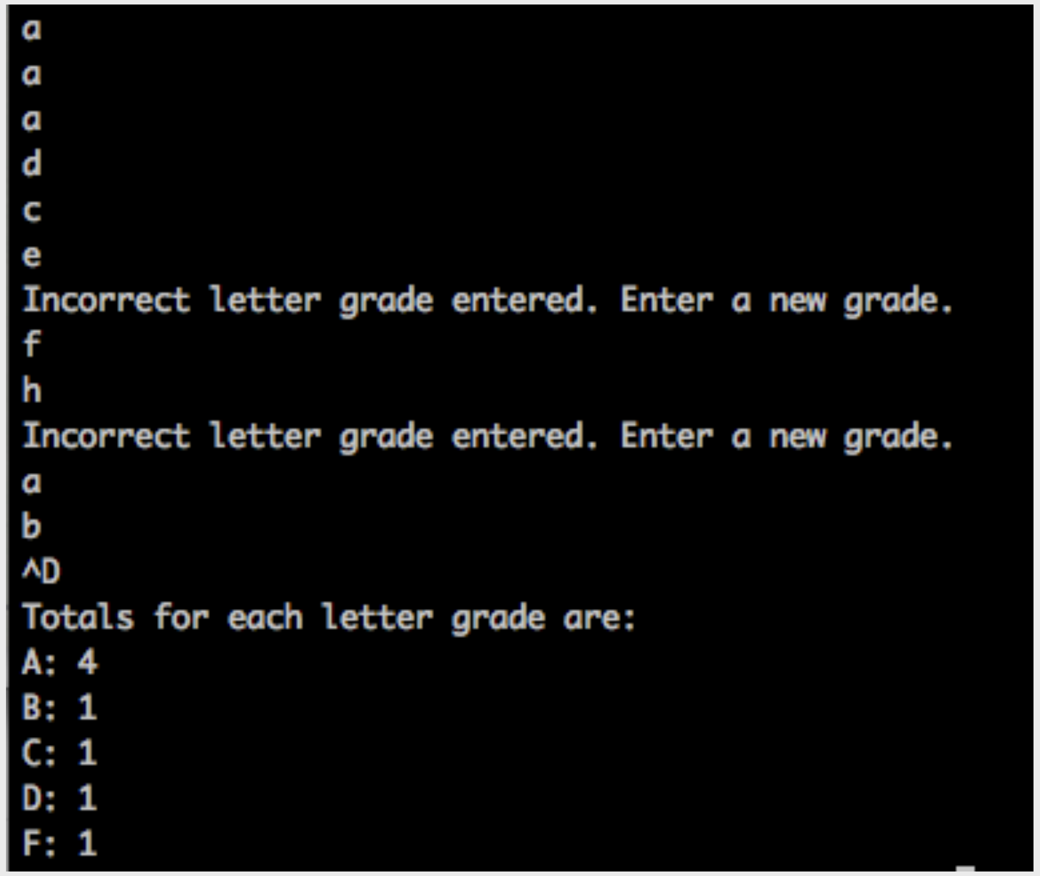


```
a
a
a
d
c
e
Incorrect letter grade entered. Enter a new grade.
f
h
Incorrect letter grade entered. Enter a new grade.
a
b
^D
Totals for each letter grade are:
A: 4
B: 1
C: 1
D: 1
F: 1
```

Example: fig04_07.c

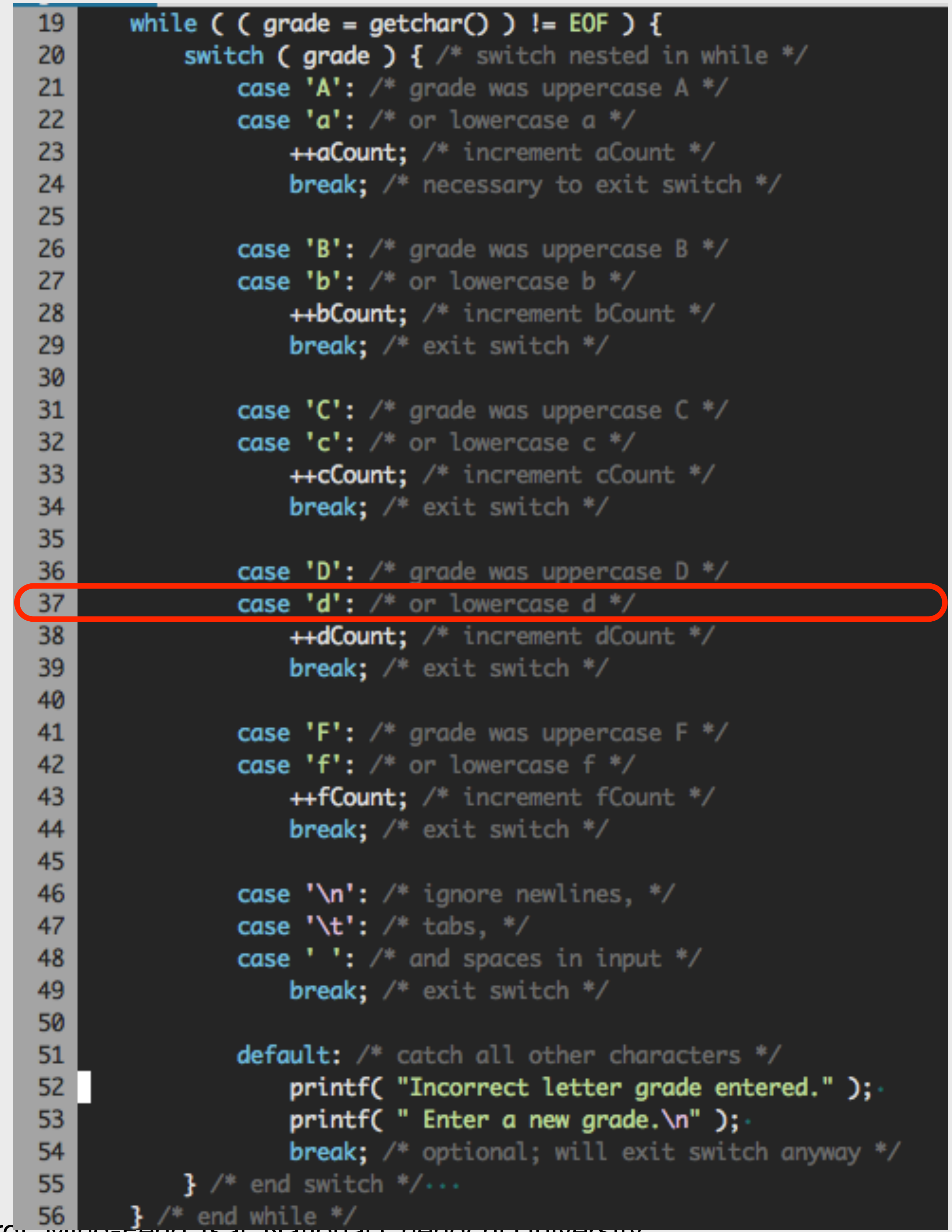


```c
    while ( ( grade = getchar() ) != EOF ) {
        switch ( grade ) { /* switch nested in while */
            case 'A': /* grade was uppercase A */
            case 'a': /* or lowercase a */
                ++aCount; /* increment aCount */
                break; /* necessary to exit switch */

            case 'B': /* grade was uppercase B */
            case 'b': /* or lowercase b */
                ++bCount; /* increment bCount */
                break; /* exit switch */

            case 'C': /* grade was uppercase C */
            case 'c': /* or lowercase c */
                ++cCount; /* increment cCount */
                break; /* exit switch */

            case 'D': /* grade was uppercase D */
            case 'd': /* or lowercase d */
                ++dCount; /* increment dCount */
                break; /* exit switch */

            case 'F': /* grade was uppercase F */
            case 'f': /* or lowercase f */
                ++fCount; /* increment fCount */
                break; /* exit switch */

            case '\n': /* ignore newlines, */
            case '\t': /* tabs, */
            case ' ': /* and spaces in input */
                break; /* exit switch */

            default: /* catch all other characters */
                printf( "Incorrect letter grade entered." );
                printf( " Enter a new grade.\n" );
                break; /* optional; will exit switch anyway */
        } /* end switch */
    } /* end while */
```

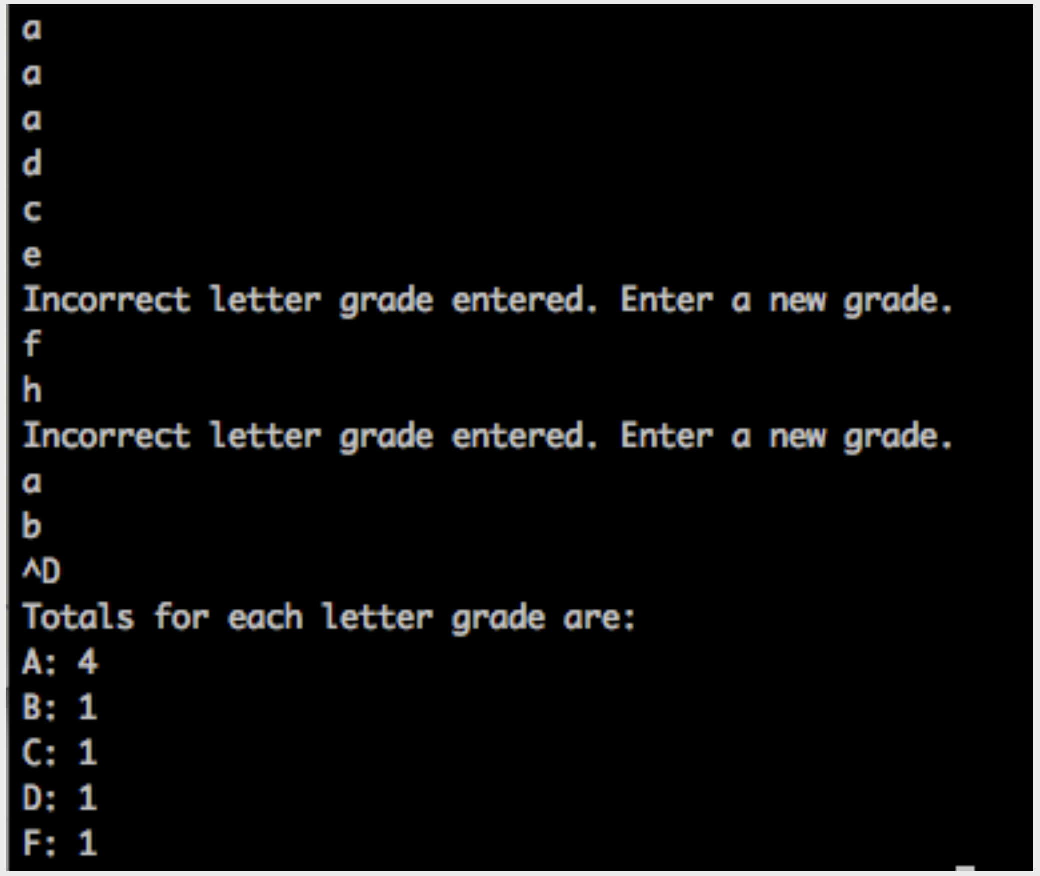


```
a
a
a
d
c
e
Incorrect letter grade entered. Enter a new grade.
f
h
Incorrect letter grade entered. Enter a new grade.
a
b
^D
Totals for each letter grade are:
A: 4
B: 1
C: 1
D: 1
F: 1
```

Example: fig04_07.c

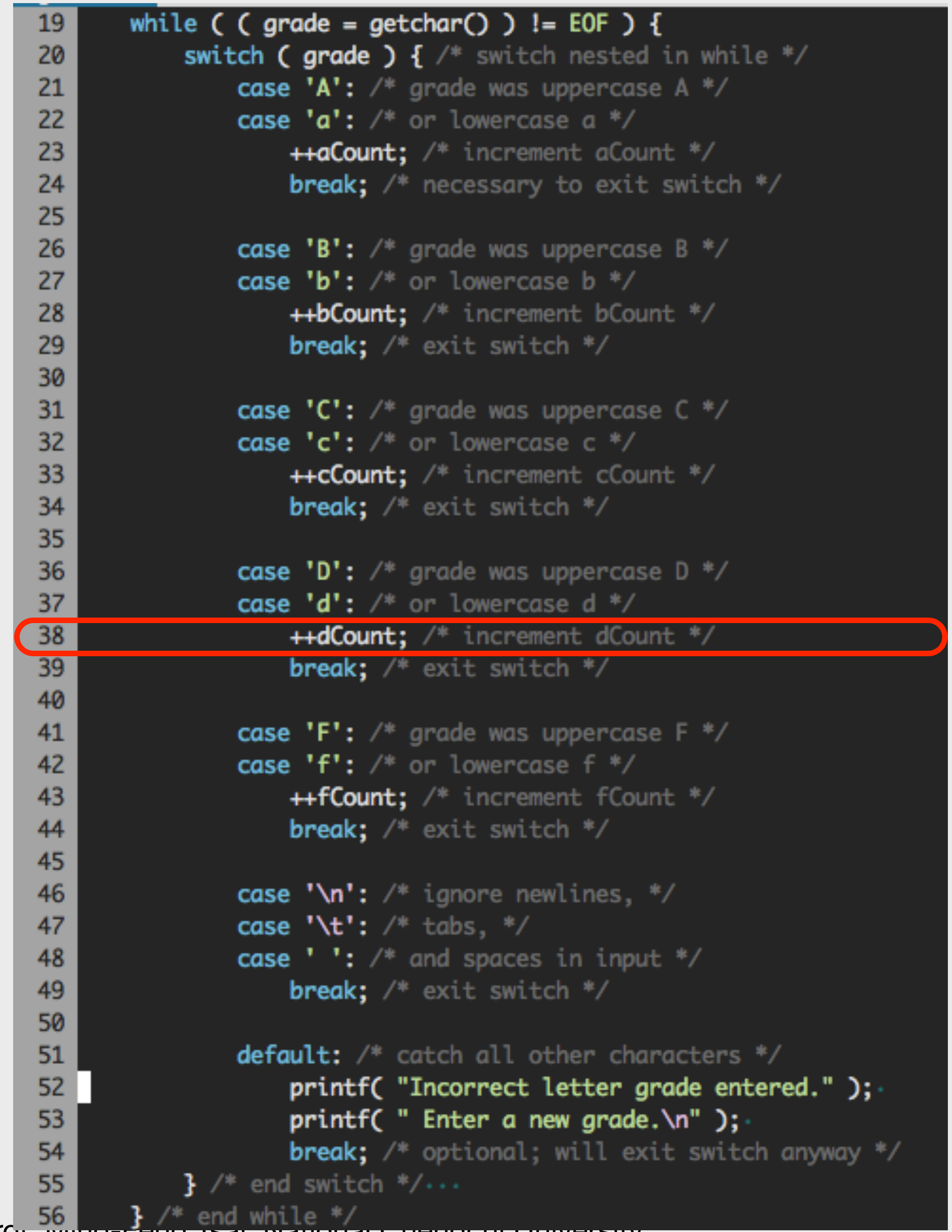


```c
    while ( ( grade = getchar() ) != EOF ) {
        switch ( grade ) { /* switch nested in while */
            case 'A': /* grade was uppercase A */
            case 'a': /* or lowercase a */
                ++aCount; /* increment aCount */
                break; /* necessary to exit switch */

            case 'B': /* grade was uppercase B */
            case 'b': /* or lowercase b */
                ++bCount; /* increment bCount */
                break; /* exit switch */

            case 'C': /* grade was uppercase C */
            case 'c': /* or lowercase c */
                ++cCount; /* increment cCount */
                break; /* exit switch */

            case 'D': /* grade was uppercase D */
            case 'd': /* or lowercase d */
                ++dCount; /* increment dCount */
                break; /* exit switch */

            case 'F': /* grade was uppercase F */
            case 'f': /* or lowercase f */
                ++fCount; /* increment fCount */
                break; /* exit switch */

            case '\n': /* ignore newlines, */
            case '\t': /* tabs, */
            case ' ': /* and spaces in input */
                break; /* exit switch */

            default: /* catch all other characters */
                printf( "Incorrect letter grade entered." );
                printf( " Enter a new grade.\n" );
                break; /* optional; will exit switch anyway */
        } /* end switch */
    } /* end while */
```

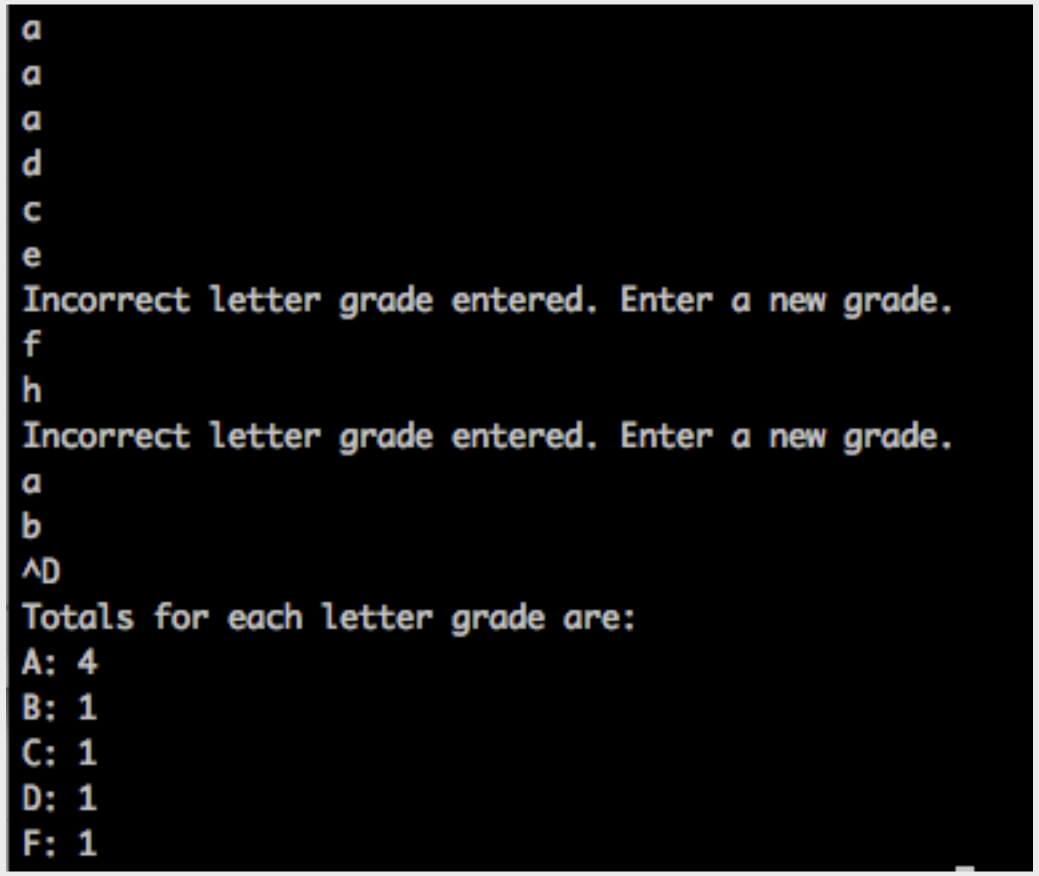


```
a
a
a
d
c
e
Incorrect letter grade entered. Enter a new grade.
f
h
Incorrect letter grade entered. Enter a new grade.
a
b
^D
Totals for each letter grade are:
A: 4
B: 1
C: 1
D: 1
F: 1
```

Example: fig04_07.c

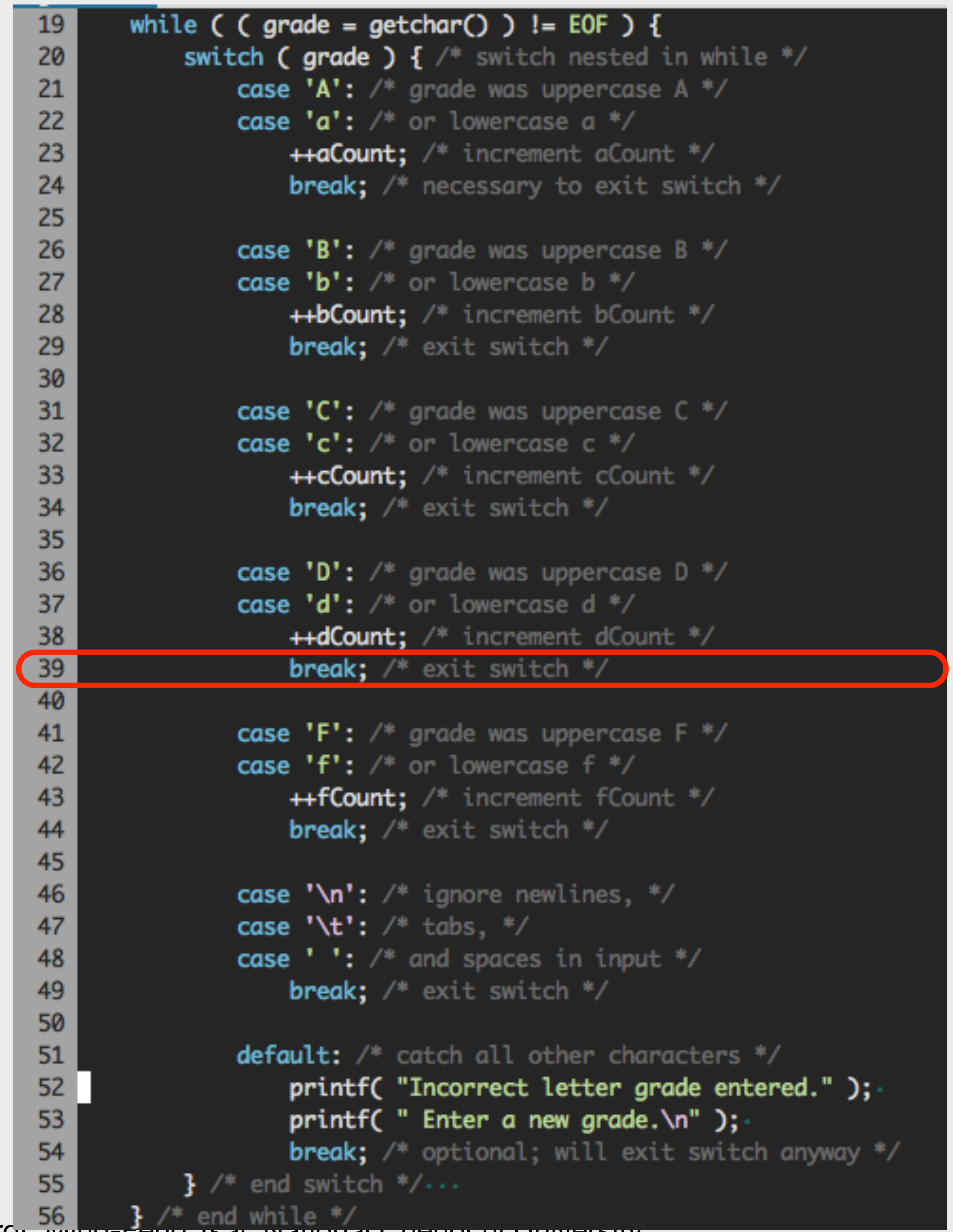


```c
    while ( ( grade = getchar() ) != EOF ) {
        switch ( grade ) { /* switch nested in while */
            case 'A': /* grade was uppercase A */
            case 'a': /* or lowercase a */
                ++aCount; /* increment aCount */
                break; /* necessary to exit switch */

            case 'B': /* grade was uppercase B */
            case 'b': /* or lowercase b */
                ++bCount; /* increment bCount */
                break; /* exit switch */

            case 'C': /* grade was uppercase C */
            case 'c': /* or lowercase c */
                ++cCount; /* increment cCount */
                break; /* exit switch */

            case 'D': /* grade was uppercase D */
            case 'd': /* or lowercase d */
                ++dCount; /* increment dCount */
                break; /* exit switch */

            case 'F': /* grade was uppercase F */
            case 'f': /* or lowercase f */
                ++fCount; /* increment fCount */
                break; /* exit switch */

            case '\n': /* ignore newlines, */
            case '\t': /* tabs, */
            case ' ': /* and spaces in input */
                break; /* exit switch */

            default: /* catch all other characters */
                printf( "Incorrect letter grade entered." );
                printf( " Enter a new grade.\n" );
                break; /* optional; will exit switch anyway */
        } /* end switch */
    } /* end while */
```

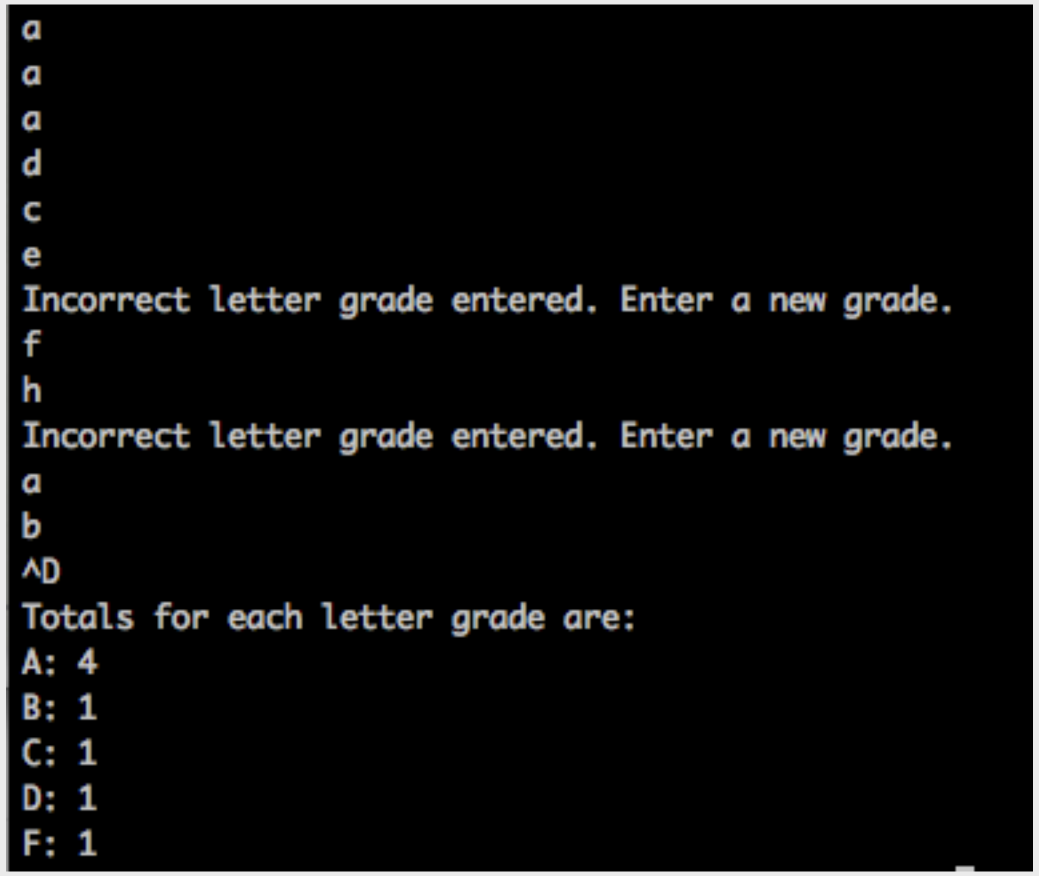


```
a
a
a
d
c
e
Incorrect letter grade entered. Enter a new grade.
f
h
Incorrect letter grade entered. Enter a new grade.
a
b
^D
Totals for each letter grade are:
A: 4
B: 1
C: 1
D: 1
F: 1
```

Example: fig04_07.c

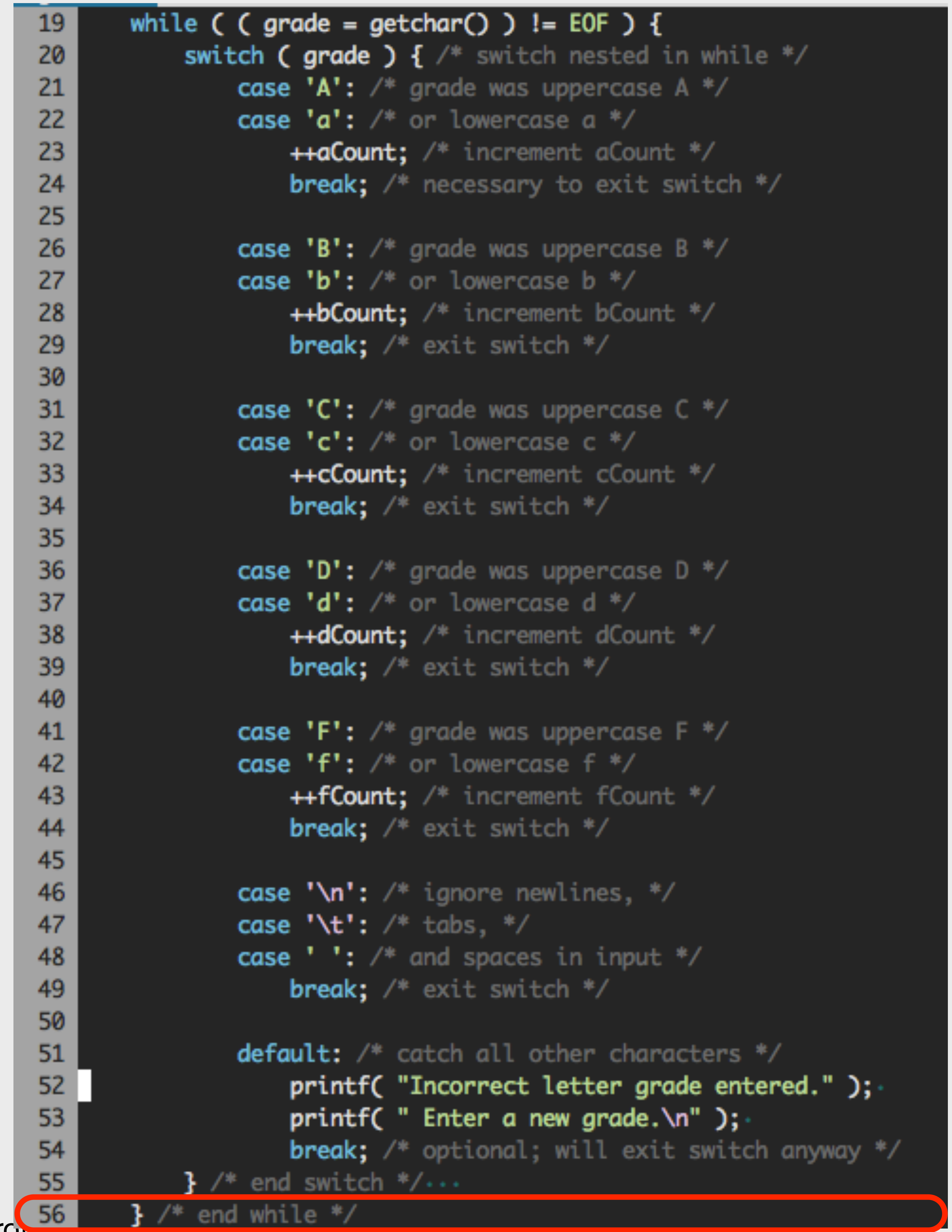


```
    while ( ( grade = getchar() ) != EOF ) {
        switch ( grade ) { /* switch nested in while */
            case 'A': /* grade was uppercase A */
            case 'a': /* or lowercase a */
                ++aCount; /* increment aCount */
                break; /* necessary to exit switch */

            case 'B': /* grade was uppercase B */
            case 'b': /* or lowercase b */
                ++bCount; /* increment bCount */
                break; /* exit switch */

            case 'C': /* grade was uppercase C */
            case 'c': /* or lowercase c */
                ++cCount; /* increment cCount */
                break; /* exit switch */

            case 'D': /* grade was uppercase D */
            case 'd': /* or lowercase d */
                ++dCount; /* increment dCount */
                break; /* exit switch */

            case 'F': /* grade was uppercase F */
            case 'f': /* or lowercase f */
                ++fCount; /* increment fCount */
                break; /* exit switch */

            case '\n': /* ignore newlines, */
            case '\t': /* tabs, */
            case ' ': /* and spaces in input */
                break; /* exit switch */

            default: /* catch all other characters */
                printf( "Incorrect letter grade entered." );
                printf( " Enter a new grade.\n" );
                break; /* optional; will exit switch anyway */
        } /* end switch */
    } /* end while */
```

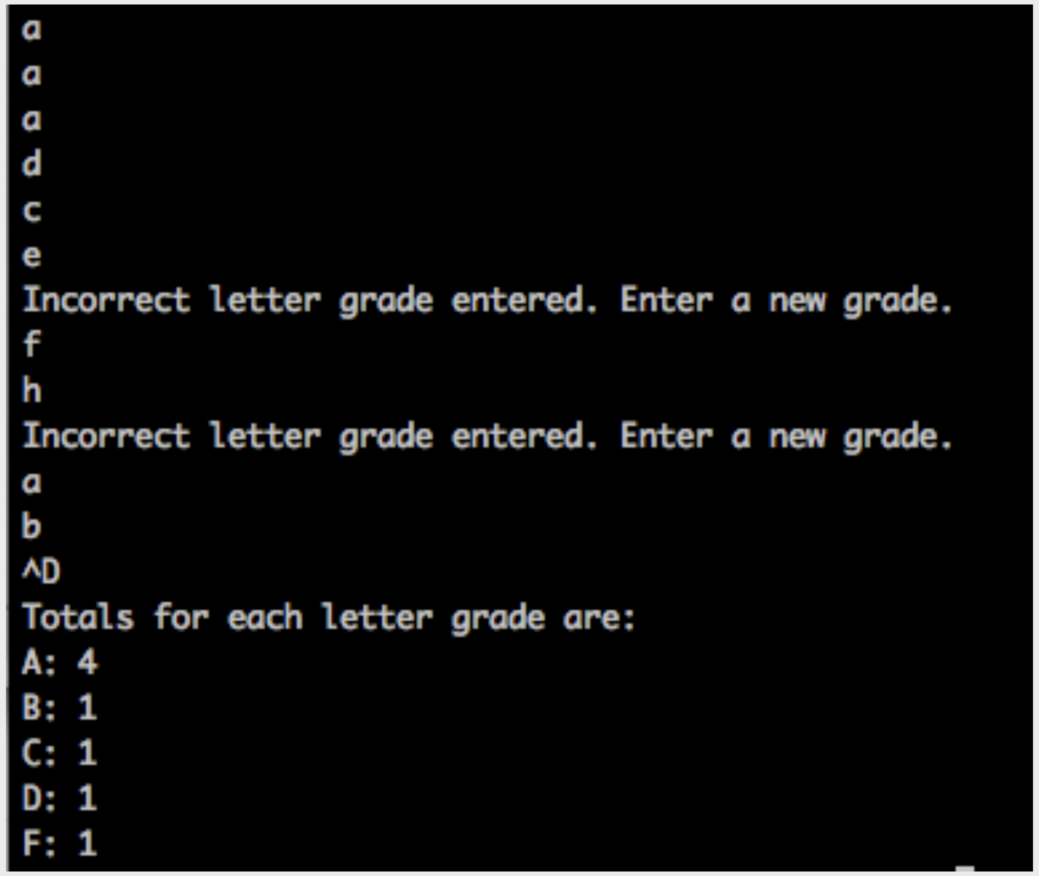


```
a
a
a
d
c
e
Incorrect letter grade entered. Enter a new grade.
f
h
Incorrect letter grade entered. Enter a new grade.
a
b
^D
Totals for each letter grade are:
A: 4
B: 1
C: 1
D: 1
F: 1
```

Example: fig04_07.c

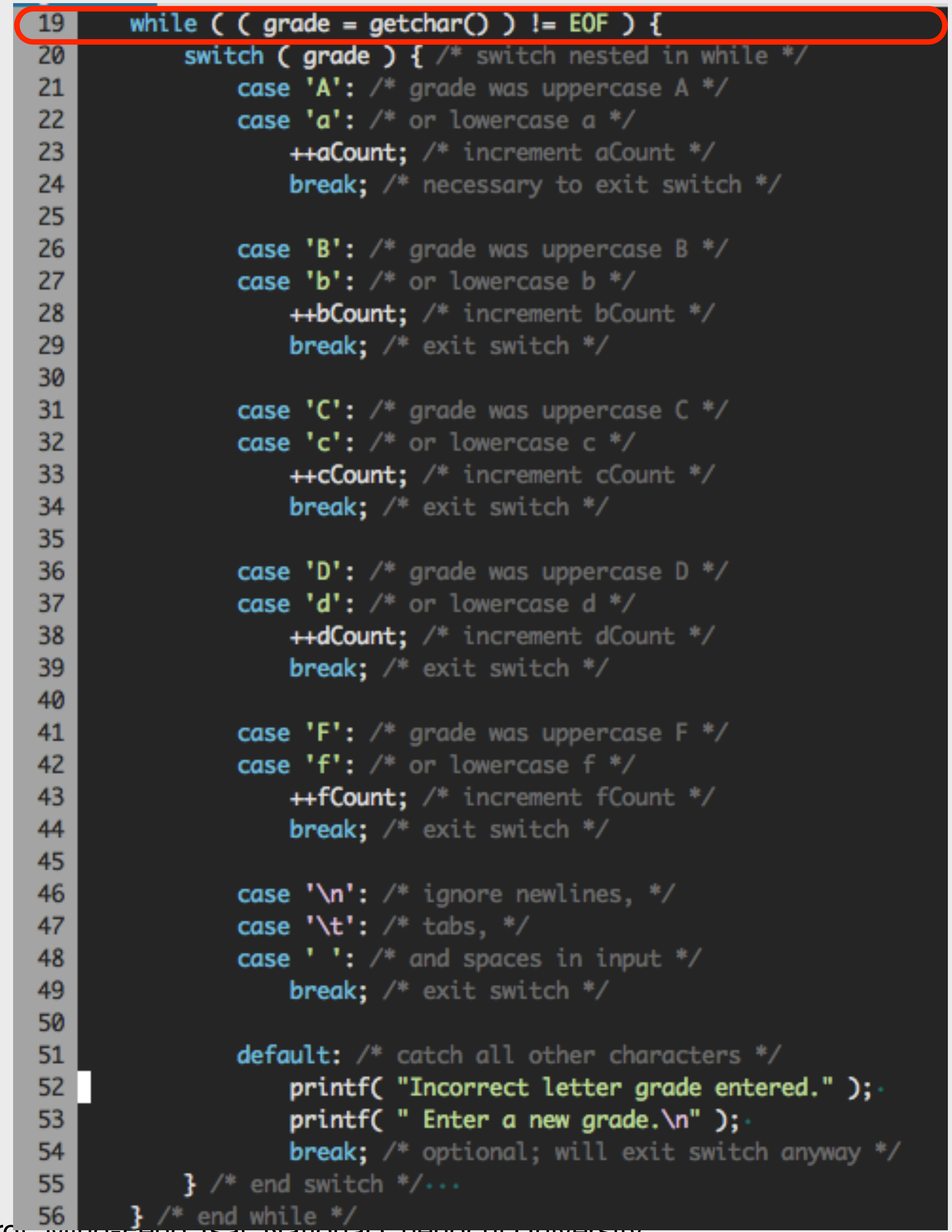


```c
    while ( ( grade = getchar() ) != EOF ) {
        switch ( grade ) { /* switch nested in while */
            case 'A': /* grade was uppercase A */
            case 'a': /* or lowercase a */
                ++aCount; /* increment aCount */
                break; /* necessary to exit switch */

            case 'B': /* grade was uppercase B */
            case 'b': /* or lowercase b */
                ++bCount; /* increment bCount */
                break; /* exit switch */

            case 'C': /* grade was uppercase C */
            case 'c': /* or lowercase c */
                ++cCount; /* increment cCount */
                break; /* exit switch */

            case 'D': /* grade was uppercase D */
            case 'd': /* or lowercase d */
                ++dCount; /* increment dCount */
                break; /* exit switch */

            case 'F': /* grade was uppercase F */
            case 'f': /* or lowercase f */
                ++fCount; /* increment fCount */
                break; /* exit switch */

            case '\n': /* ignore newlines, */
            case '\t': /* tabs, */
            case ' ': /* and spaces in input */
                break; /* exit switch */

            default: /* catch all other characters */
                printf( "Incorrect letter grade entered." );
                printf( " Enter a new grade.\n" );
                break; /* optional; will exit switch anyway */
        } /* end switch */
    } /* end while */
```

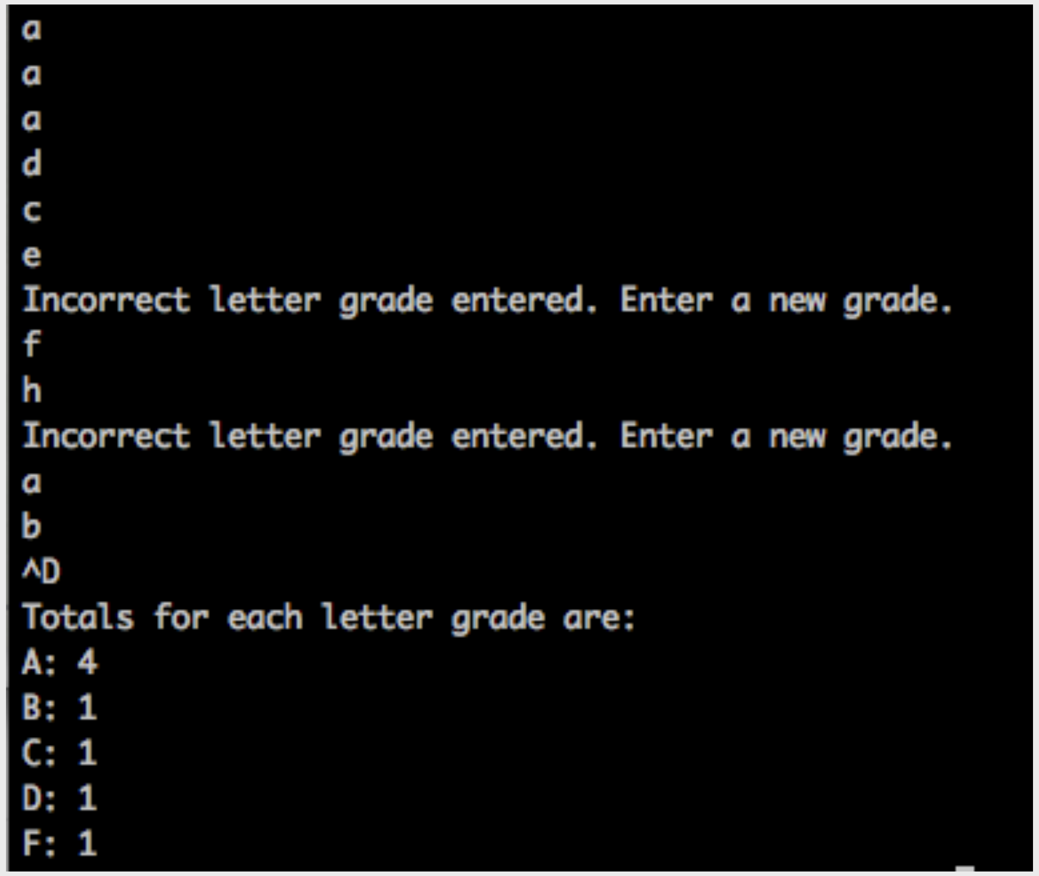


```
a
a
a
d
c
e
Incorrect letter grade entered. Enter a new grade.
f
h
Incorrect letter grade entered. Enter a new grade.
a
b
^D
Totals for each letter grade are:
A: 4
B: 1
C: 1
D: 1
F: 1
```

Example: fig04_07.c



```
    while ( ( grade = getchar() ) != EOF ) {
        switch ( grade ) { /* switch nested in while */
            case 'A': /* grade was uppercase A */
            case 'a': /* or lowercase a */
                ++aCount; /* increment aCount */
                break; /* necessary to exit switch */

            case 'B': /* grade was uppercase B */
            case 'b': /* or lowercase b */
                ++bCount; /* increment bCount */
                break; /* exit switch */

            case 'C': /* grade was uppercase C */
            case 'c': /* or lowercase c */
                ++cCount; /* increment cCount */
                break; /* exit switch */

            case 'D': /* grade was uppercase D */
            case 'd': /* or lowercase d */
                ++dCount; /* increment dCount */
                break; /* exit switch */

            case 'F': /* grade was uppercase F */
            case 'f': /* or lowercase f */
                ++fCount; /* increment fCount */
                break; /* exit switch */

            case '\n': /* ignore newlines, */
            case '\t': /* tabs, */
            case ' ': /* and spaces in input */
                break; /* exit switch */

            default: /* catch all other characters */
                printf( "Incorrect letter grade entered." );
                printf( " Enter a new grade.\n" );
                break; /* optional; will exit switch anyway */
        } /* end switch */
    } /* end while */
```



```
a
a
a
d
c
e
Incorrect letter grade entered. Enter a new grade.
f
h
Incorrect letter grade entered. Enter a new grade.
a
b
^D
Totals for each letter grade are:
A: 4
B: 1
C: 1
D: 1
F: 1
```

`switch` Multiple-Selection Statement (Cont.)

- The **`getchar`** function (from **`<stdio.h>`**) reads one character from the keyboard and stores that character in the integer variable grade.
    - Many computers today use the ASCII (American Standard Code for Information Interchange) character set in which 97 represents the lowercase letter 'a'.
- In the program, the value of the assignment **`grade = getchar()`** is compared with the value of **`EOF`** (a symbol whose acronym stands for “end of file”).

`switch` Multiple-Selection Statement (Cont.)

- On Linux/UNIX/Mac OS X systems, the **`EOF`** indicator is entered by typing
  - **`<Ctrl>  d`**
- On other systems, such as Microsoft Windows, the **`EOF`** indicator can be entered by typing
  - **`<Ctrl>  z`**

`switch` Multiple-Selection Statement (Cont.)

- Keyword switch is followed by the variable name grade in parentheses.
  - This is called the controlling expression.
  - The value of this expression is compared with each of the case labels.
  - The **`break`** statement causes program control to continue with the first statement after the switch statement.
  - If no match occurs, the **`default`** case is executed, and an error message is printed.

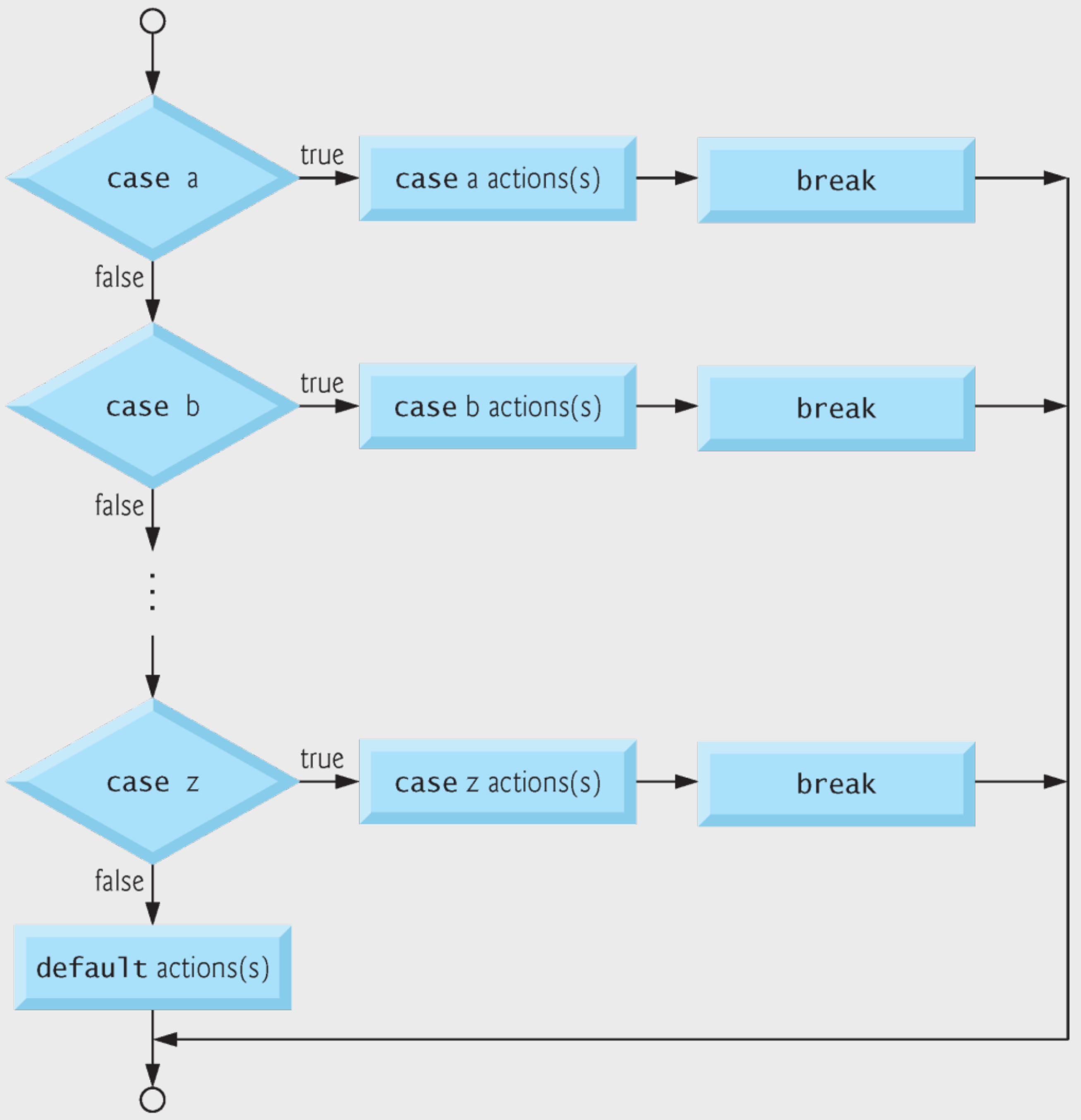
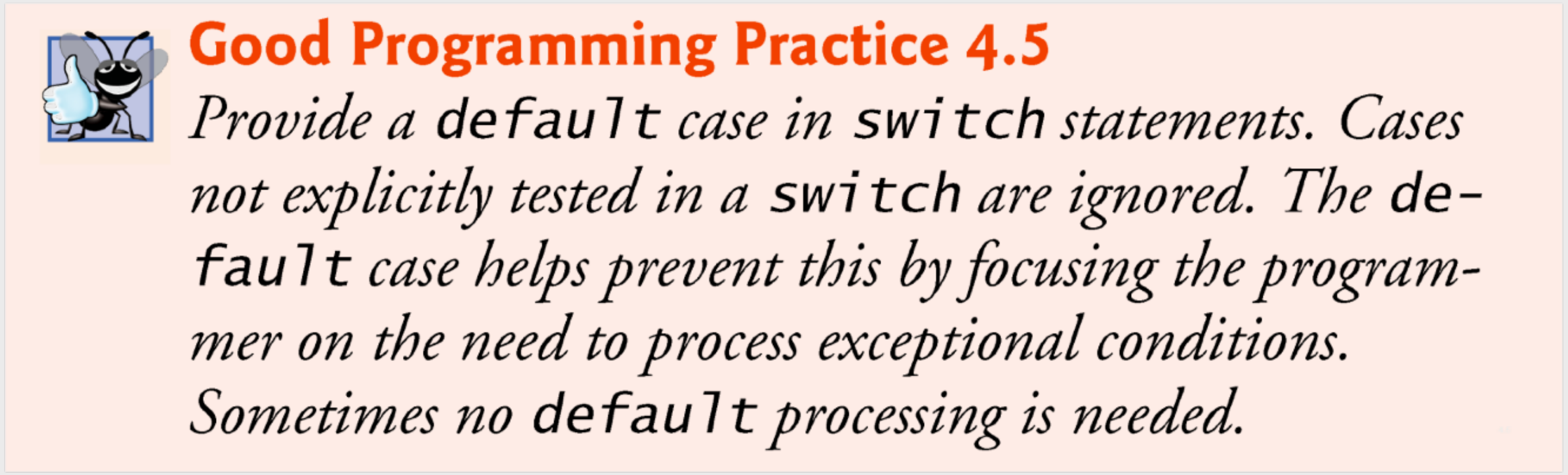


Fig. 4.8 | switch multiple-selection statement with breaks.

`switch` Multiple-Selection Statement (Cont.)

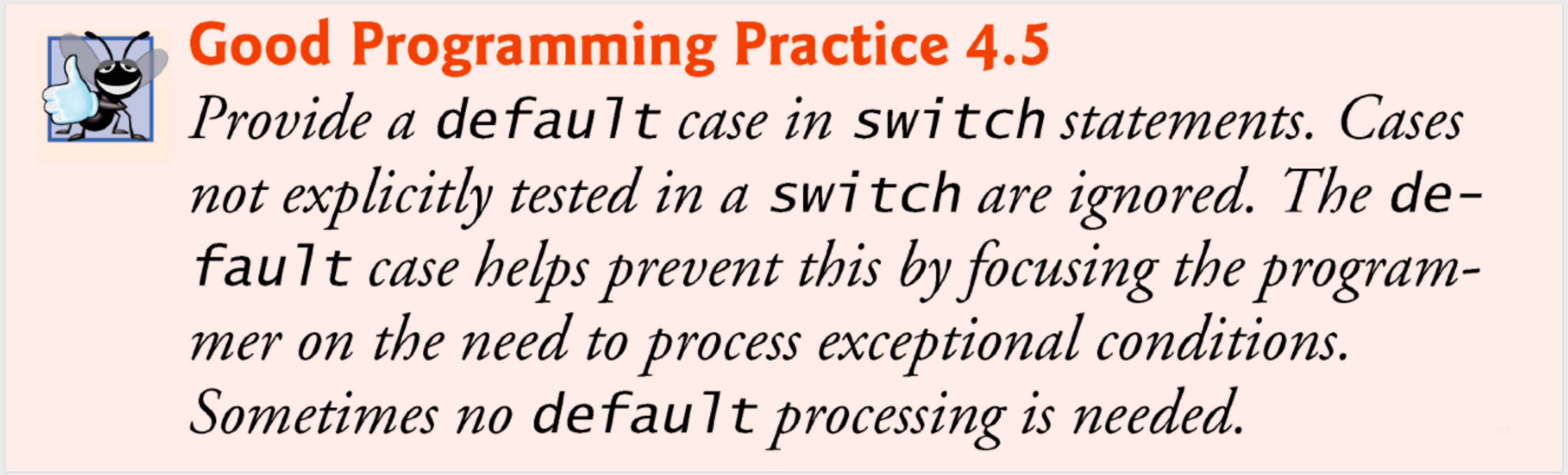
`switch` Multiple-Selection Statement (Cont.)



Good Programming Practice 4.5

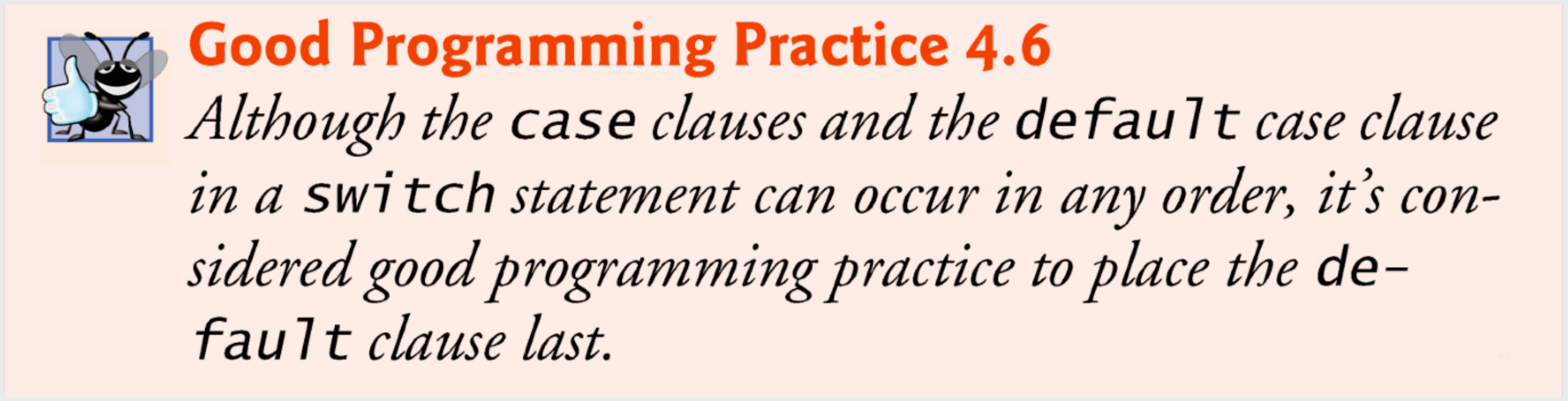
Provide a `default` *case in* `switch` *statements. Cases not explicitly tested in a* `switch` *are ignored. The* `default` *case helps prevent this by focusing the programmer on the need to process exceptional conditions. Sometimes no* `default` *processing is needed.*

`switch` Multiple-Selection Statement (Cont.)



Good Programming Practice 4.5

Provide a `default` *case in* `switch` *statements. Cases not explicitly tested in a* `switch` *are ignored. The* `default` *case helps prevent this by focusing the programmer on the need to process exceptional conditions. Sometimes no* `default` *processing is needed.*



Good Programming Practice 4.6

Although the `case` *clauses and the* `default` *case clause in a* `switch` *statement can occur in any order, it's considered good programming practice to place the* `default` *clause last.*

`switch` Multiple-Selection Statement (Cont.)

- In the switch statement of fig04_07.c, the lines

```
case '\n': /* ignore newlines, */
case '\t': /* tabs, */
case ' ':  /* and spaces in input */
    break; /* exit switch */
```

cause the program to skip newline, tab and blank characters.

`switch` Multiple-Selection Statement (Cont.)

- Characters must be enclosed within single quotes to be recognized as character constants—characters in double quotes are recognized as strings.
- Only character and integer can be used as case label

`do...while` Repetition Statement

- The **`do...while`** repetition statement is similar to the **`while`** statement.
- In the **`while`** statement, the loop-continuation condition is tested at the beginning of the loop before the body of the loop is performed.
- The **`do...while`** statement tests the loop-continuation condition after the loop body is performed.

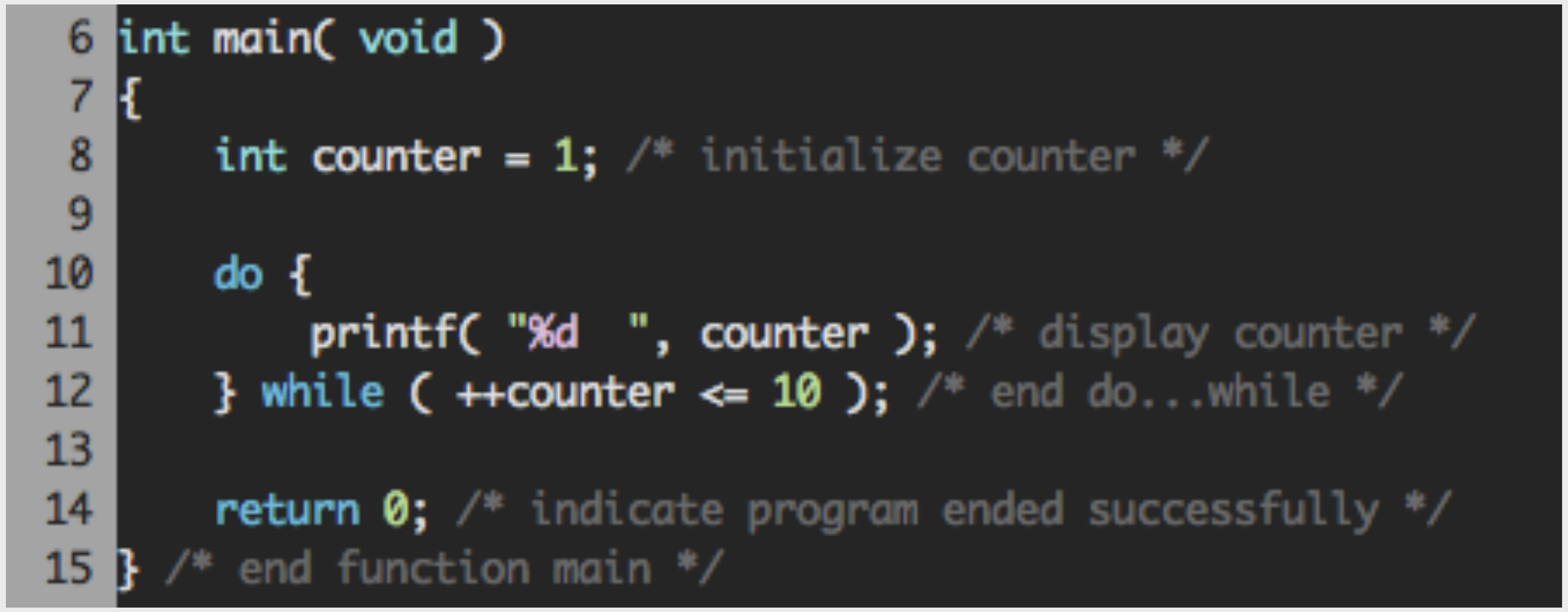
`do...while` Repetition Statement (Cont.)

- A **`do…while`** with no braces around the single-statement body appears as

```
do {
    statements
} while ( condition );
```

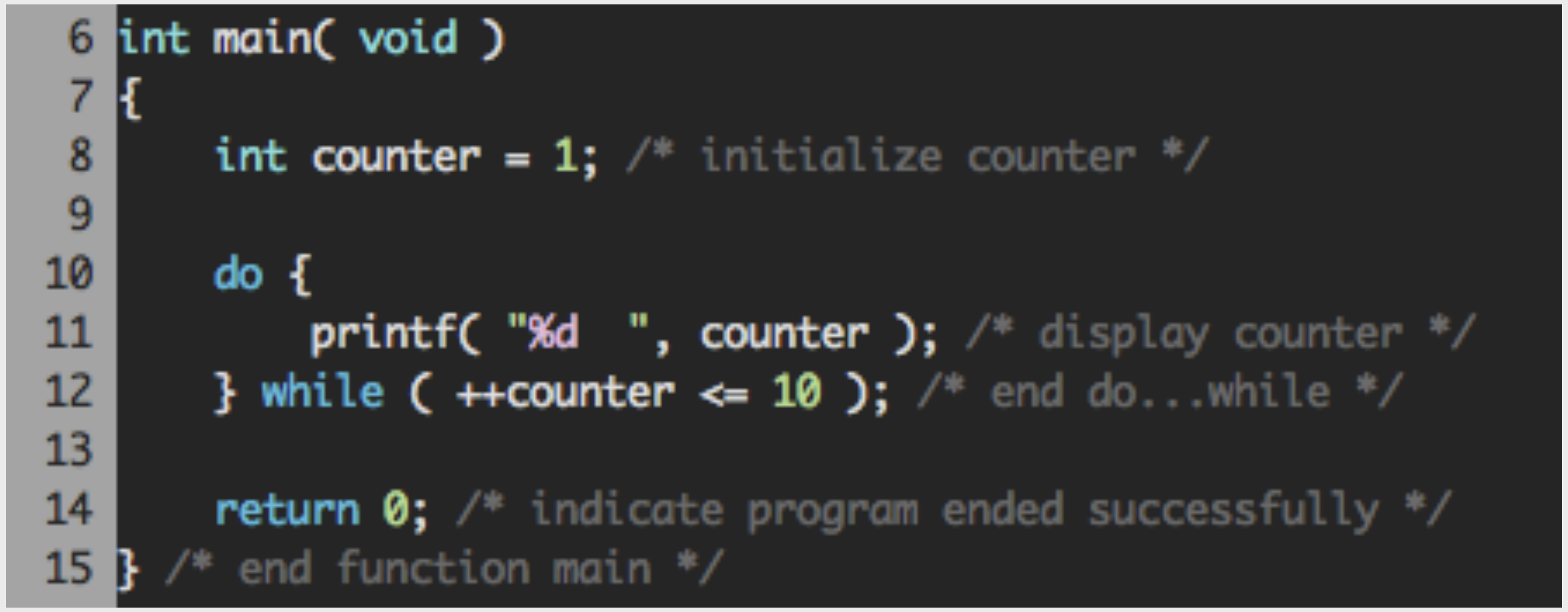
- Difference between **`while`** and **`do...while`**
  - **`do...while`** will be executed at least once

Example: fig04_09.c

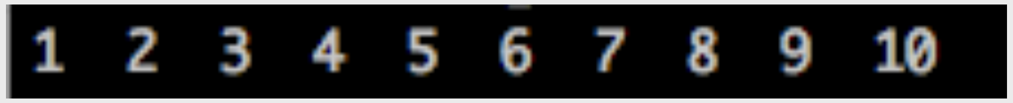


```c
int main( void )
{
    int counter = 1; /* initialize counter */

    do {
        printf( "%d  ", counter ); /* display counter */
    } while ( ++counter <= 10 ); /* end do...while */

    return 0; /* indicate program ended successfully */
} /* end function main */
```

Example: fig04_09.c



```c
int main( void )
{
    int counter = 1; /* initialize counter */

    do {
        printf( "%d  ", counter ); /* display counter */
    } while ( ++counter <= 10 ); /* end do...while */

    return 0; /* indicate program ended successfully */
} /* end function main */
```



```
1  2  3  4  5  6  7  8  9  10
```

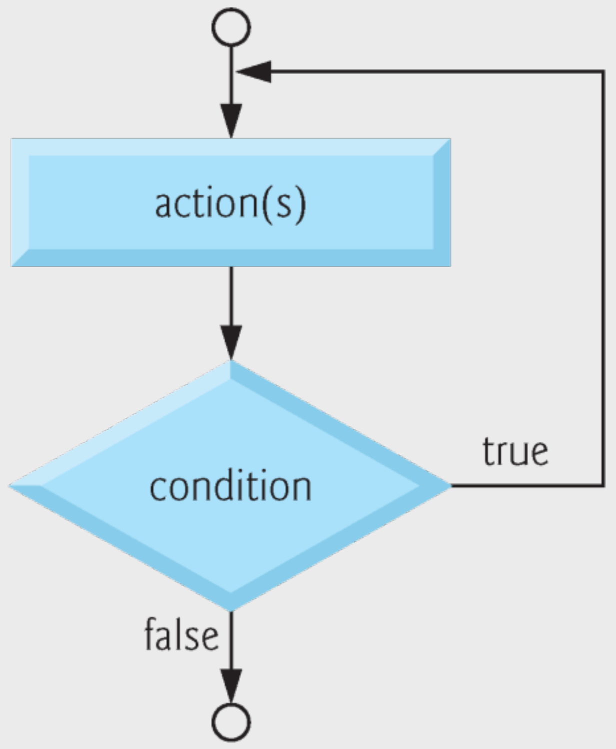
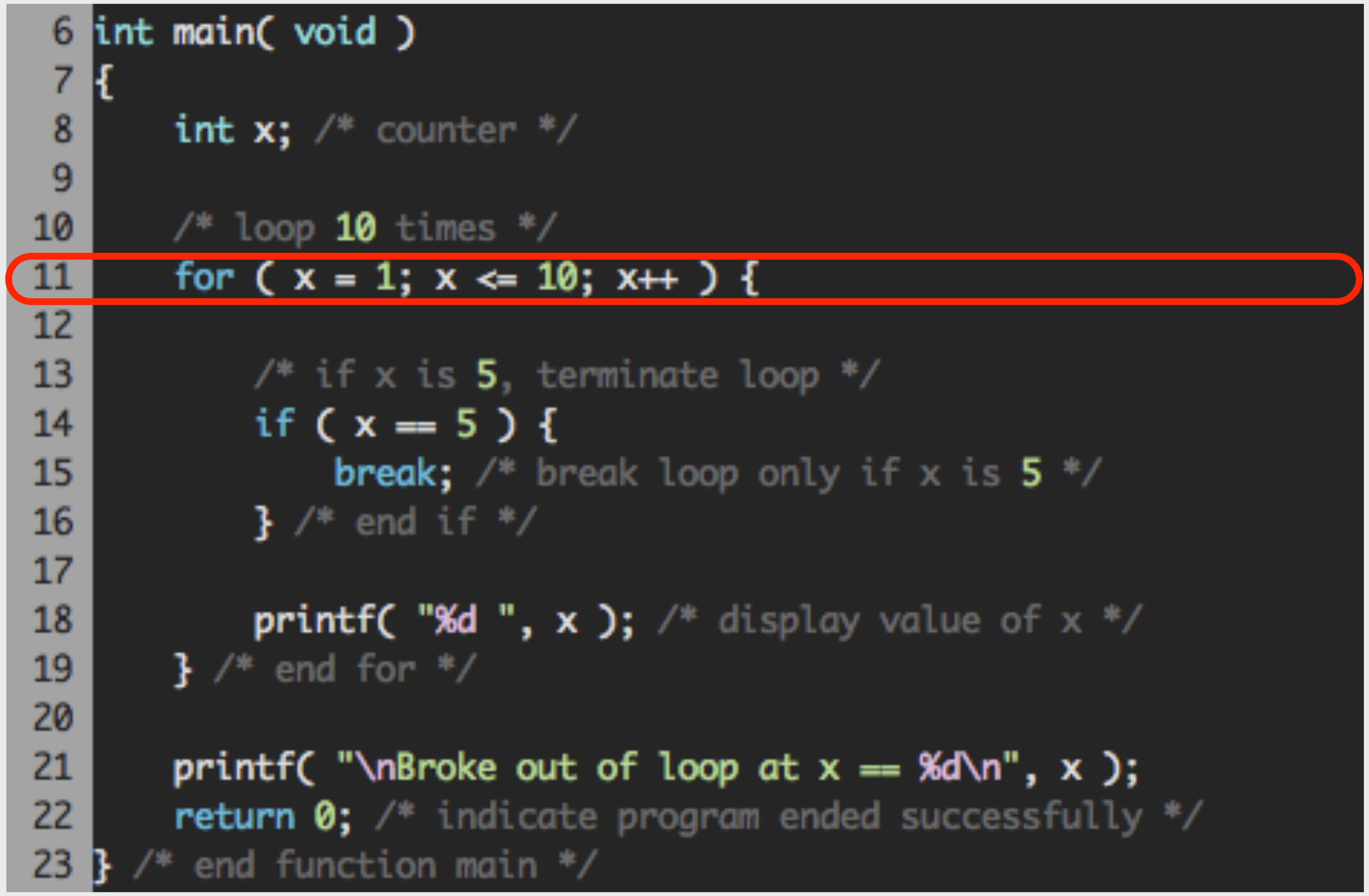


Fig. 4.10 | Flowcharting the `do…while` repetition statement.

`break` and **`continue`** Statements

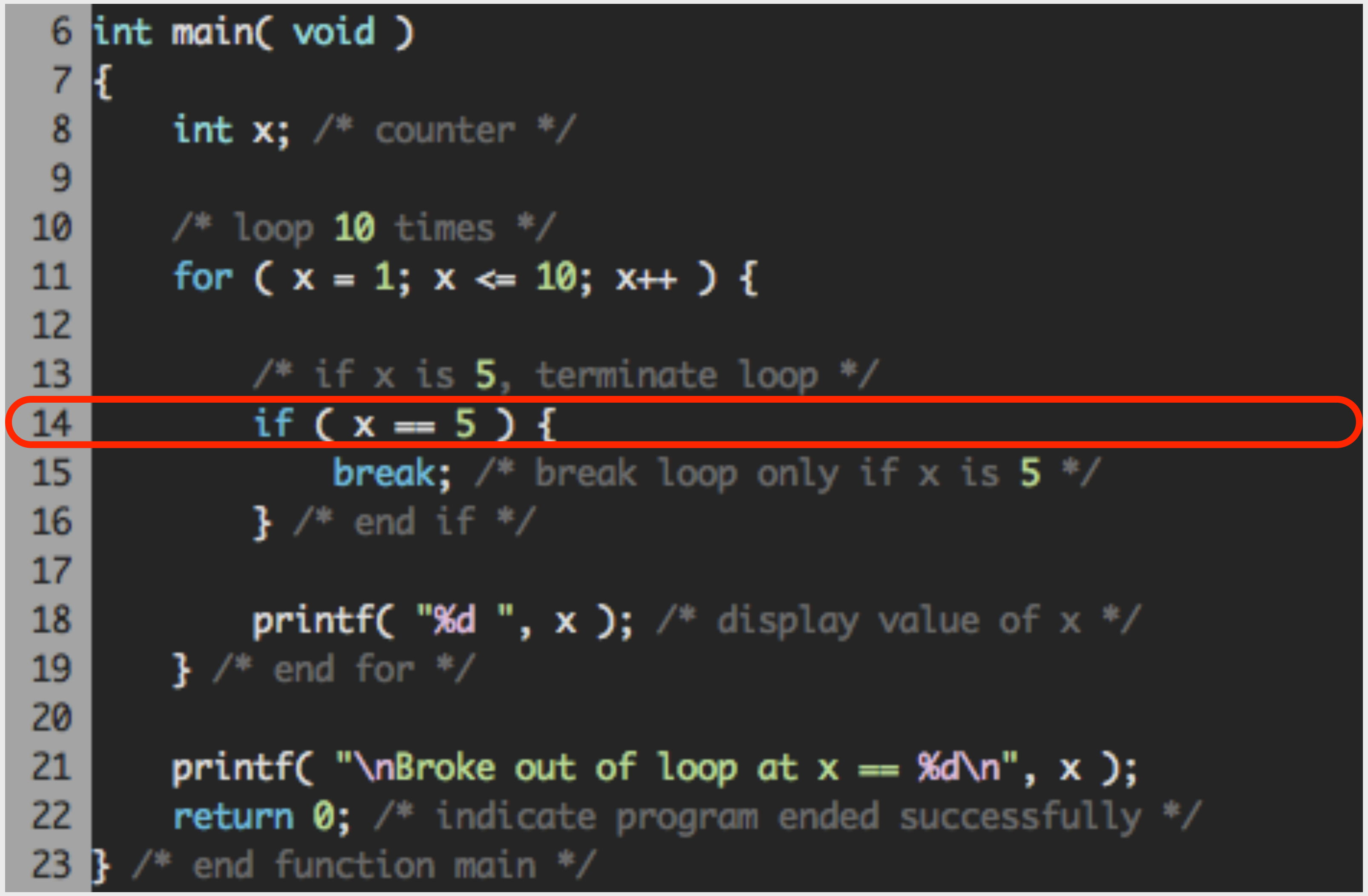
- The **`break`** and **`continue`** statements are used to alter the flow of control.
- Program execution continues with the next statement.
- The **`break`** statement, when executed in a **`while`**, **`for`**, **`do…while`** or **`switch`** statement, causes an immediate exit from that statement.

Example: fig04_11.c



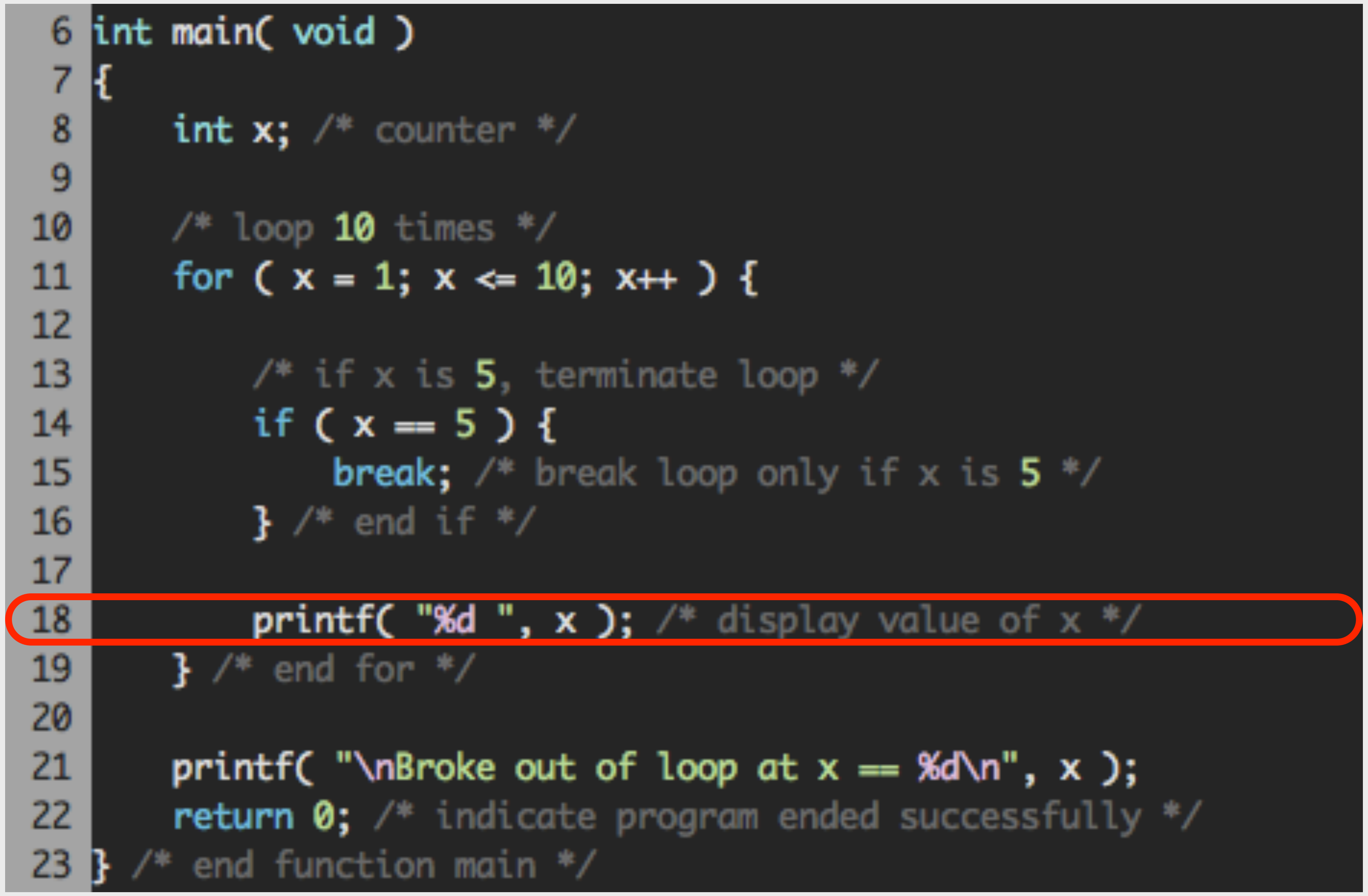
```c
int main( void )
{
    int x; /* counter */

    /* loop 10 times */
    for ( x = 1; x <= 10; x++ ) {

        /* if x is 5, terminate loop */
        if ( x == 5 ) {
            break; /* break loop only if x is 5 */
        } /* end if */

        printf( "%d ", x ); /* display value of x */
    } /* end for */

    printf( "\nBroke out of loop at x == %d\n", x );
    return 0; /* indicate program ended successfully */
} /* end function main */
```

Example: fig04_11.c



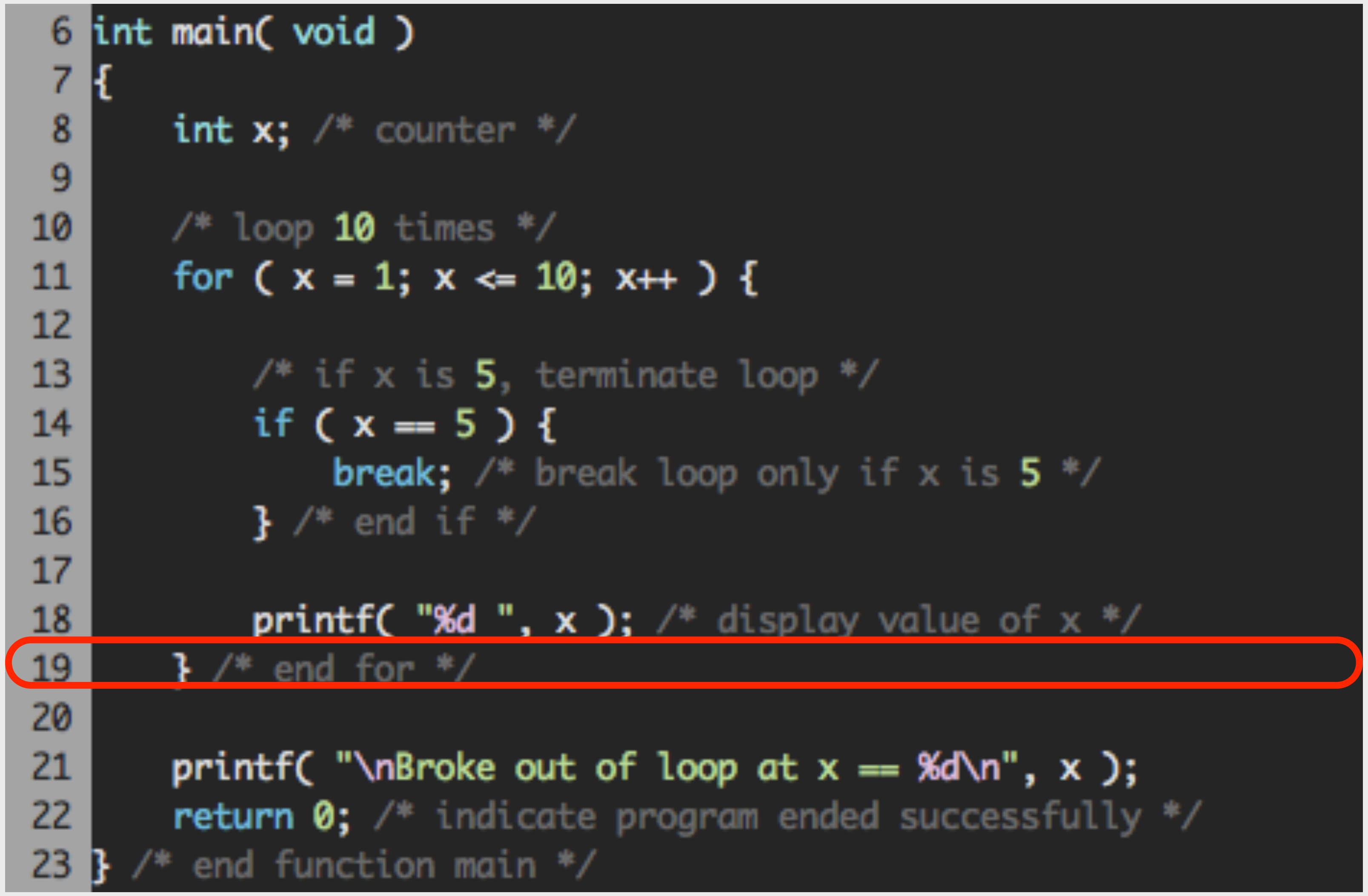
```c
int main( void )
{
    int x; /* counter */

    /* loop 10 times */
    for ( x = 1; x <= 10; x++ ) {

        /* if x is 5, terminate loop */
        if ( x == 5 ) {
            break; /* break loop only if x is 5 */
        } /* end if */

        printf( "%d ", x ); /* display value of x */
    } /* end for */

    printf( "\nBroke out of loop at x == %d\n", x );
    return 0; /* indicate program ended successfully */
} /* end function main */
```

Example: fig04_11.c



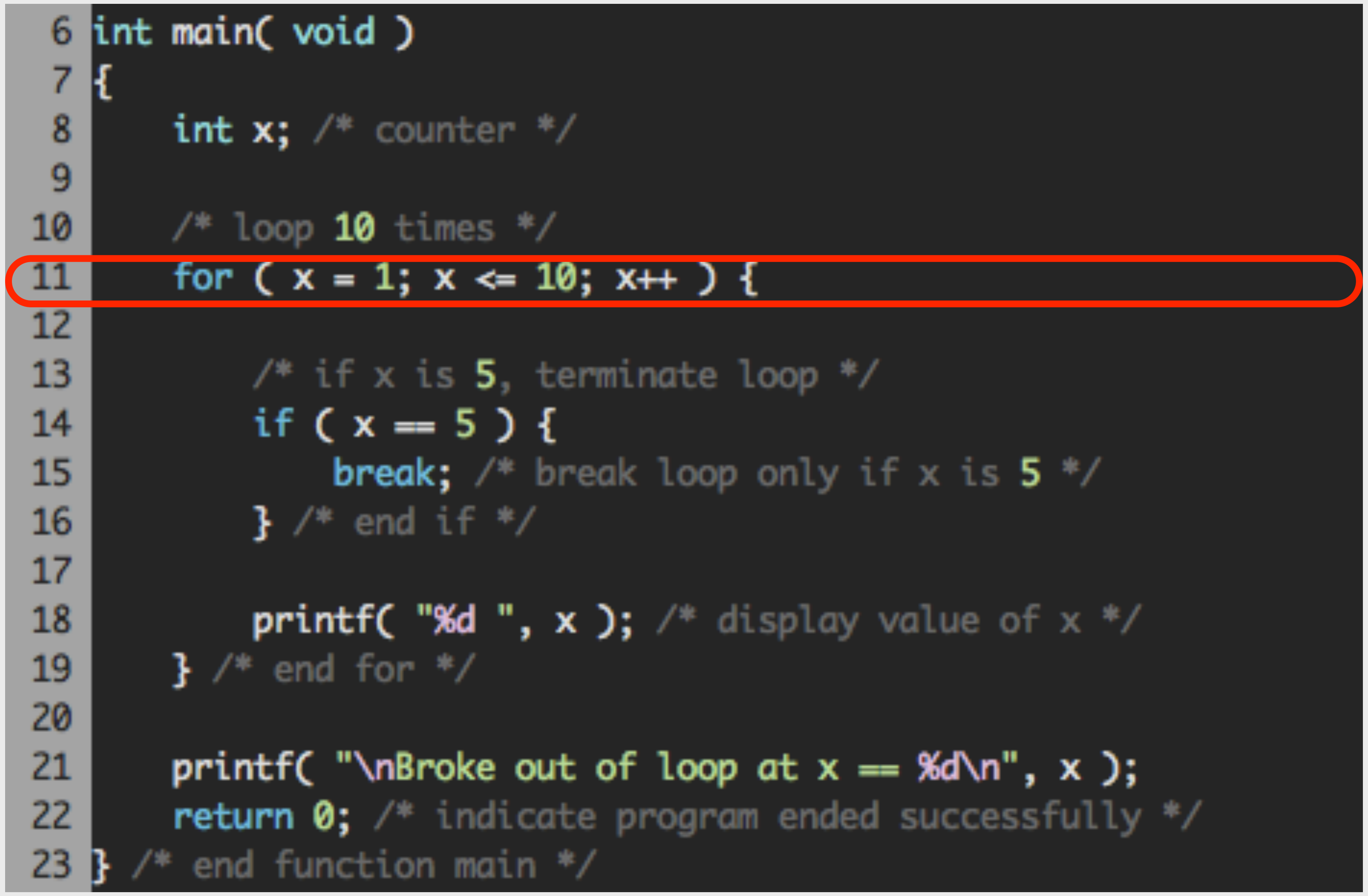
```c
int main( void )
{
    int x; /* counter */

    /* loop 10 times */
    for ( x = 1; x <= 10; x++ ) {

        /* if x is 5, terminate loop */
        if ( x == 5 ) {
            break; /* break loop only if x is 5 */
        } /* end if */

        printf( "%d ", x ); /* display value of x */
    } /* end for */

    printf( "\nBroke out of loop at x == %d\n", x );
    return 0; /* indicate program ended successfully */
} /* end function main */
```

Example: fig04_11.c



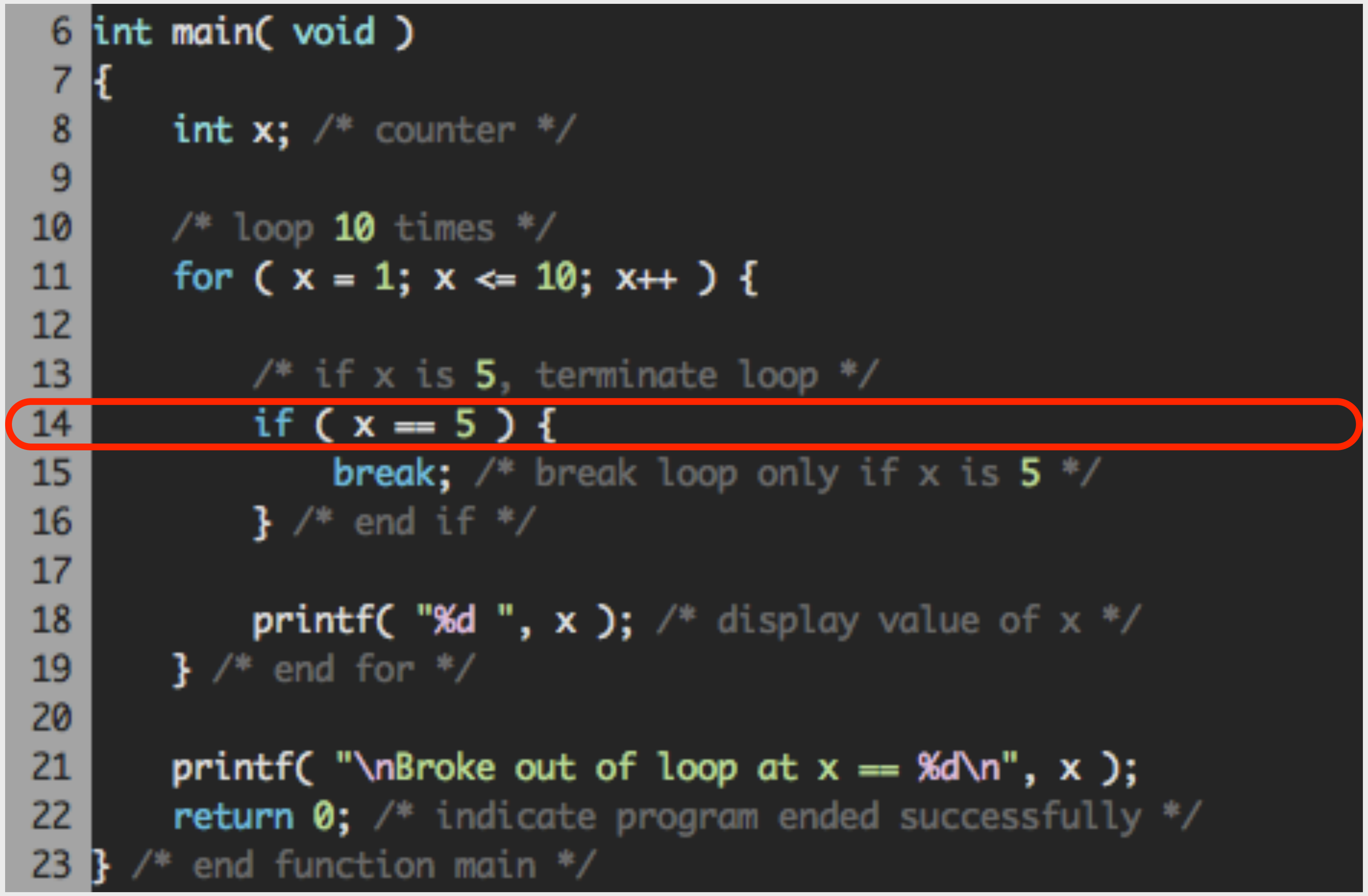
```c
int main( void )
{
    int x; /* counter */

    /* loop 10 times */
    for ( x = 1; x <= 10; x++ ) {

        /* if x is 5, terminate loop */
        if ( x == 5 ) {
            break; /* break loop only if x is 5 */
        } /* end if */

        printf( "%d ", x ); /* display value of x */
    } /* end for */

    printf( "\nBroke out of loop at x == %d\n", x );
    return 0; /* indicate program ended successfully */
} /* end function main */
```

Example: fig04_11.c



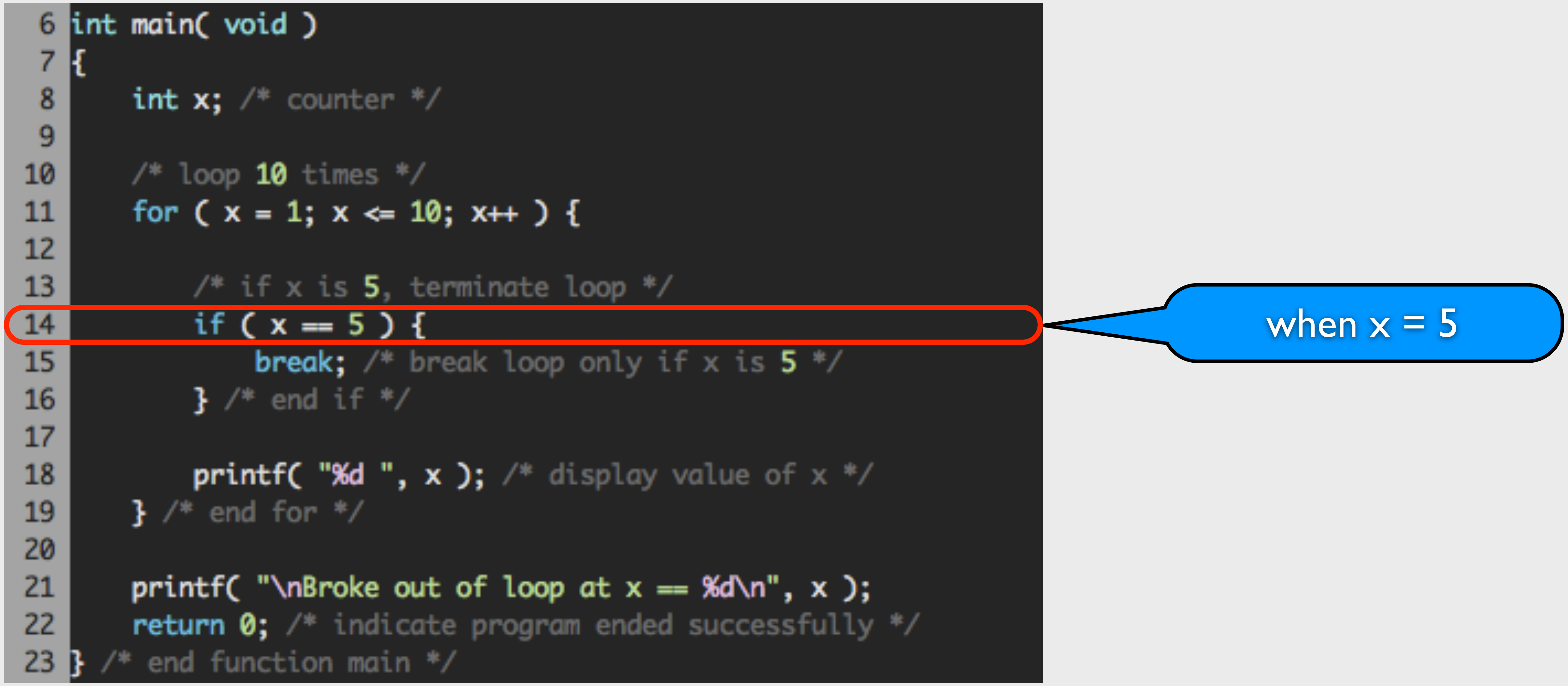
```
int main( void )
{
    int x; /* counter */

    /* loop 10 times */
    for ( x = 1; x <= 10; x++ ) {

        /* if x is 5, terminate loop */
        if ( x == 5 ) {
            break; /* break loop only if x is 5 */
        } /* end if */

        printf( "%d ", x ); /* display value of x */
    } /* end for */

    printf( "\nBroke out of loop at x == %d\n", x );
    return 0; /* indicate program ended successfully */
} /* end function main */
```

Example: fig04_11.c



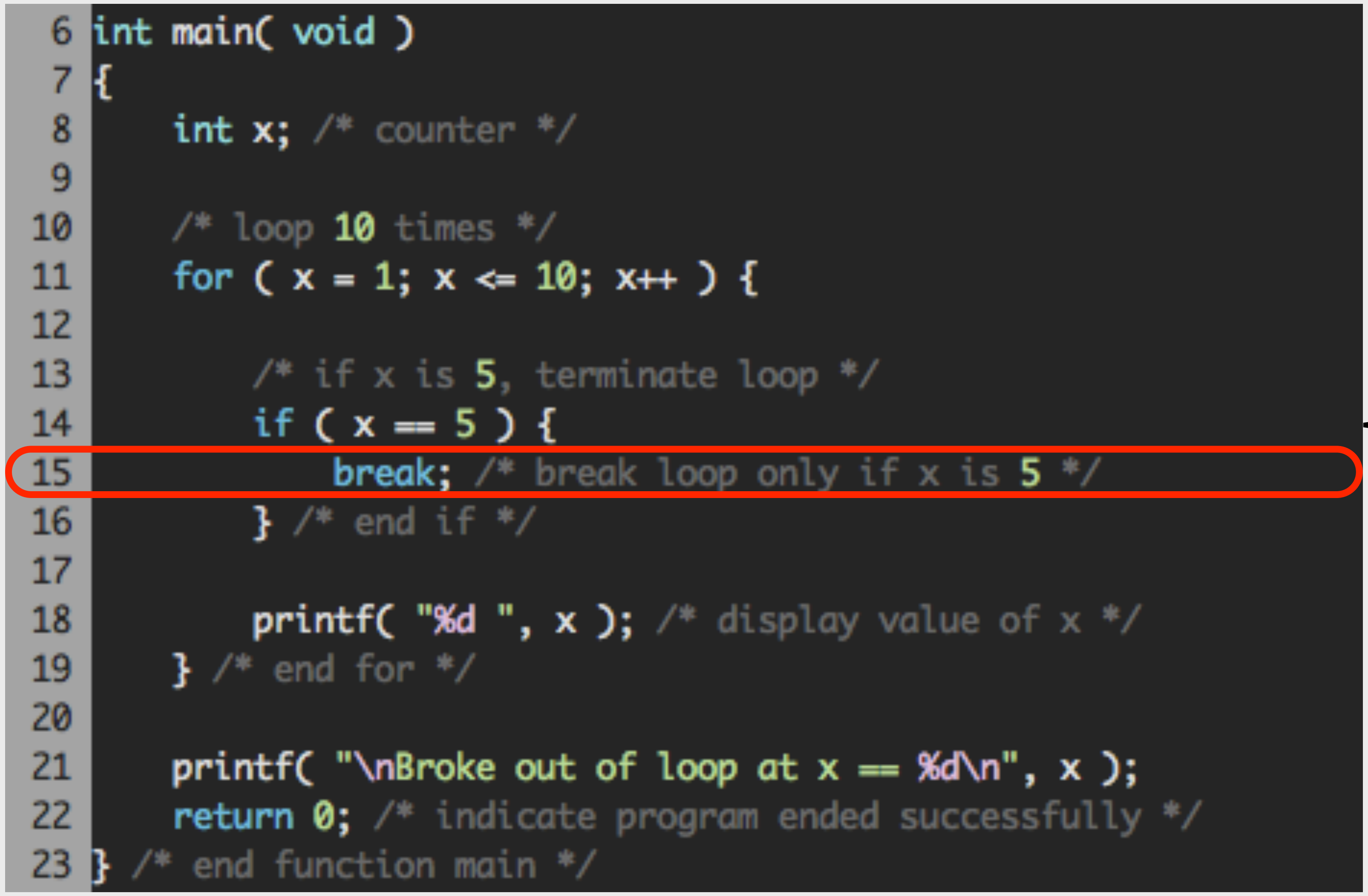
```c
int main( void )
{
    int x; /* counter */

    /* loop 10 times */
    for ( x = 1; x <= 10; x++ ) {

        /* if x is 5, terminate loop */
        if ( x == 5 ) {
            break; /* break loop only if x is 5 */
        } /* end if */

        printf( "%d ", x ); /* display value of x */
    } /* end for */

    printf( "\nBroke out of loop at x == %d\n", x );
    return 0; /* indicate program ended successfully */
} /* end function main */
```

Example: fig04_11.c



```
int main( void )
{
    int x; /* counter */

    /* loop 10 times */
    for ( x = 1; x <= 10; x++ ) {

        /* if x is 5, terminate loop */
        if ( x == 5 ) {
            break; /* break loop only if x is 5 */
        } /* end if */

        printf( "%d ", x ); /* display value of x */
    } /* end for */

    printf( "\nBroke out of loop at x == %d\n", x );
    return 0; /* indicate program ended successfully */
} /* end function main */
```

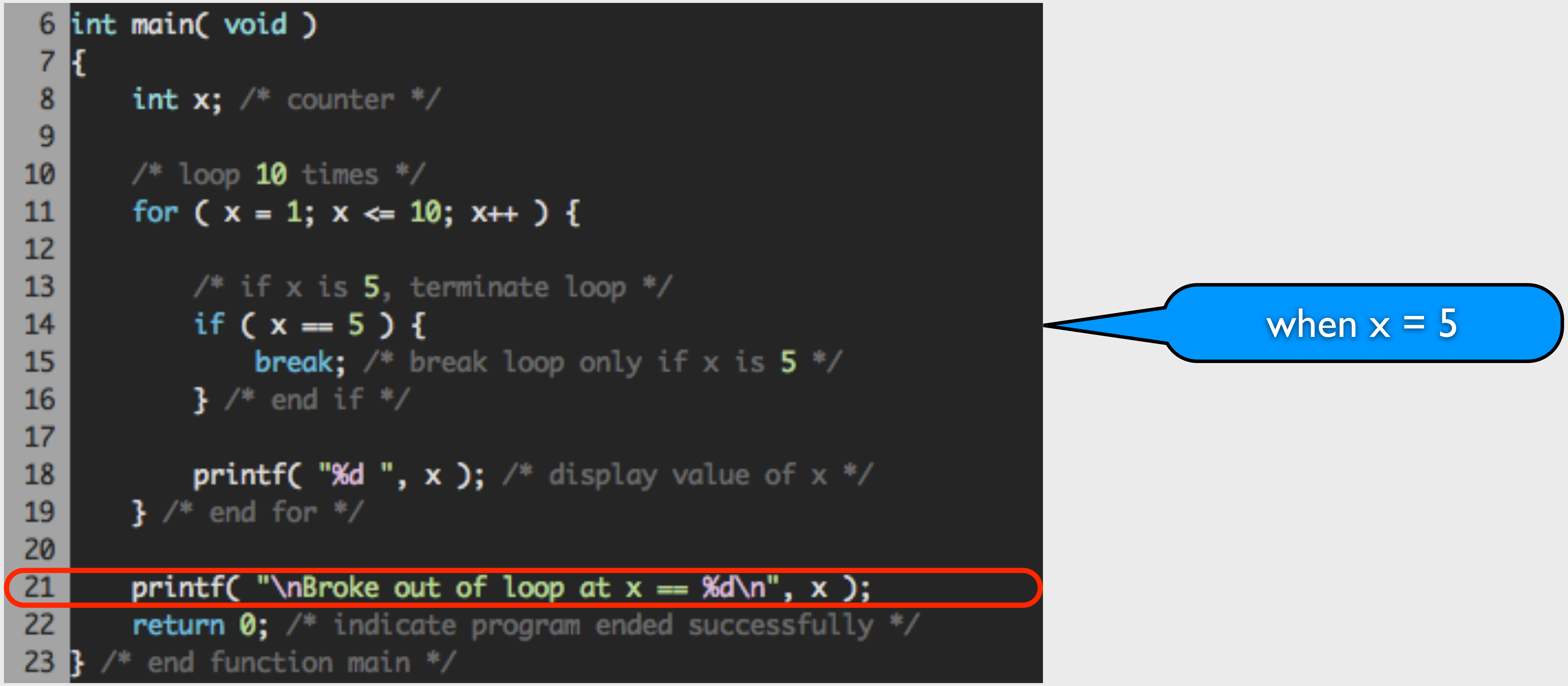
Example: fig04_11.c



```c
int main( void )
{
    int x; /* counter */

    /* loop 10 times */
    for ( x = 1; x <= 10; x++ ) {

        /* if x is 5, terminate loop */
        if ( x == 5 ) {
            break; /* break loop only if x is 5 */
        } /* end if */

        printf( "%d ", x ); /* display value of x */
    } /* end for */

    printf( "\nBroke out of loop at x == %d\n", x );
    return 0; /* indicate program ended successfully */
} /* end function main */
```

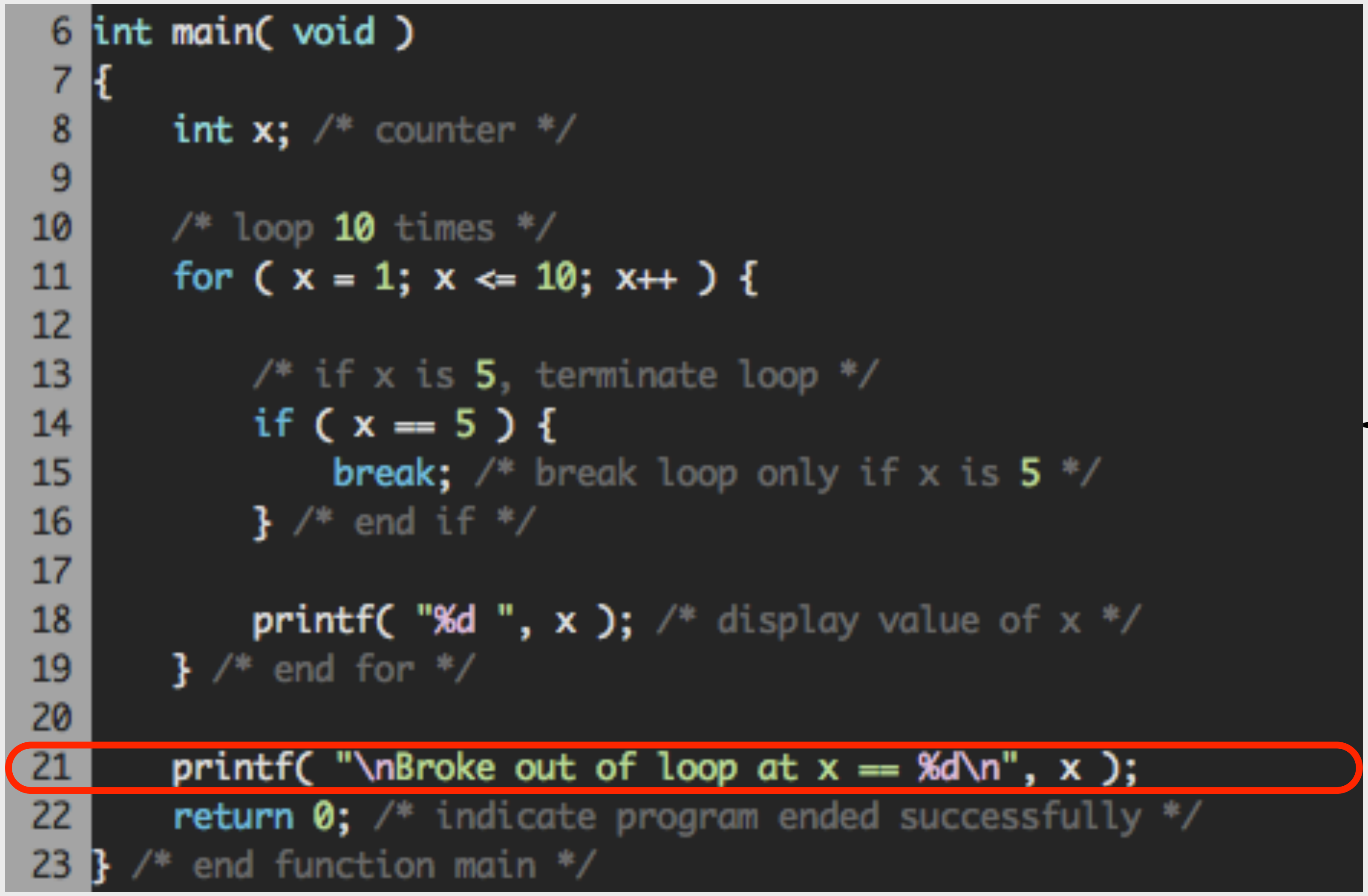
when x = 5

Example: fig04_11.c

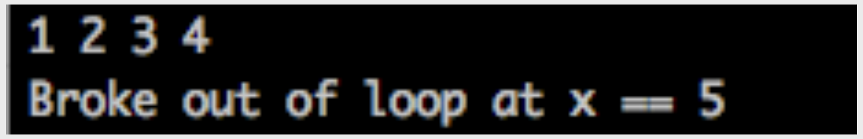


```
int main( void )
{
    int x; /* counter */

    /* loop 10 times */
    for ( x = 1; x <= 10; x++ ) {

        /* if x is 5, terminate loop */
        if ( x == 5 ) {
            break; /* break loop only if x is 5 */
        } /* end if */

        printf( "%d ", x ); /* display value of x */
    } /* end for */

    printf( "\nBroke out of loop at x == %d\n", x );
    return 0; /* indicate program ended successfully */
} /* end function main */
```

Example: fig04_11.c



```
int main( void )
{
    int x; /* counter */

    /* loop 10 times */
    for ( x = 1; x <= 10; x++ ) {

        /* if x is 5, terminate loop */
        if ( x == 5 ) {
            break; /* break loop only if x is 5 */
        } /* end if */

        printf( "%d ", x ); /* display value of x */
    } /* end for */

    printf( "\nBroke out of loop at x == %d\n", x );
    return 0; /* indicate program ended successfully */
} /* end function main */
```



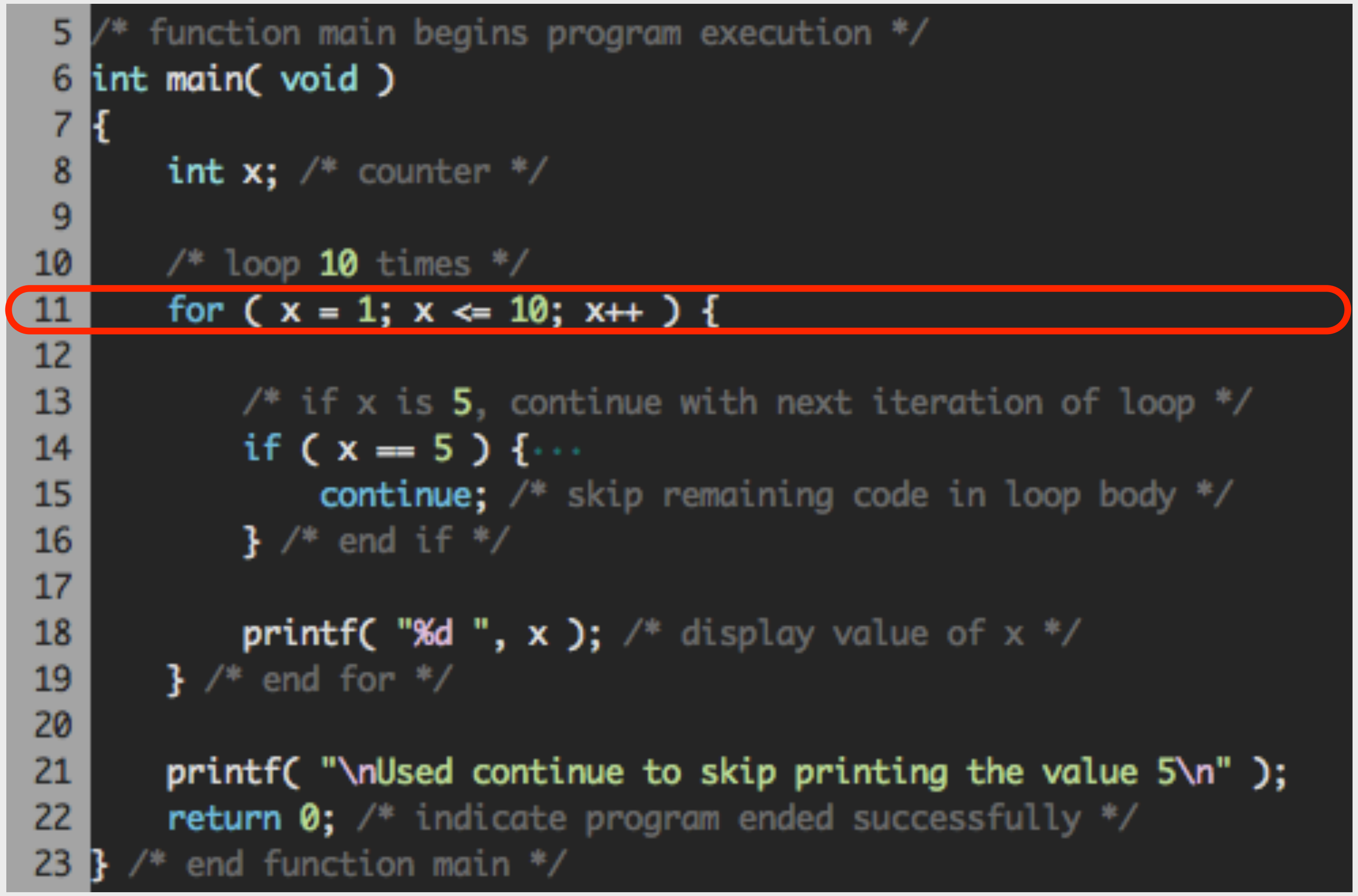
```
1 2 3 4
Broke out of loop at x == 5
```

when x = 5

`break` and `continue` Statements (Cont.)

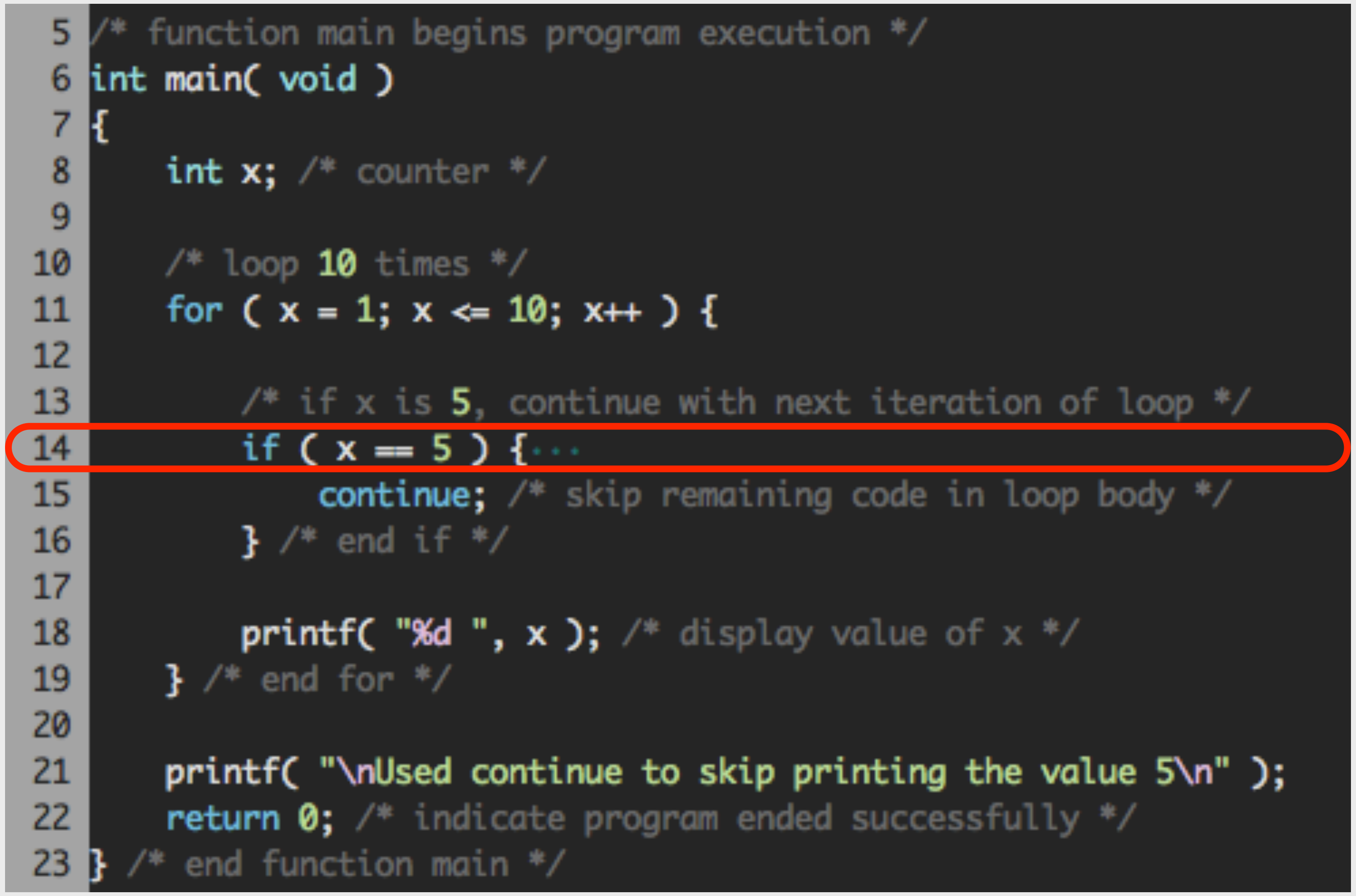
- The **`continue`** statement, when executed in a **`while`**, **`for`** or **`do…while`** statement
    - skips the remaining statements in the body of that control statement and performs the next iteration of the loop.

Example: fig04_12.c



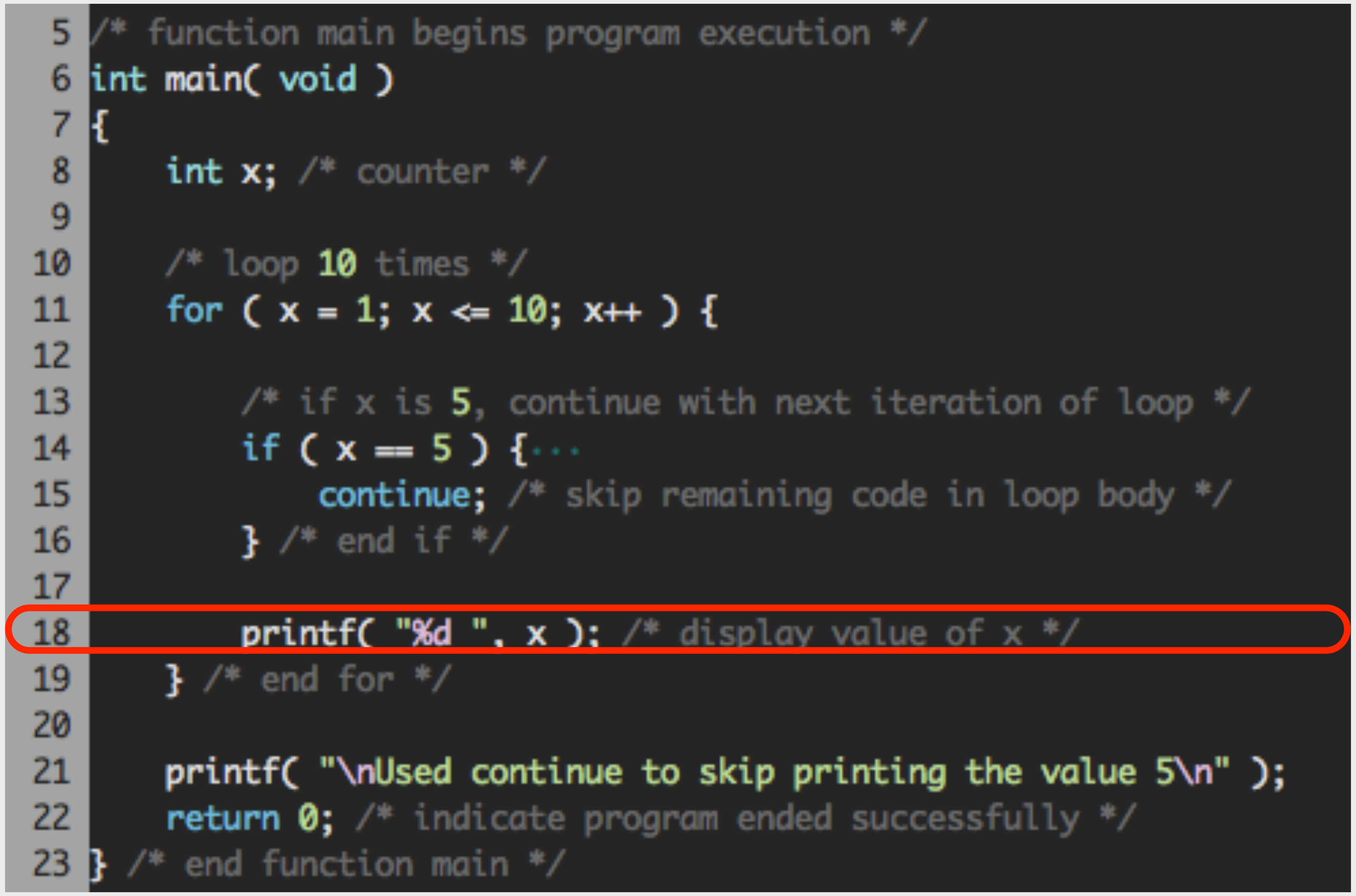
```c
/* function main begins program execution */
int main( void )
{
    int x; /* counter */

    /* loop 10 times */
    for ( x = 1; x <= 10; x++ ) {

        /* if x is 5, continue with next iteration of loop */
        if ( x == 5 ) {
            continue; /* skip remaining code in loop body */
        } /* end if */

        printf( "%d ", x ); /* display value of x */
    } /* end for */

    printf( "\nUsed continue to skip printing the value 5\n" );
    return 0; /* indicate program ended successfully */
} /* end function main */
```

Example: fig04_12.c



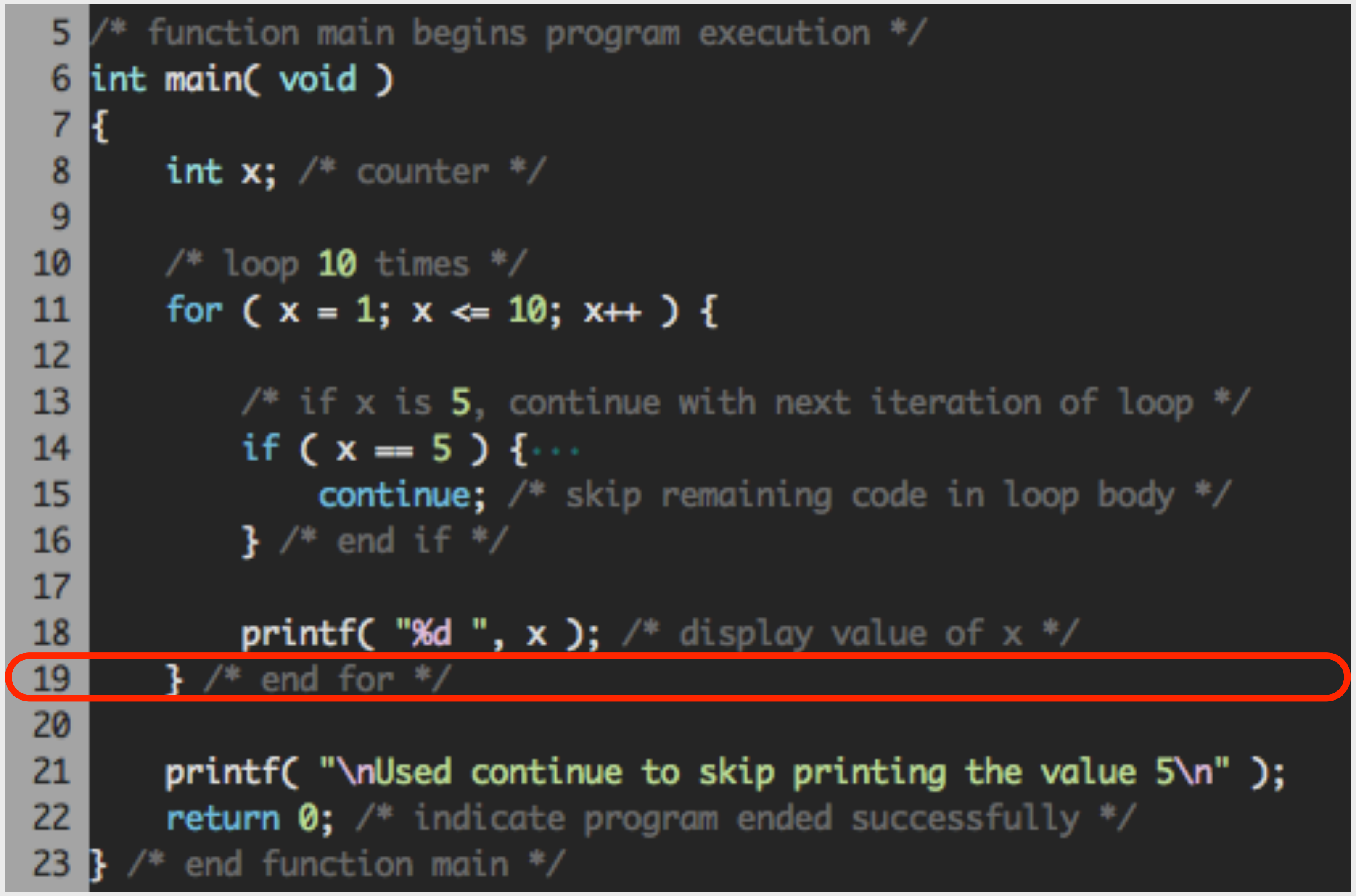
```
/* function main begins program execution */
int main( void )
{
    int x; /* counter */

    /* loop 10 times */
    for ( x = 1; x <= 10; x++ ) {

        /* if x is 5, continue with next iteration of loop */
        if ( x == 5 ) {···
            continue; /* skip remaining code in loop body */
        } /* end if */

        printf( "%d ", x ); /* display value of x */
    } /* end for */

    printf( "\nUsed continue to skip printing the value 5\n" );
    return 0; /* indicate program ended successfully */
} /* end function main */
```

Example: fig04_12.c



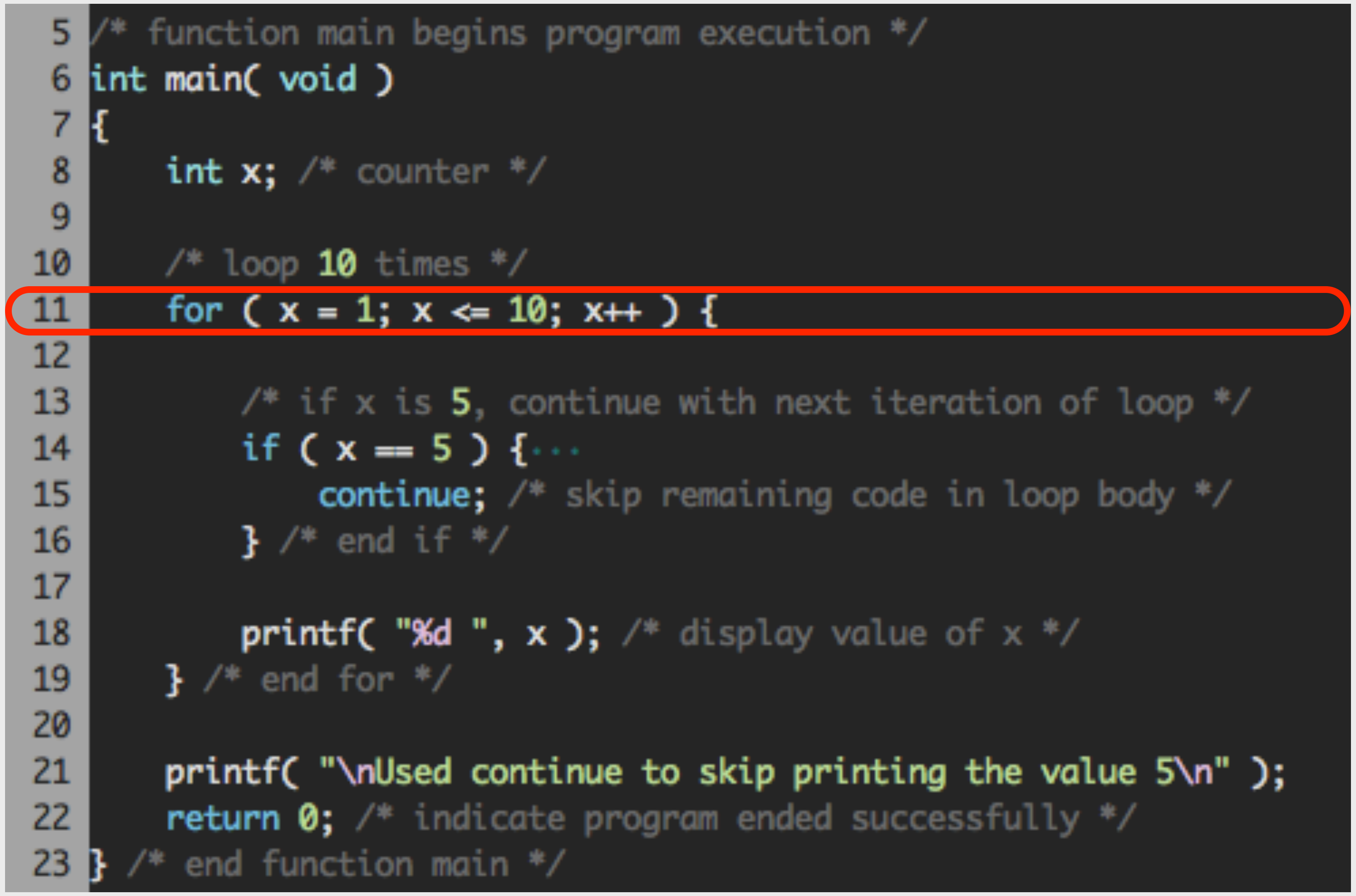
```
/* function main begins program execution */
int main( void )
{
    int x; /* counter */

    /* loop 10 times */
    for ( x = 1; x <= 10; x++ ) {

        /* if x is 5, continue with next iteration of loop */
        if ( x == 5 ) {
            continue; /* skip remaining code in loop body */
        } /* end if */

        printf( "%d ", x ); /* display value of x */
    } /* end for */

    printf( "\nUsed continue to skip printing the value 5\n" );
    return 0; /* indicate program ended successfully */
} /* end function main */
```

Example: fig04_12.c



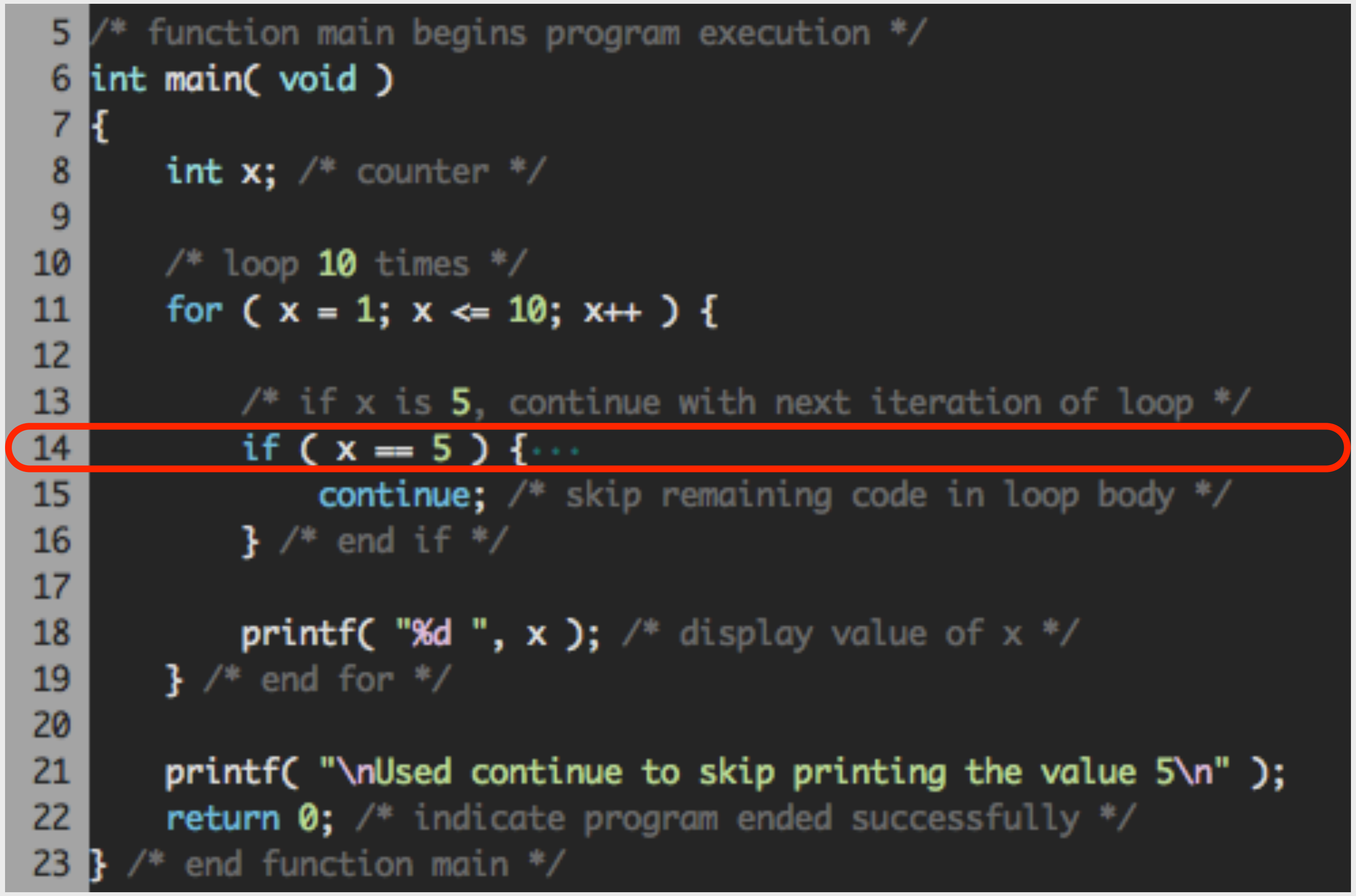
```
/* function main begins program execution */
int main( void )
{
    int x; /* counter */

    /* loop 10 times */
    for ( x = 1; x <= 10; x++ ) {

        /* if x is 5, continue with next iteration of loop */
        if ( x == 5 ) {···
            continue; /* skip remaining code in loop body */
        } /* end if */

        printf( "%d ", x ); /* display value of x */
    } /* end for */

    printf( "\nUsed continue to skip printing the value 5\n" );
    return 0; /* indicate program ended successfully */
} /* end function main */
```

Example: fig04_12.c



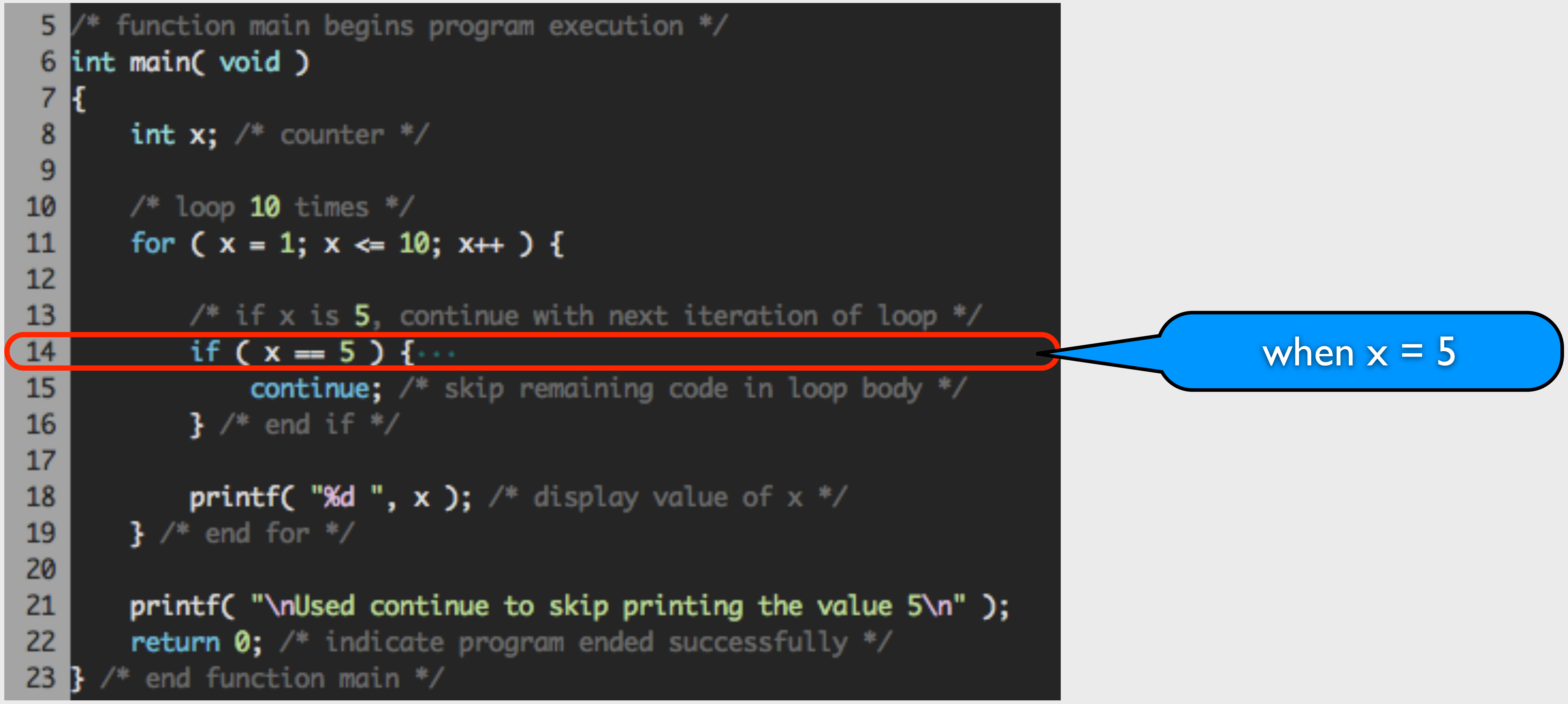
```c
/* function main begins program execution */
int main( void )
{
    int x; /* counter */

    /* loop 10 times */
    for ( x = 1; x <= 10; x++ ) {

        /* if x is 5, continue with next iteration of loop */
        if ( x == 5 ) {
            continue; /* skip remaining code in loop body */
        } /* end if */

        printf( "%d ", x ); /* display value of x */
    } /* end for */

    printf( "\nUsed continue to skip printing the value 5\n" );
    return 0; /* indicate program ended successfully */
} /* end function main */
```

Example: fig04_12.c



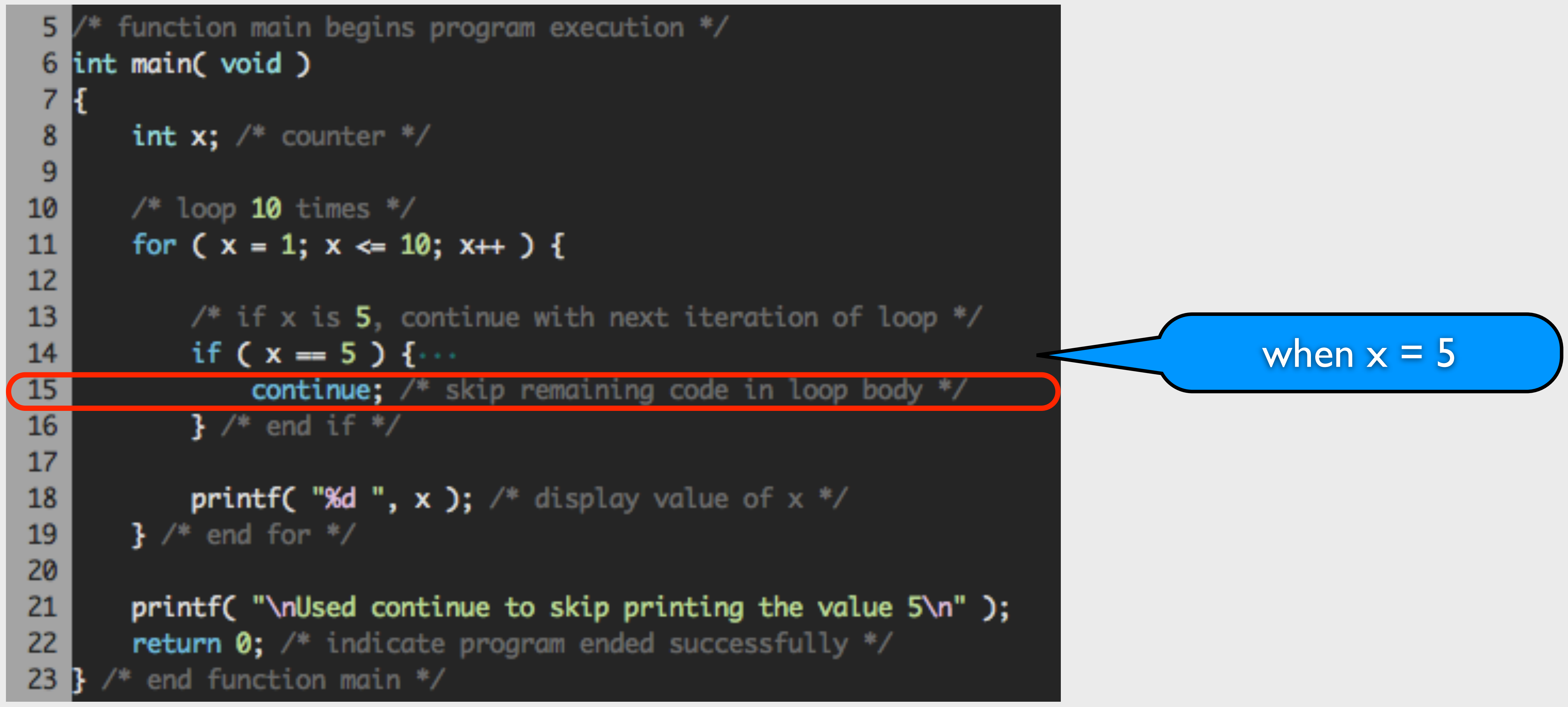
```c
/* function main begins program execution */
int main( void )
{
    int x; /* counter */

    /* loop 10 times */
    for ( x = 1; x <= 10; x++ ) {

        /* if x is 5, continue with next iteration of loop */
        if ( x == 5 ) {
            continue; /* skip remaining code in loop body */
        } /* end if */

        printf( "%d ", x ); /* display value of x */
    } /* end for */

    printf( "\nUsed continue to skip printing the value 5\n" );
    return 0; /* indicate program ended successfully */
} /* end function main */
```

Example: fig04_12.c



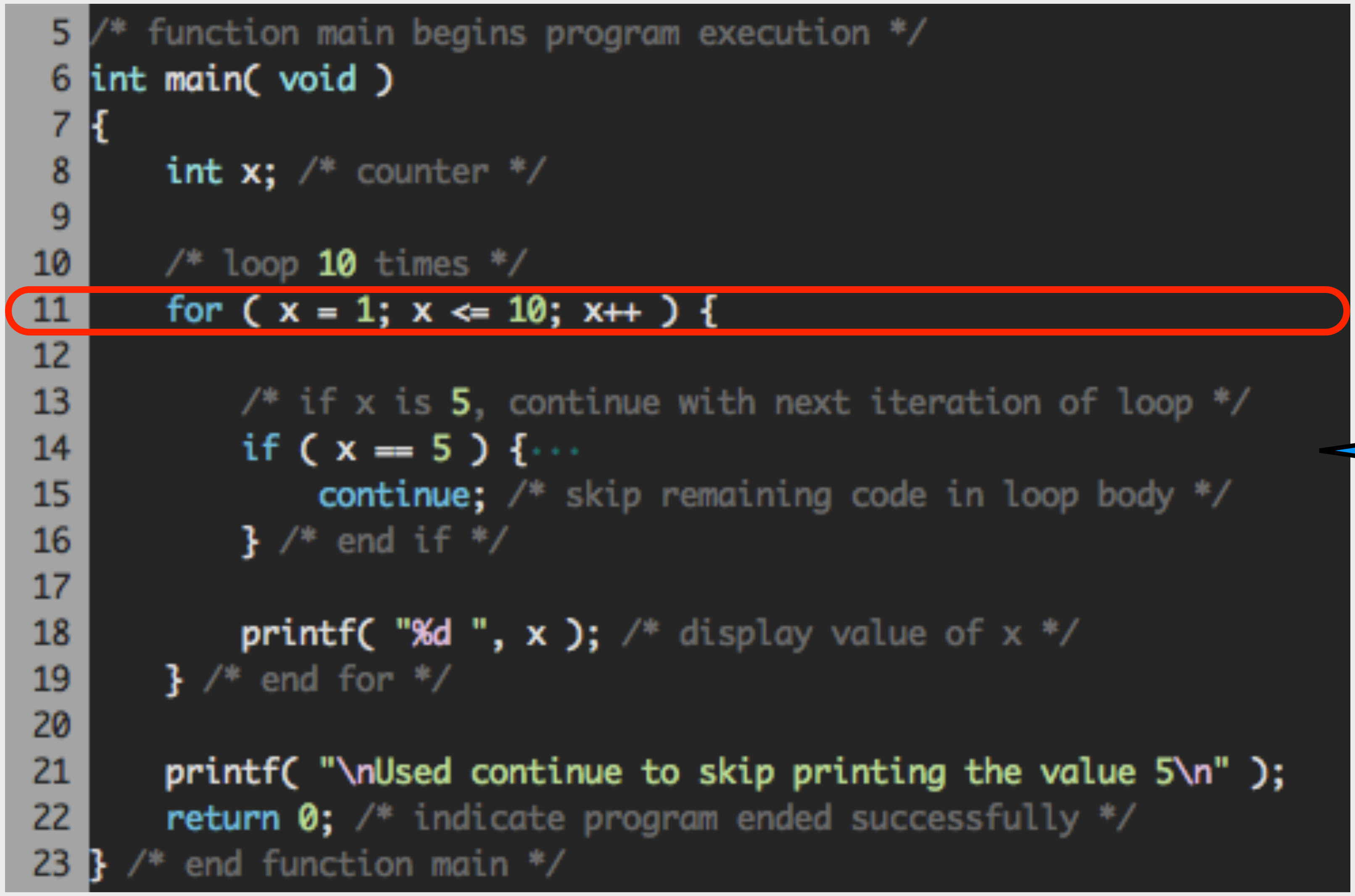
```
/* function main begins program execution */
int main( void )
{
    int x; /* counter */

    /* loop 10 times */
    for ( x = 1; x <= 10; x++ ) {

        /* if x is 5, continue with next iteration of loop */
        if ( x == 5 ) {···
            continue; /* skip remaining code in loop body */
        } /* end if */

        printf( "%d ", x ); /* display value of x */
    } /* end for */

    printf( "\nUsed continue to skip printing the value 5\n" );
    return 0; /* indicate program ended successfully */
} /* end function main */
```

Example: fig04_12.c



```
/* function main begins program execution */
int main( void )
{
    int x; /* counter */

    /* loop 10 times */
    for ( x = 1; x <= 10; x++ ) {

        /* if x is 5, continue with next iteration of loop */
        if ( x == 5 ) {···
            continue; /* skip remaining code in loop body */
        } /* end if */

        printf( "%d ", x ); /* display value of x */
    } /* end for */

    printf( "\nUsed continue to skip printing the value 5\n" );
    return 0; /* indicate program ended successfully */
} /* end function main */
```

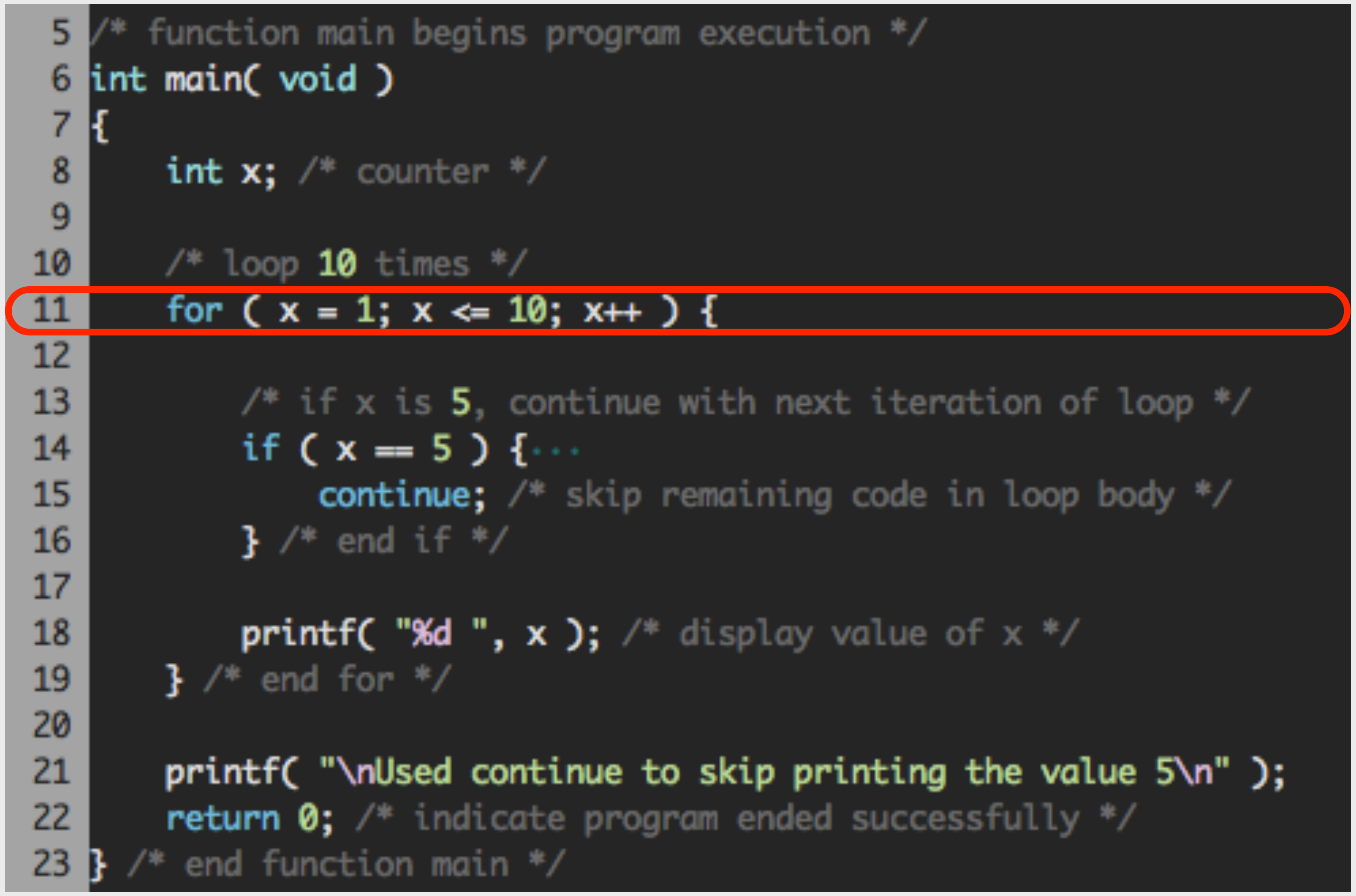
Example: fig04_12.c



```
/* function main begins program execution */
int main( void )
{
    int x; /* counter */

    /* loop 10 times */
    for ( x = 1; x <= 10; x++ ) {

        /* if x is 5, continue with next iteration of loop */
        if ( x == 5 ) {
            continue; /* skip remaining code in loop body */
        } /* end if */

        printf( "%d ", x ); /* display value of x */
    } /* end for */

    printf( "\nUsed continue to skip printing the value 5\n" );
    return 0; /* indicate program ended successfully */
} /* end function main */
```

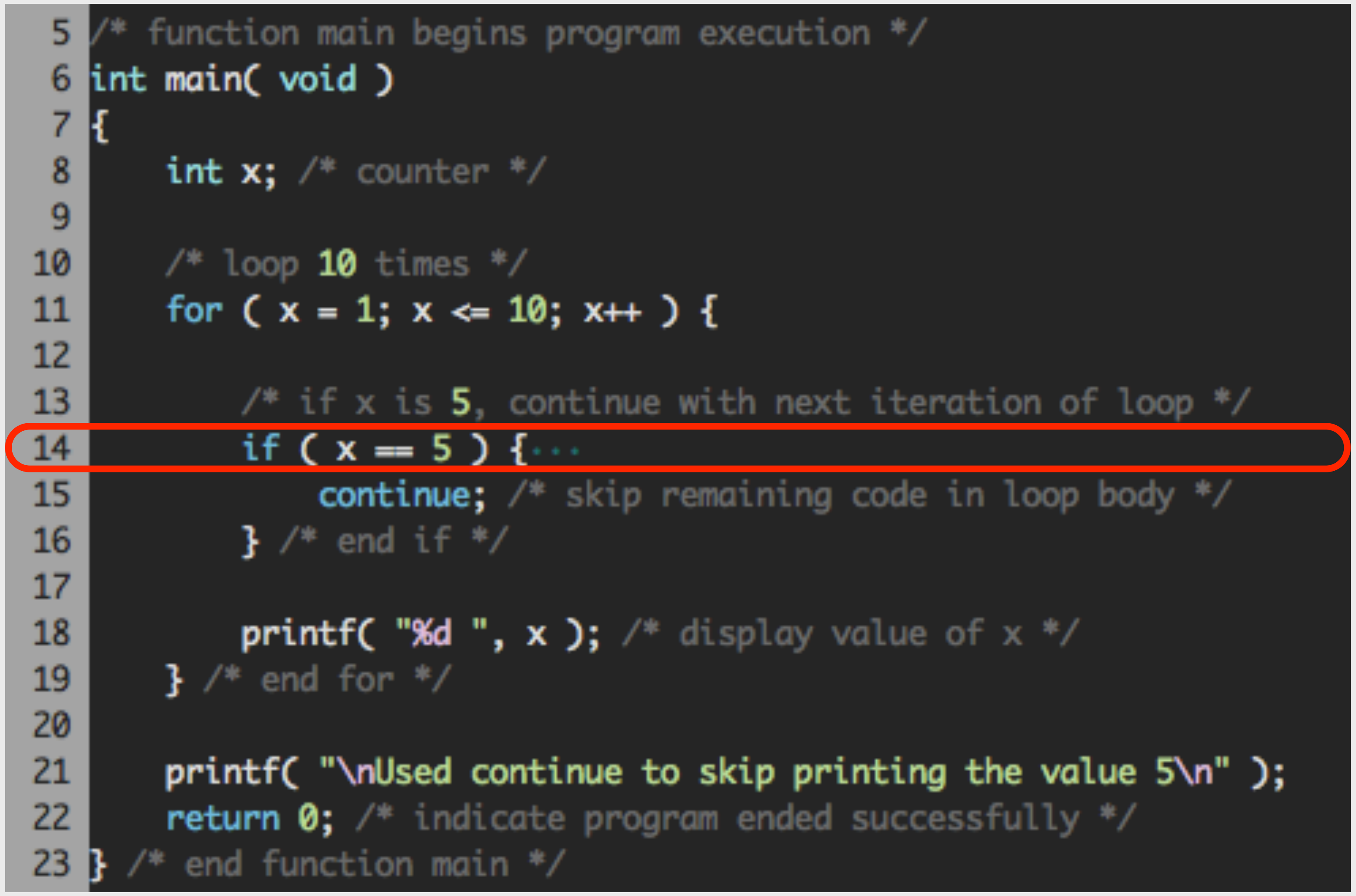
when x = 5

Example: fig04_12.c



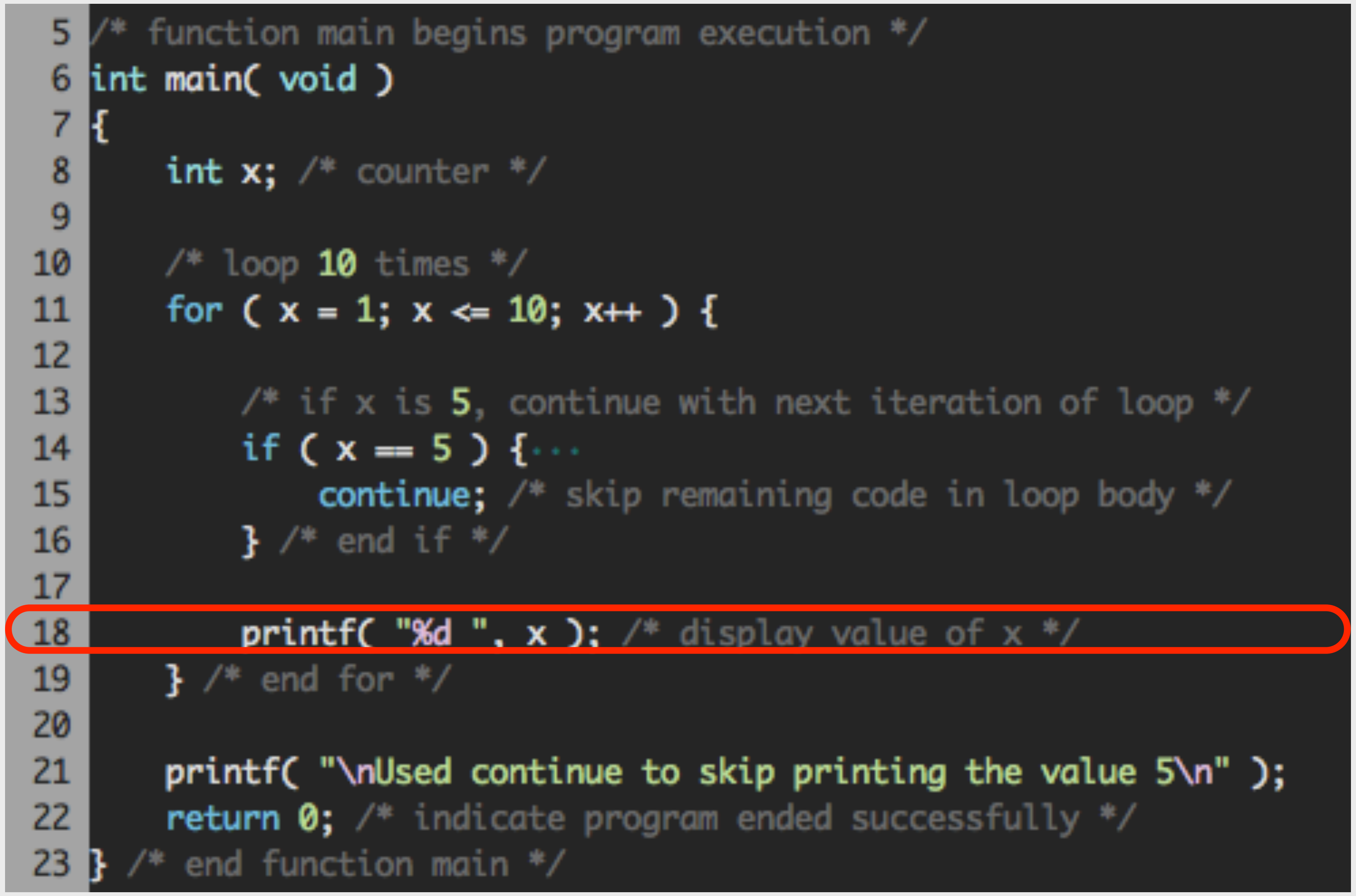
```c
/* function main begins program execution */
int main( void )
{
    int x; /* counter */

    /* loop 10 times */
    for ( x = 1; x <= 10; x++ ) {

        /* if x is 5, continue with next iteration of loop */
        if ( x == 5 ) {···
            continue; /* skip remaining code in loop body */
        } /* end if */

        printf( "%d ", x ); /* display value of x */
    } /* end for */

    printf( "\nUsed continue to skip printing the value 5\n" );
    return 0; /* indicate program ended successfully */
} /* end function main */
```

Example: fig04_12.c



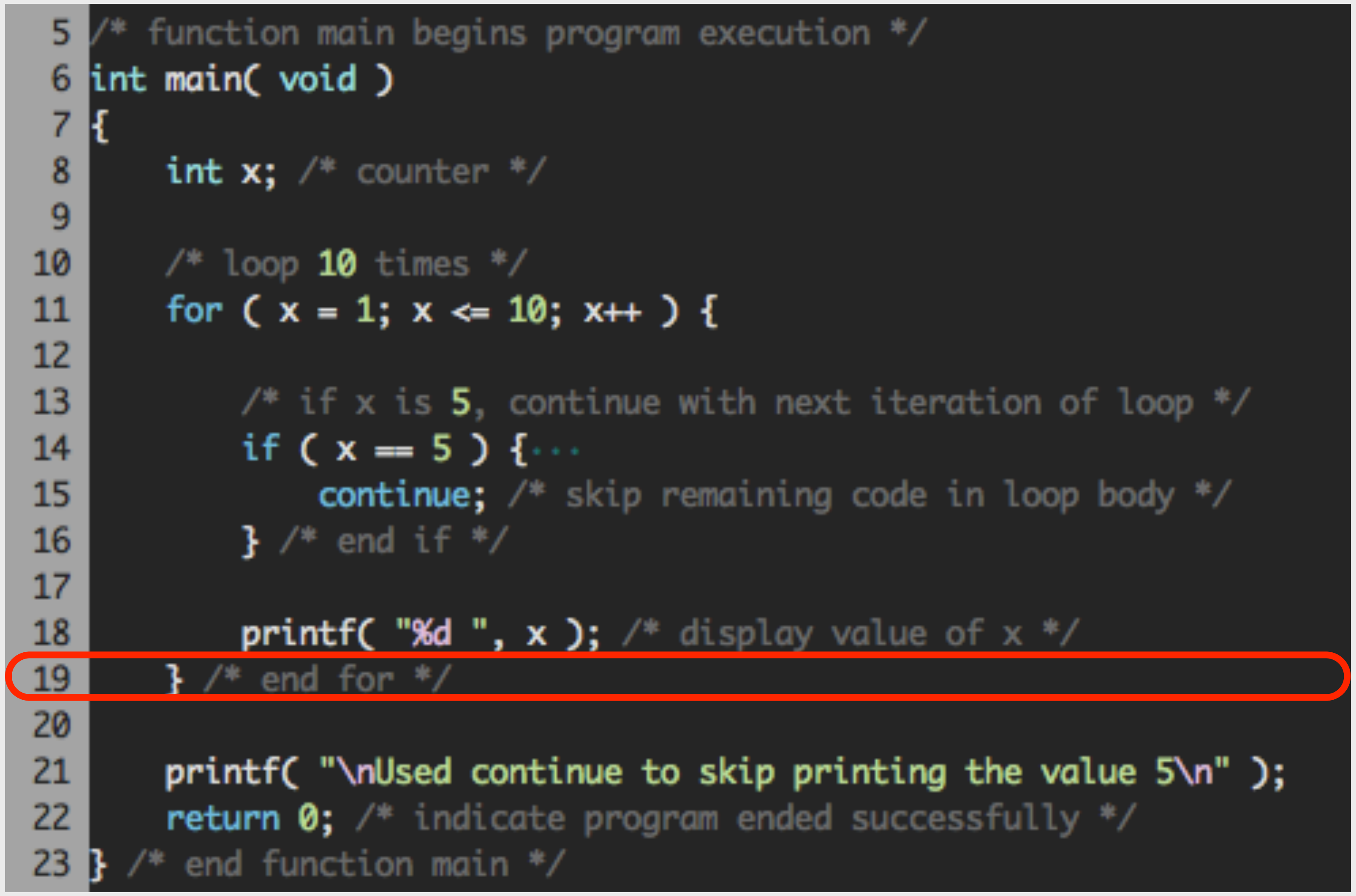
```
/* function main begins program execution */
int main( void )
{
    int x; /* counter */

    /* loop 10 times */
    for ( x = 1; x <= 10; x++ ) {

        /* if x is 5, continue with next iteration of loop */
        if ( x == 5 ) {···
            continue; /* skip remaining code in loop body */
        } /* end if */

        printf( "%d ", x ); /* display value of x */
    } /* end for */

    printf( "\nUsed continue to skip printing the value 5\n" );
    return 0; /* indicate program ended successfully */
} /* end function main */
```

Example: fig04_12.c



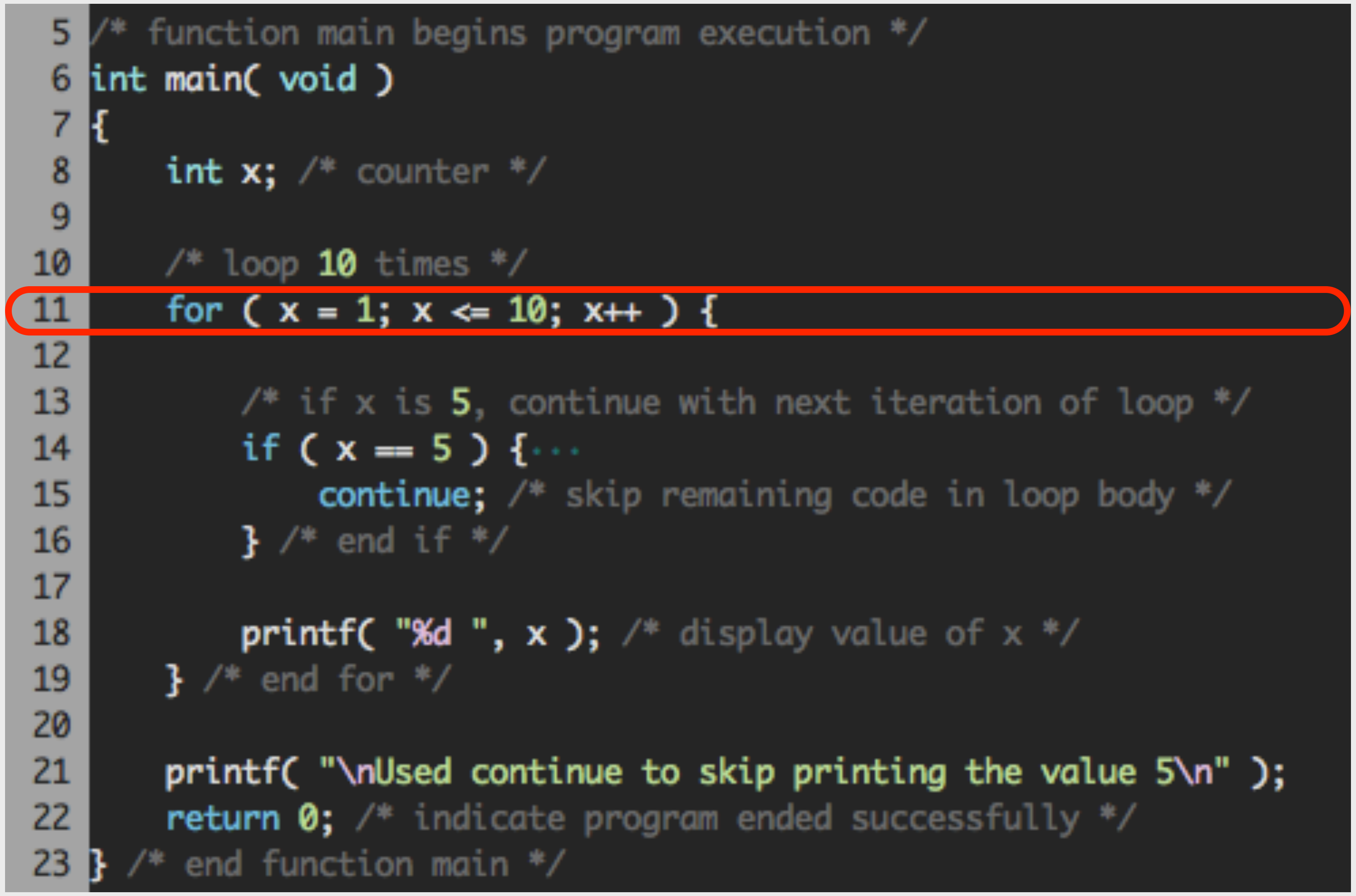
```c
/* function main begins program execution */
int main( void )
{
    int x; /* counter */

    /* loop 10 times */
    for ( x = 1; x <= 10; x++ ) {

        /* if x is 5, continue with next iteration of loop */
        if ( x == 5 ) {
            continue; /* skip remaining code in loop body */
        } /* end if */

        printf( "%d ", x ); /* display value of x */
    } /* end for */

    printf( "\nUsed continue to skip printing the value 5\n" );
    return 0; /* indicate program ended successfully */
} /* end function main */
```

Example: fig04_12.c



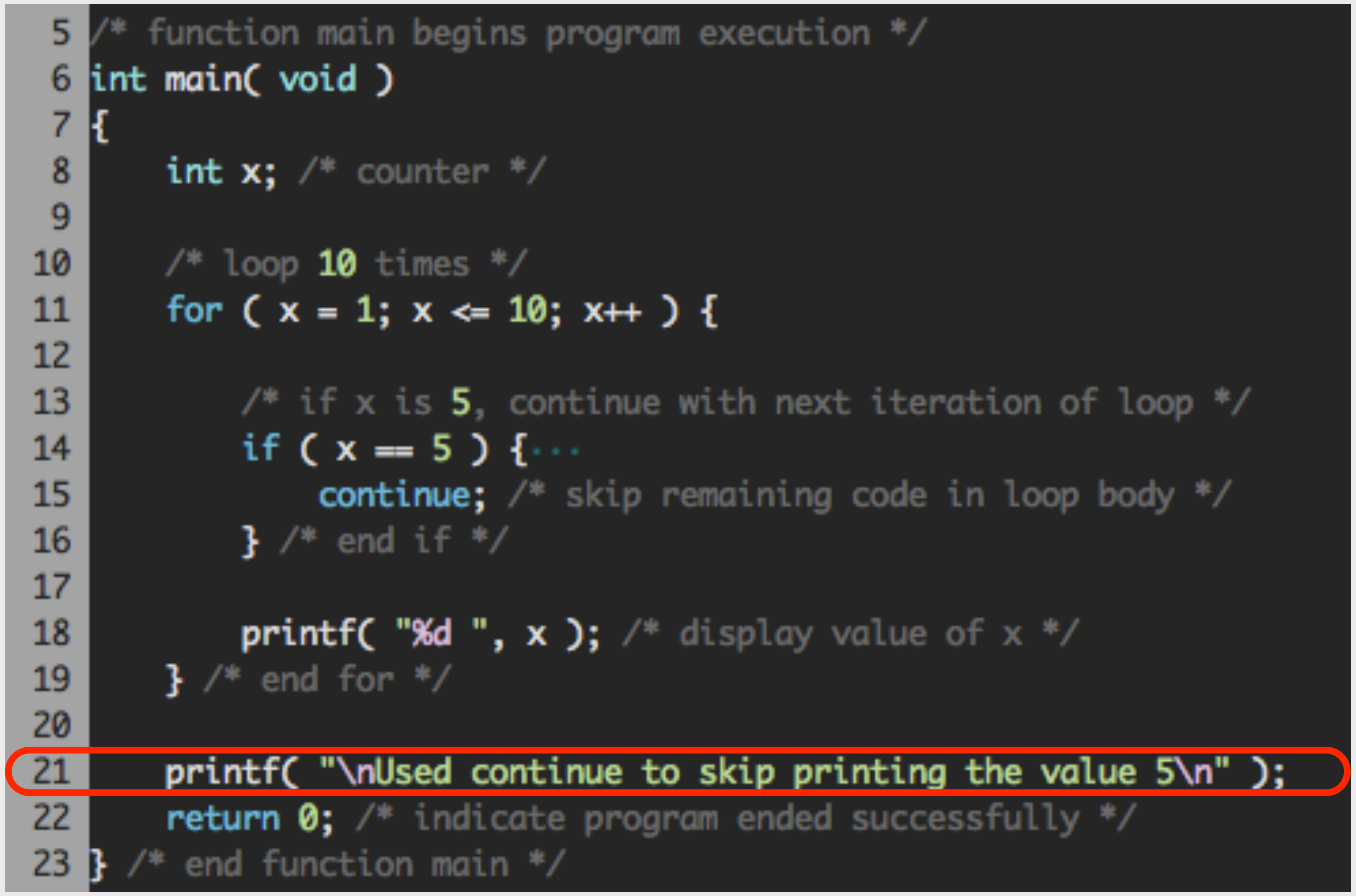
```
/* function main begins program execution */
int main( void )
{
    int x; /* counter */

    /* loop 10 times */
    for ( x = 1; x <= 10; x++ ) {

        /* if x is 5, continue with next iteration of loop */
        if ( x == 5 ) {
            continue; /* skip remaining code in loop body */
        } /* end if */

        printf( "%d ", x ); /* display value of x */
    } /* end for */

    printf( "\nUsed continue to skip printing the value 5\n" );
    return 0; /* indicate program ended successfully */
} /* end function main */
```

Example: fig04_12.c



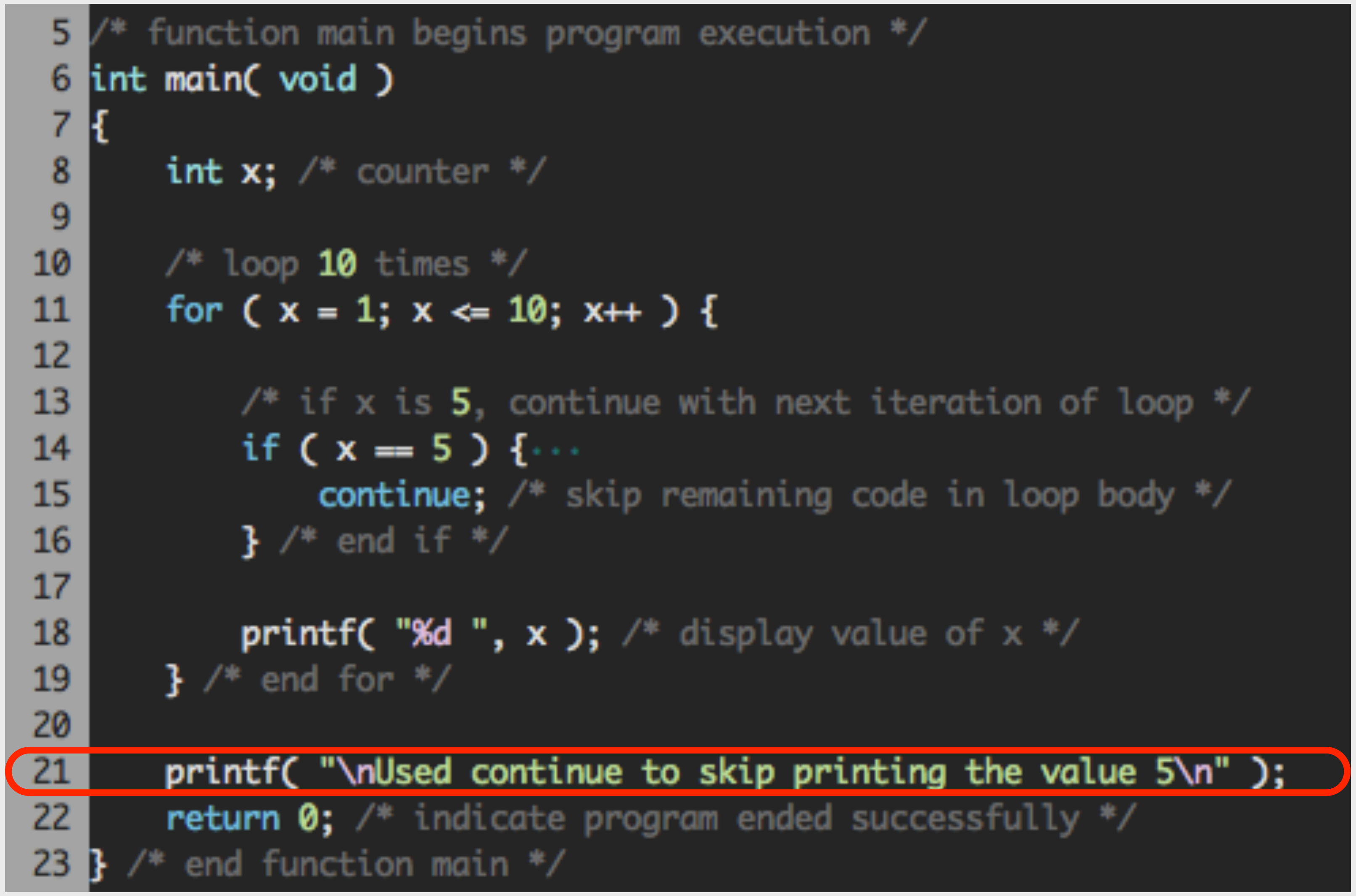
```c
/* function main begins program execution */
int main( void )
{
    int x; /* counter */

    /* loop 10 times */
    for ( x = 1; x <= 10; x++ ) {

        /* if x is 5, continue with next iteration of loop */
        if ( x == 5 ) {
            continue; /* skip remaining code in loop body */
        } /* end if */

        printf( "%d ", x ); /* display value of x */
    } /* end for */

    printf( "\nUsed continue to skip printing the value 5\n" );
    return 0; /* indicate program ended successfully */
} /* end function main */
```

Example: fig04_12.c

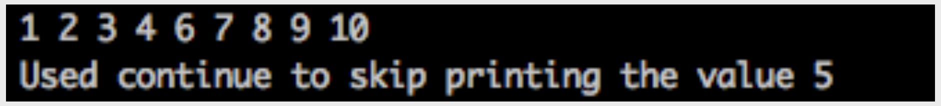


```c
/* function main begins program execution */
int main( void )
{
    int x; /* counter */

    /* loop 10 times */
    for ( x = 1; x <= 10; x++ ) {

        /* if x is 5, continue with next iteration of loop */
        if ( x == 5 ) {
            continue; /* skip remaining code in loop body */
        } /* end if */

        printf( "%d ", x ); /* display value of x */
    } /* end for */

    printf( "\nUsed continue to skip printing the value 5\n" );
    return 0; /* indicate program ended successfully */
} /* end function main */
```

Example: fig04_12.c



```c
/* function main begins program execution */
int main( void )
{
    int x; /* counter */

    /* loop 10 times */
    for ( x = 1; x <= 10; x++ ) {

        /* if x is 5, continue with next iteration of loop */
        if ( x == 5 ) {
            continue; /* skip remaining code in loop body */
        } /* end if */

        printf( "%d ", x ); /* display value of x */
    } /* end for */

    printf( "\nUsed continue to skip printing the value 5\n" );
    return 0; /* indicate program ended successfully */
} /* end function main */
```



```
1 2 3 4 6 7 8 9 10
Used continue to skip printing the value 5
```

Logical Operators

- The logical operators are **`&&`** (logical AND), **`||`** (logical OR) and **`!`** (logical NOT also called logical negation).
- In this case, we can use the logical operator **`&&`** as follows:

```
if ( gender == 1 && age >= 65 )
     ++seniorFemales;
```

Logical Operators (Cont.)

- In this case, we use the `||` operator as in the following program segment

```
if(semesterAverage >= 90 || finalExam >= 90)
        printf( "Student grade is A\n" );
```

- For example, the preceding statement may also be written as follows:

```
if(grade != sentinelValue)
    printf("The next grade is %f\n", grade);
```

Confusing Equality (==) and Assignment (=) Operators

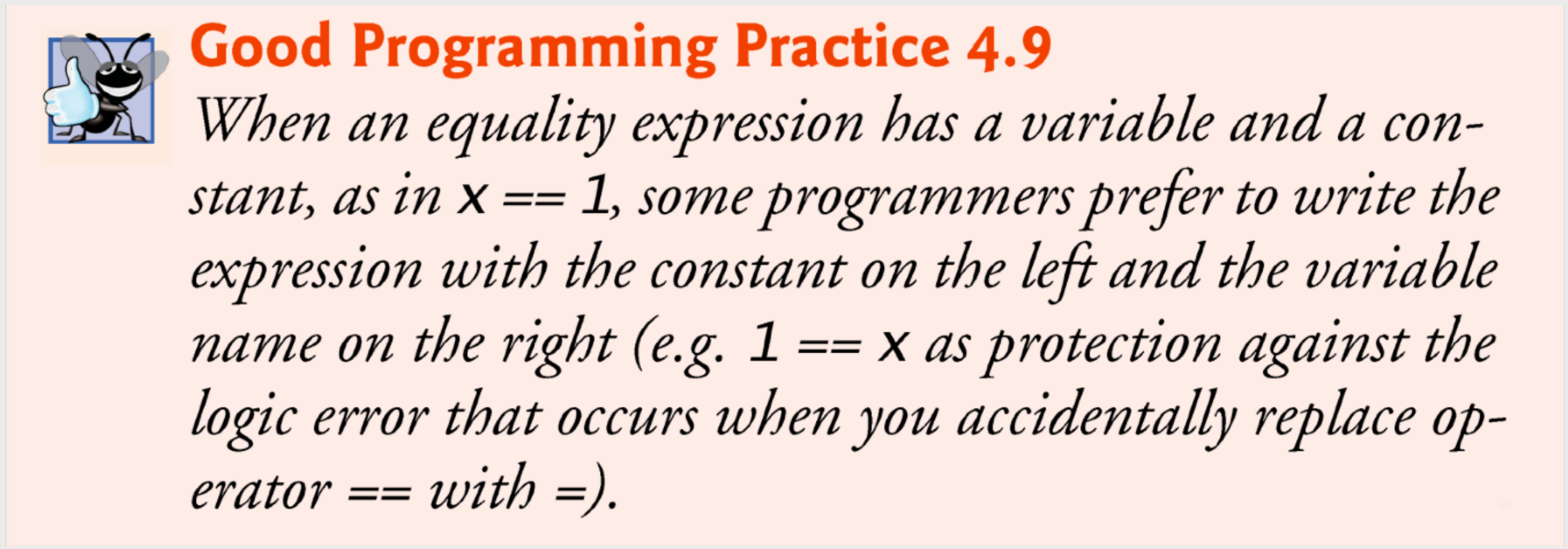
- For example, suppose we intend to write

```
if ( payCode == 4 )
     printf( "You get a bonus!" );
```

  but we accidentally write

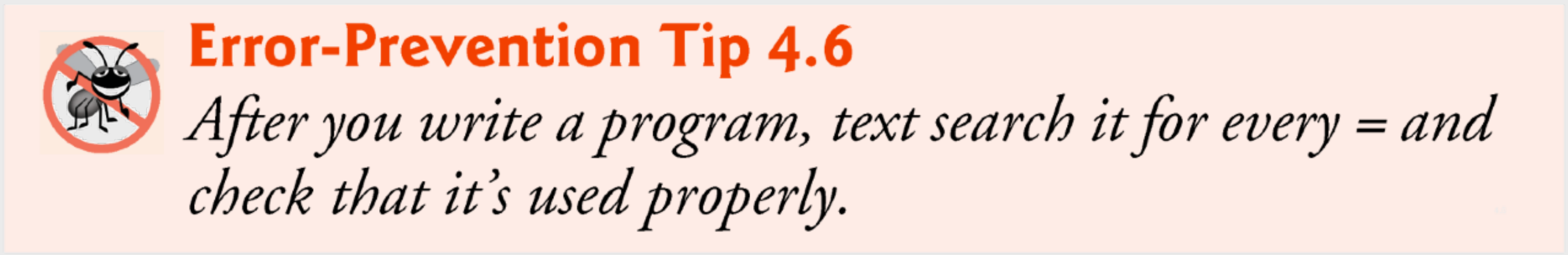
```
if ( payCode = 4 )
     printf( "You get a bonus!" );
```

Confusing Equality (==) and Assignment (=) Operators (Cont.)



Good Programming Practice 4.9

When an equality expression has a variable and a constant, as in `x == 1`*, some programmers prefer to write the expression with the constant on the left and the variable name on the right (e.g.* `1 == x` *as protection against the logic error that occurs when you accidentally replace operator == with =).*



Error-Prevention Tip 4.6

After you write a program, text search it for every = and check that it's used properly.

Structured Programming Summary

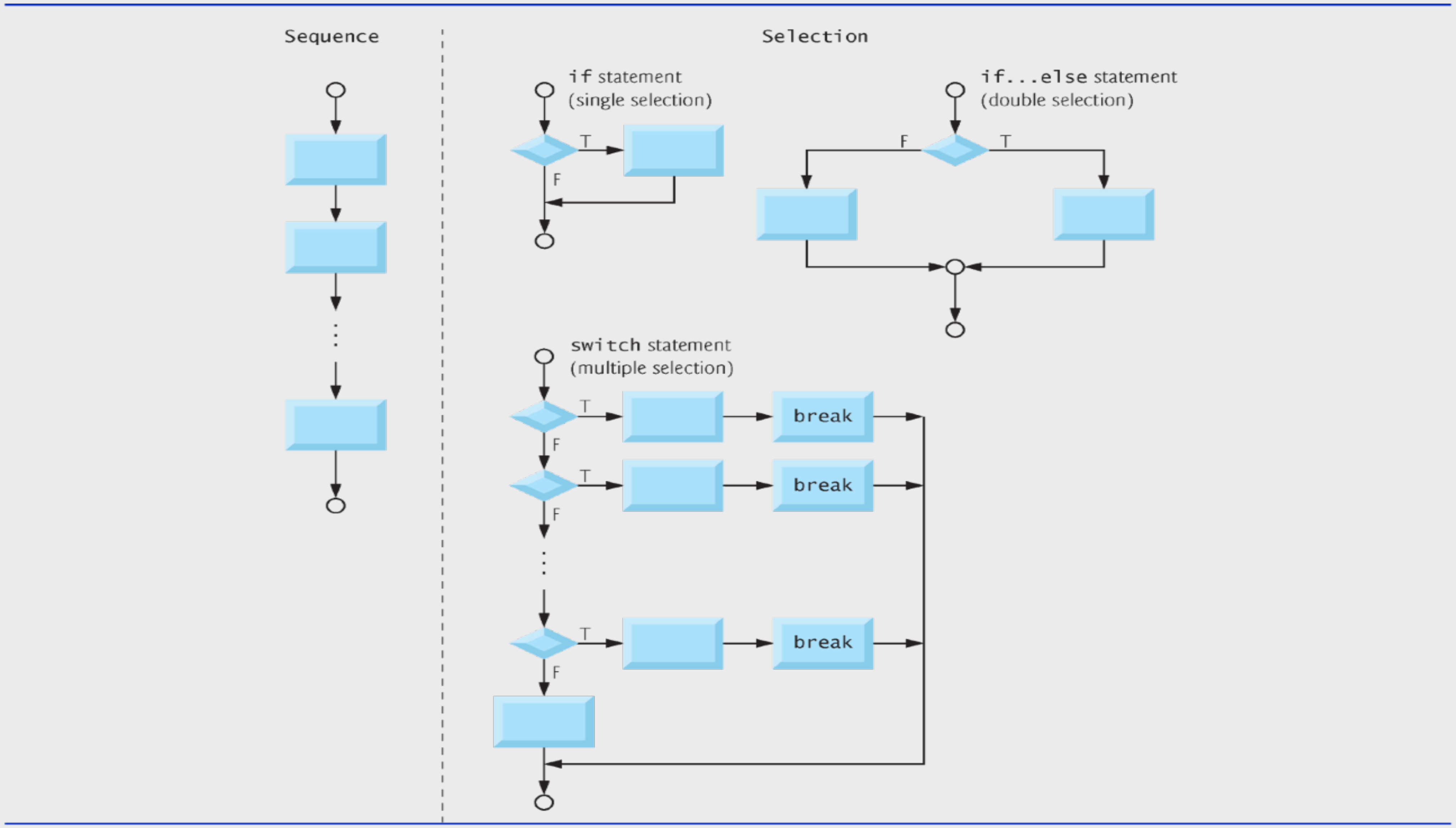


Fig. 4.17 | C's single-entry/single-exit sequence, selection and repetition statements. (Part 1 of 2.)

Structured Programming Summary (Cont.)

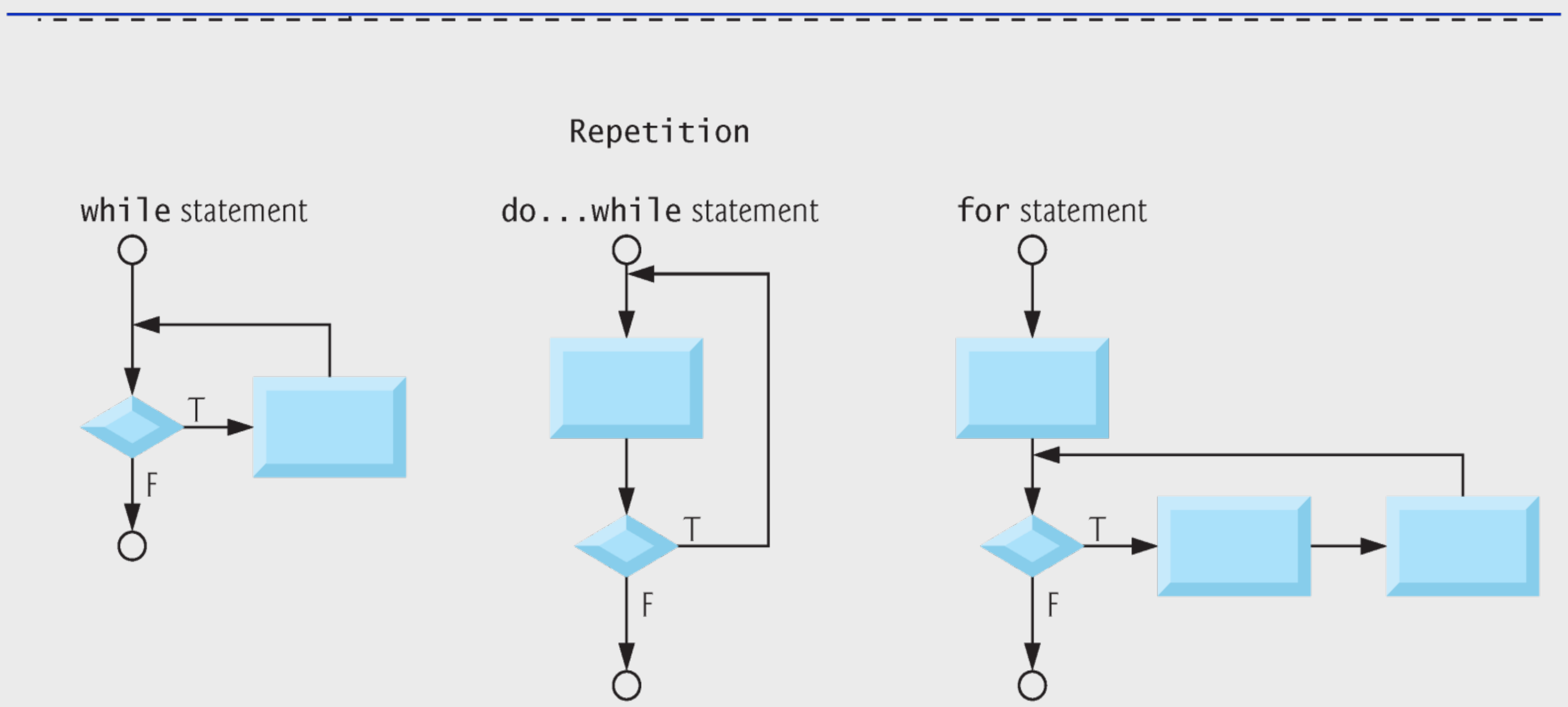


Fig. 4.17 | C's single-entry/single-exit sequence, selection and repetition statements. (Part 2 of 2.)

Structured Programming Summary (Cont.)

Rules for Forming Structured Programs

1)Begin with the “simplest flowchart” (Fig. 4.19).

2)Any rectangle (action) can be replaced by two rectangles (actions) in sequence.

3)Any rectangle (action) can be rep’laced by any control statement (sequence, `if`, `if...else`, `switch`, `while`, `do...while` or `for`).

4)Rules 2 and 3 may be applied as often as you like and in any order.

Fig. 4.18 | Rules for forming structured programs.

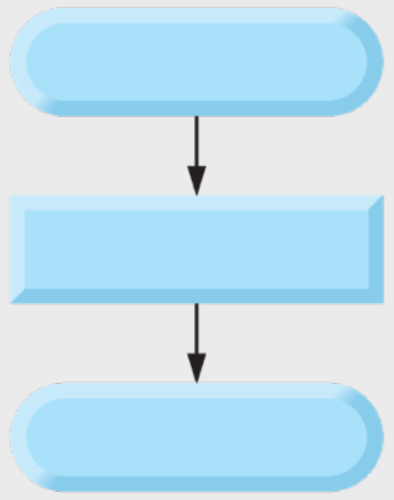


Fig. 4.19 | Simplest flowchart.

Structured Programming Summary (Cont.)

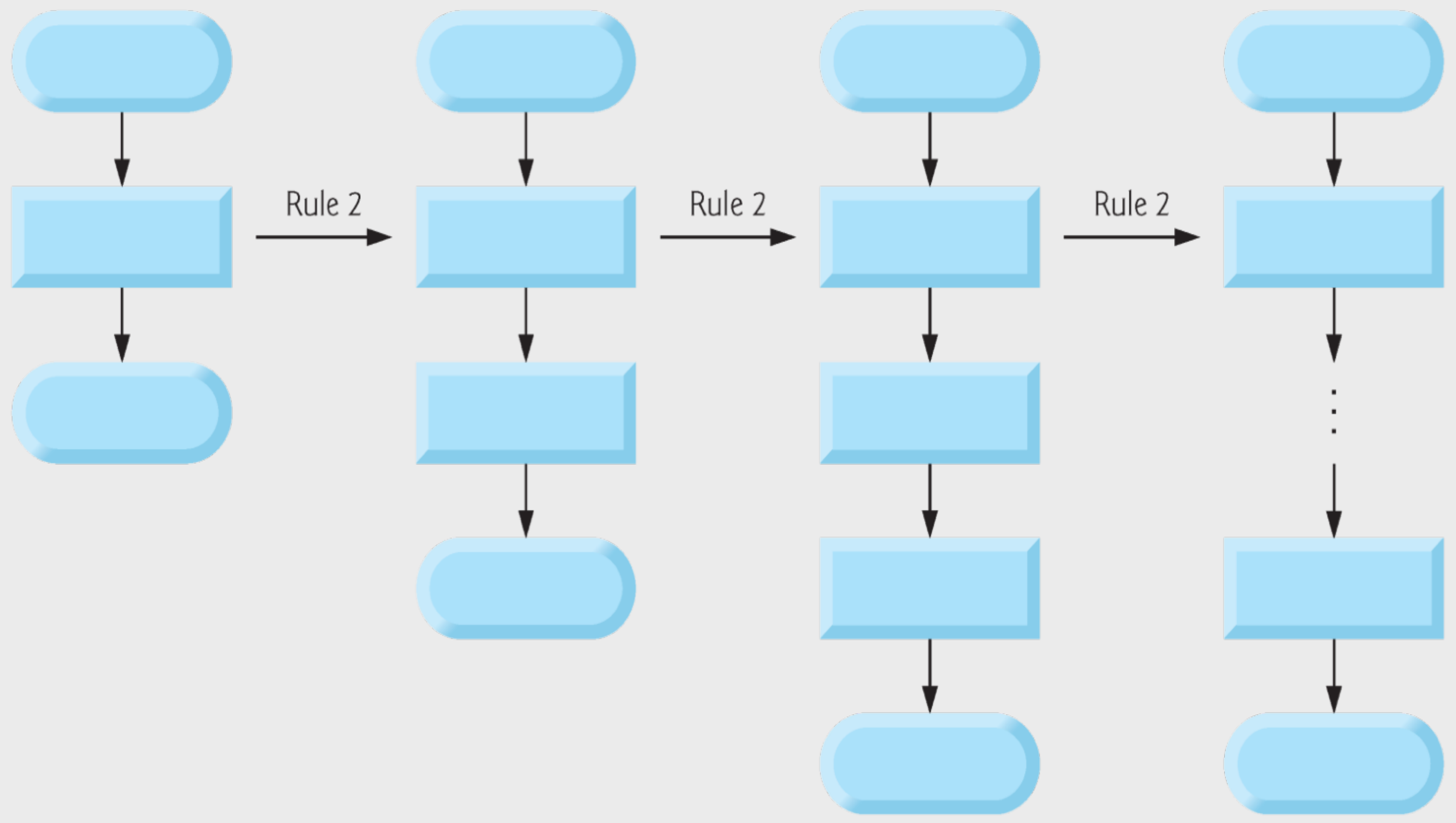


Fig. 4.20 | Repeatedly applying Rule 2 of Fig. 4.18 to the simplest flowchart.

Structured Programming Summary (Cont.)

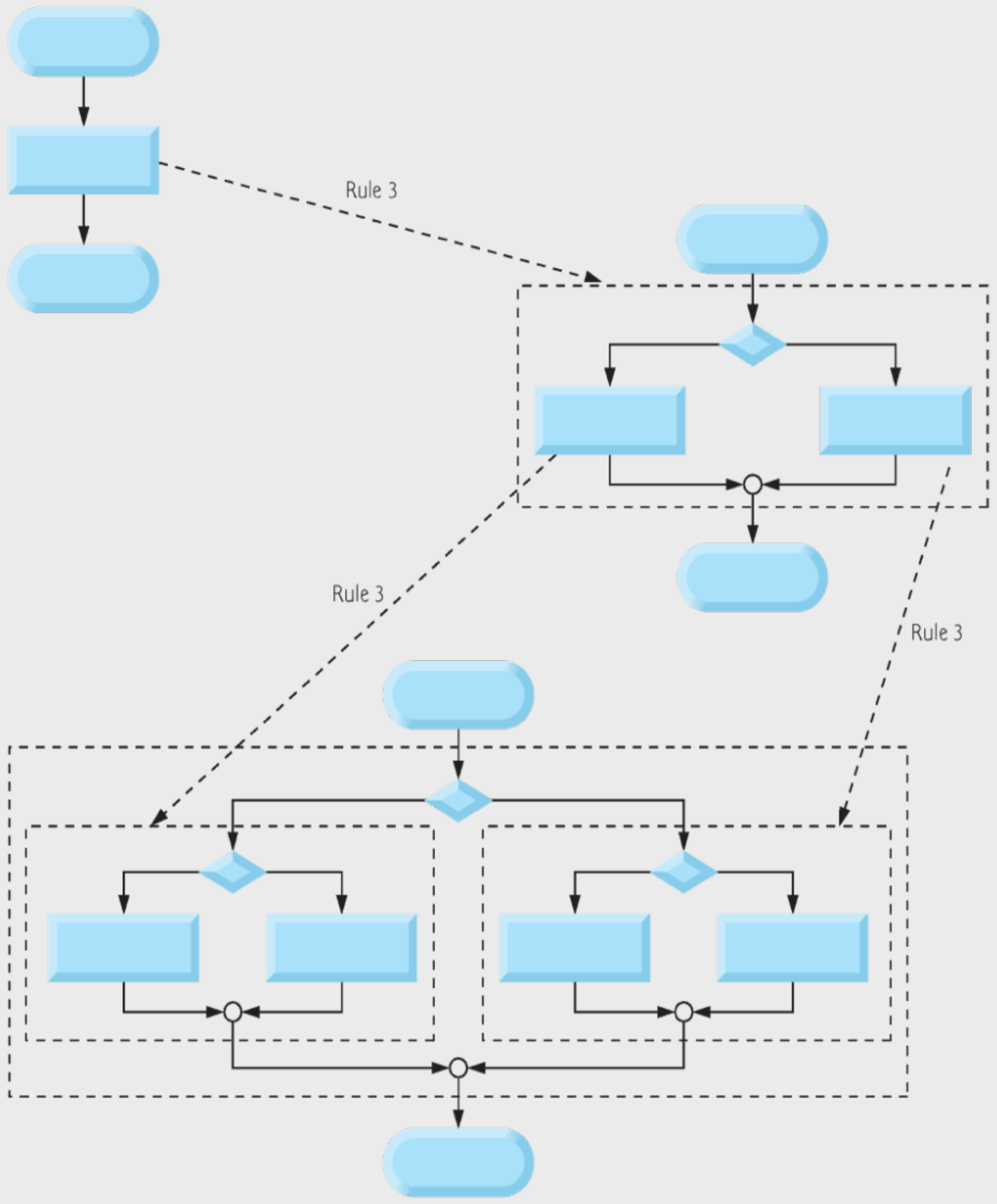


Fig. 4.21 | Applying Rule 3 of Fig. 4.18 to the simplest flowchart.

Structured Programming Summary (Cont.)

- Sequence is straightforward.
- Selection is implemented in one of three ways:
  - **`if`** statement (single selection)
  - **`if…else`** statement (double selection)
  - **`switch`** statement (multiple selection)

Structured Programming Summary (Cont.)

- Repetition is implemented in one of three ways:
  - **`while`** statement
  - **`do…while`** statement
  - **`for`** statement

Unix Tips

- Terminal Multiplexer (tmux)
  - A software to multiplex several virtual consoles, allowing a user to access multiple separate terminal sessions inside a single terminal window or remote terminal session
  - tmux 基本教學
  - tmux Tutorial