The `while` Statement

- A general repetition statement
- Format:

```
while (expression)
        statement;
```

- Function:
  - `expression` is evaluated the same as an if-else statement
  - Statement following expression executed repeatedly as long as expression has non-zero value

The `while` Statement (cont'd.)

- Steps in execution of `while` statement:
  1. Test the expression
  2. If the expression has a non-zero (true) value
     - a. Execute the statement following the parentheses
     - b. Go back to step 1

     else
     - a. Exit the `while` statement

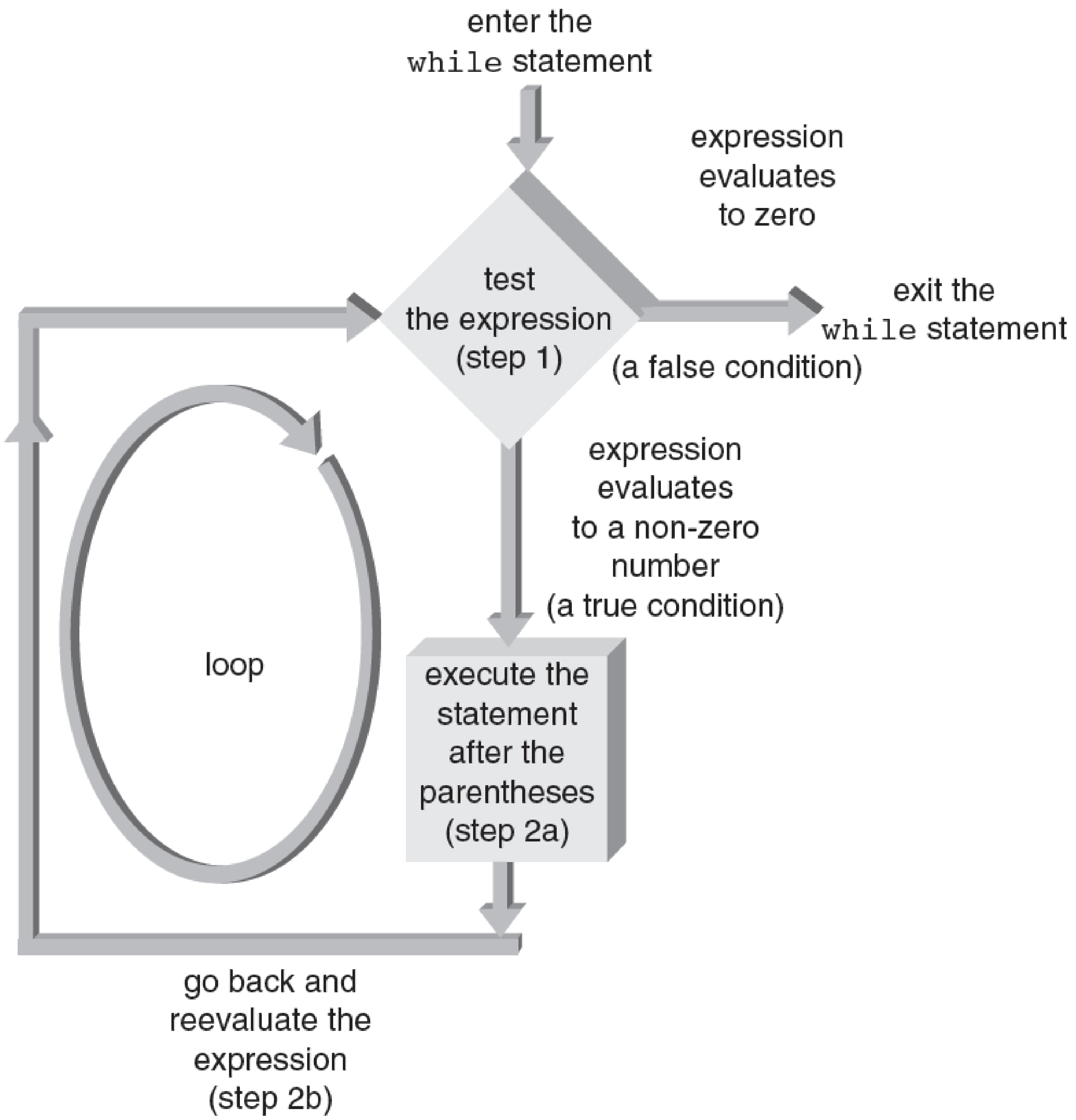
enter the while statement
expression evaluates to zero
test the expression (step 1)
exit the while statement
(a false condition)
loop
expression evaluates to a non-zero number (a true condition)
execute the statement after the parentheses (step 2a)
go back and reevaluate the expression (step 2b)

Figure 5.1 Anatomy of a `while` loop

The `while` Statement (cont'd.)

Program 5.1

```
#include <iostream>
using namespace std;

int main()
{
  int count;

  count = 1;              // initialize count
  while (count <= 10)
  {
    cout << count << " ";
    count++;              // increment count
  }

  return 0;
}
```

This is the output for Program 5.1:

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

The `while` Statement (cont'd.)

- Infinite loop: one that never ends
- Fixed-count loop: tested expression is a counter that checks for a fixed number of repetitions
- Variation: counter is incremented by a value other than 1
- Example:
  - Celsius-to-Fahrenheit temperature conversion
  - Display Fahrenheit and Celsius temperatures, from 5-50 degrees C, in 5 degree increments

The `while` Statement (cont'd.)

```
celsius = 5;                // starting Celsius
value
while (celsius <= 50)
{
  fahren = (9.0/5.0) * celsius + 32.0;
  cout << setw(4) << celsius
       << setw(13) << fahren << endl;
  celsius = celsius + 5;
}
```

Interactive `while` Loops

- Combining interactive data entry with the repetition of a `while` statement
  - Produces very powerful and adaptable programs
- Example (Program 5.5): `while` statement accepts and displays four user-entered numbers
  - Numbers accepted and displayed one at a time

Program 5.5

```
#include <iostream>
using namespace std;

int main()
{
  const int MAXNUMS = 4;
  int count;
  double num;

  cout << "\nThis program will ask you to enter "
       << MAXNUMS << " numbers.\n";
  count = 1;

  while (count <= MAXNUMS)
  {
    cout << "\nEnter a number: ";
    cin  >> num;
    cout << "The number entered is " << num;
    count++;
  }
  cout << endl;

  return 0;
}
```

Interactive `while` Loops (cont'd.)

- Sample run of Program 5.5:

```
This program will ask you to enter 4
numbers.
Enter a number: 26.2
The number entered is 26.2
Enter a number: 5
The number entered is 5
Enter a number: 103.456
The number entered is 103.456
Enter a number: 1267.89
The number entered is 1267.89
```

Interactive `while` Loops (cont'd.)

- Example (Program 5.6): adding a single number to a total
  - A number is entered by the user
  - Accumulating statement adds the number to total

    ```
    total = total + num;
    ```

  - A `while` statement repeats the process

Interactive `while` Loops (cont'd.)

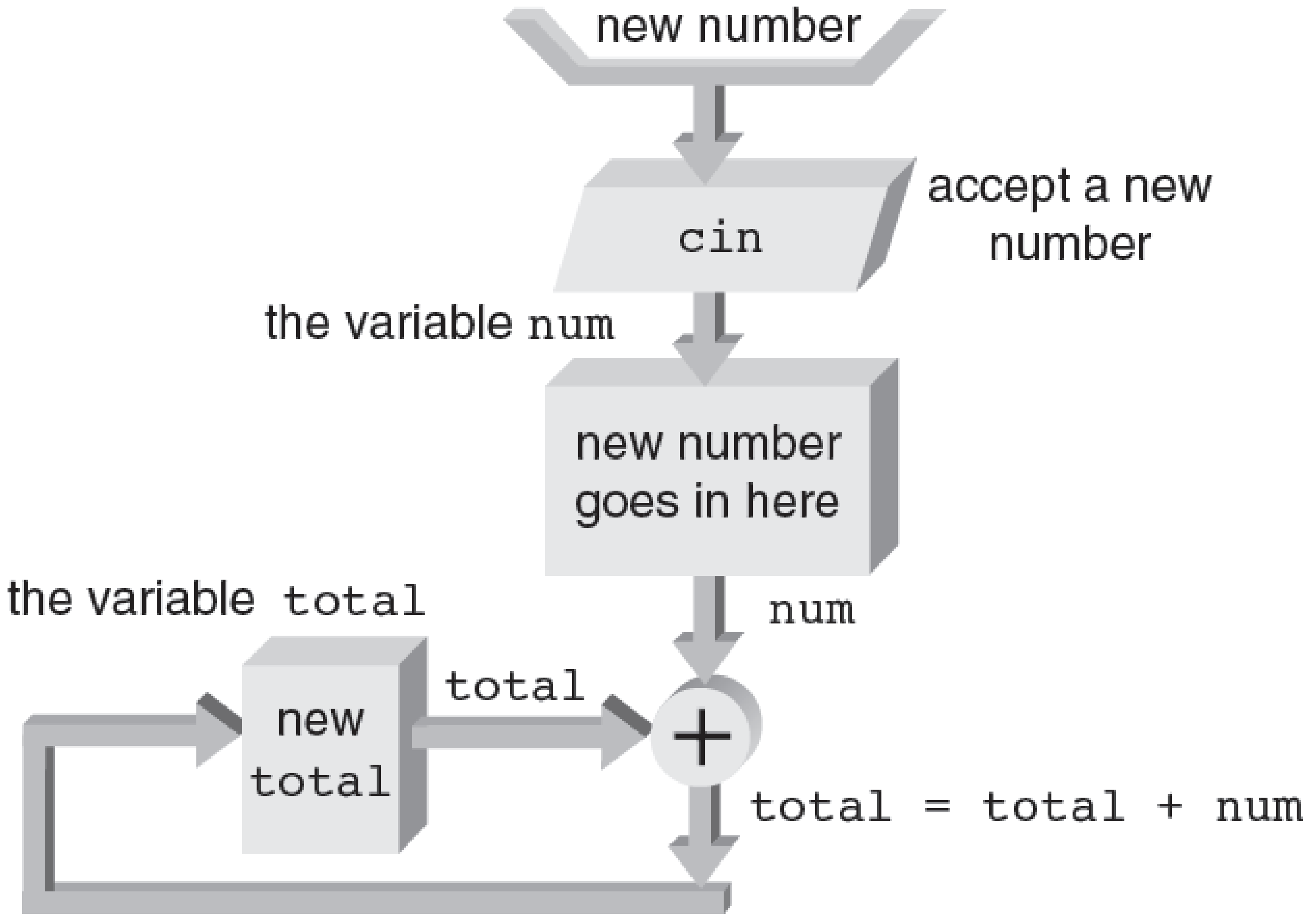

Figure 5.3 Accepting and adding a number to a total

Program 5.6

```
#include <iostream>
using namespace std;

int main()
{
  const int MAXNUMS = 4;
  int count;
  double num, total;

  cout << "\nThis program will ask you to enter "
       << MAXNUMS << " numbers.\n";
  count = 1;
  total = 0;

  while (count <= MAXNUMS)
  {
    cout << "\nEnter a number: ";
    cin  >> num;
    total = total + num;
    cout << "The total is now " << total;
    count++;
  }

  cout   << "\n\nThe final total is " << total << endl;

  return 0;
}
```

Interactive `while` Loops (cont'd.)

- Sample run of Program 5.6:

```
This program will ask you to enter 4 numbers.
Enter a number: 26.2
The total is now 26.2
Enter a number: 5
The total is now 31.2
Enter a number: 103.456
The total is now 134.656
Enter a number: 1267.89
The total is now 1402.546

The final total is 1402.546
```

Sentinels

- Sentinels
  - Data values used to signal the start or end of data series
- Values must be selected so as not to conflict with legitimate data values

`break` and `continue` Statements

- `break`: forces immediate exit from structures
  - Use in switch statements:
    - The desired case has been detected and processed
  - Use in `while`, `for`, and `do-while` statements:
    - An unusual condition is detected
- Format:

```
break;
```

`break` and `continue` Statements (cont'd.)

- `continue`: causes the next iteration of the loop to begin immediately
  - Execution transferred to the top of the loop
  - Applies only to `while`, `do-while`, and `for` statements
- Format:

```
    continue;
```

The Null Statement

- Used where a statement is syntactically required but no action is called for
  - A do-nothing statement
  - Typically used with `while` or `for` statements