```
int number;
cout << "Enter a number: ";
cin >> number;

if (number > 0)
{
   cout << "Zero\n";

   if (number > 10)
   {
      cout << "Ten\n";

      if (number > 20)
      {
         cout << "Twenty\n";
      }
   }
}
```

4.6 The `if/else if` Statement

CONCEPT: The `if/else if` statement tests a series of conditions. It is often simpler to test a series of conditions with the `if/else if` statement than with a set of nested `if/else` statements.

VideoNote
The `if/else if` Statement

Even though Program 4-12 is a simple example, the logic of the nested decision structure is fairly complex. In C++, and many other languages, you can alternatively test a series of conditions using the `if/else if` statement. The `if/else if` statement makes certain types of nested decision logic simpler to write. Here is the general format of the `if/else if` statement:

```
if (expression_1)
{
   statement          }   If expression_1 is true these state-
   statement          }   ments are executed, and the rest of the
   etc.               }   structure is ignored.
}
else if (expression_2)
{
   statement          }   Otherwise, if expression_2 is true these
   statement          }   statements are executed, and the rest of the
   etc.               }   structure is ignored.
}
Insert as many else if clauses as necessary
else
{
   statement          }   These statements are executed
   statement          }   if none of the expressions above
   etc.               }   are true.
}
```

When the statement executes, *`expression_1`* is tested. If *`expression_1`* is true, the block of statements that immediately follows is executed, and the rest of the structure is ignored. If *`expression_1`* is false, however, the program jumps to the very next `else if` clause and tests *`expression_2`*. If it is true, the block of statements that immediately follows is executed, and then the rest of the structure is ignored. This process continues, from the top of the structure to the bottom, until one of the expressions is found to be true. If none of the expressions are true, the last `else` clause takes over, and the block of statements immediately following it is executed.

The last `else` clause, which does not have an `if` statement following it, is referred to as the *trailing* `else`. The trailing `else` is optional, but in most cases you will use it.

NOTE: The general format shows braces surrounding each block of conditionally executed statements. As with other forms of the `if` statement, the braces are required only when more than one statement is conditionally executed.

Program 4-13 shows an example of the `if/else` if statement. This program is a modification of Program 4-12, which appears in the previous *In the Spotlight* section.

Program 4-13

```
// This program uses an if/else if statement to assign a
// letter grade (A, B, C, D, or F) to a numeric test score.
#include <iostream>
using namespace std;

int main()
{
    // Constants for grade thresholds
    const int A_SCORE = 90,
              B_SCORE = 80,
              C_SCORE = 70,
              D_SCORE = 60;

    int testScore; // To hold a numeric test score

    // Get the numeric test score.
    cout << "Enter your numeric test score and I will\n"
         << "tell you the letter grade you earned: ";
    cin >> testScore;

    // Determine the letter grade.
    if (testScore >= A_SCORE)
       cout << "Your grade is A.\n";
    else if (testScore >= B_SCORE)
       cout << "Your grade is B.\n";
    else if (testScore >= C_SCORE)
       cout << "Your grade is C.\n";
    else if (testScore >= D_SCORE)
       cout << "Your grade is D.\n";
    else
       cout << "Your grade is F.\n";
```

(program continues)

Program 4-13 *(continued)*

```
32
33        return 0;
34    }
```

Program Output with Example Input Shown in Bold

```
Enter your numeric test score and I will
tell you the letter grade you earned: 78 [Enter]
Your grade is C.
```

Program Output with Different Example Input Shown in Bold

```
Enter your numeric test score and I will
tell you the letter grade you earned: 84 [Enter]
Your grade is B.
```

Let's analyze how the `if/else if` statement in lines 22 through 31 works. First, the expression `testScore >= A_SCORE` is tested in line 22:

```
→ if (testScore >= A_SCORE)
     cout << "Your grade is A.\n";
  else if (testScore >= B_SCORE)
     cout << "Your grade is B.\n";
  else if (testScore >= C_SCORE)
     cout << "Your grade is C.\n";
  else if (testScore >= D_SCORE)
     cout << "Your grade is D.\n";
  else
     cout << "Your grade is F.\n";
```

If `testScore` is greater than or equal to 90, the message `"Your grade is A.\n"` is displayed and the rest of the `if/else if` statement is skipped. If `testScore` is not greater than or equal to 90, the `else` clause in line 24 takes over and causes the next `if` statement to be executed:

```
  if (testScore >= A_SCORE)
      cout << "Your grade is A.\n";

→ else if (testScore >= B_SCORE)
     cout << "Your grade is B.\n";
  else if (testScore >= C_SCORE)
     cout << "Your grade is C.\n";
  else if (testScore >= D_SCORE)
     cout << "Your grade is D.\n";
  else
     cout << "Your grade is F.\n";
```

The first `if` statement handles all of the grades greater than or equal to 90, so when this `if` statement executes, `testScore` will have a value of 89 or less. If `testScore` is greater than or equal to 80, the message `"Your grade is B.\n"` is displayed and the rest of the `if/else if` statement is skipped. This chain of events continues until one of the expressions is found to be true, or the last `else` clause at the end of the statement is encountered.

Notice the alignment and indentation that is used with the `if/else if` statement: The starting `if` clause, the `else if` clauses, and the trailing `else` clause are all aligned, and the conditionally executed statements are indented.