26. The `break` statement causes an exit
    a. only from the innermost loop.
    b. only from the innermost switch.
    c. from all loops and switches.
    d. from the innermost loop or switch.
27. Executing the `continue` operator from within a loop causes control to go to ________.
28. The `goto` statement causes control to go to
    a. an operator.
    b. a label.
    c. a variable.
    d. a function.

## Exercises

Answers to the starred exercises can be found in Appendix G.

*1. Assume that you want to generate a table of multiples of any given number. Write a program that allows the user to enter the number and then generates the table, formatting it into 10 columns and 20 lines. Interaction with the program should look like this (only the first three lines are shown):

```
Enter a number: 7
     7   14   21   28   35   42   49   56   63   70
    77   84   91   98  105  112  119  126  133  140
   147  154  161  168  175  182  189  196  203  210
```

*2. Write a temperature-conversion program that gives the user the option of converting Fahrenheit to Celsius or Celsius to Fahrenheit. Then carry out the conversion. Use floating-point numbers. Interaction with the program might look like this:

```
Type 1 to convert Fahrenheit to Celsius,
     2 to convert Celsius to Fahrenheit: 1
Enter temperature in Fahrenheit: 70
In Celsius that's 21.111111
```

*3. Operators such as >>, which read input from the keyboard, must be able to convert a series of digits into a number. Write a program that does the same thing. It should allow the user to type up to six digits, and then display the resulting number as a type `long` integer. The digits should be read individually, as characters, using `getche()`. Constructing the number involves multiplying the existing value by 10 and then adding the new digit. (Hint: Subtract 48 or ‘0’ to go from ASCII to a numerical digit.)

Here's some sample interaction:

```
Enter a number: 123456
Number is: 123456
```

*4. Create the equivalent of a four-function calculator. The program should ask the user to enter a number, an operator, and another number. (Use floating point.) It should then carry out the specified arithmetical operation: adding, subtracting, multiplying, or dividing the two numbers. Use a `switch` statement to select the operation. Finally, display the result.

When it finishes the calculation, the program should ask whether the user wants to do another calculation. The response can be `'y'` or `'n'`. Some sample interaction with the program might look like this:

```
Enter first number, operator, second number: 10 / 3
Answer = 3.333333
Do another (y/n)? y
Enter first number, operator, second number: 12 + 100
Answer = 112
Do another (y/n)? n
```

5. Use `for` loops to construct a program that displays a pyramid of Xs on the screen. The pyramid should look like this

```
    X
   XXX
  XXXXX
 XXXXXXX
XXXXXXXXX
```

except that it should be 20 lines high, instead of the 5 lines shown here. One way to do this is to nest two inner loops, one to print spaces and one to print Xs, inside an outer loop that steps down the screen from line to line.

6. Modify the FACTOR program in this chapter so that it repeatedly asks for a number and calculates its factorial, until the user enters 0, at which point it terminates. You can enclose the relevant statements in FACTOR in a `while` loop or a `do` loop to achieve this effect.

7. Write a program that calculates how much money you'll end up with if you invest an amount of money at a fixed interest rate, compounded yearly. Have the user furnish the initial amount, the number of years, and the yearly interest rate in percent. Some interaction with the program might look like this:

```
Enter initial amount: 3000
Enter number of years: 10
Enter interest rate (percent per year): 5.5
At the end of 10 years, you will have 5124.43 dollars.
```

At the end of the first year you have 3000 + (3000 * 0.055), which is 3165. At the end of the second year you have 3165 + (3165 * 0.055), which is 3339.08. Do this as many times as there are years. A `for` loop makes the calculation easy.

8. Write a program that repeatedly asks the user to enter two money amounts expressed in old-style British currency: pounds, shillings, and pence. (See Exercises 10 and 12 in Chapter 2, "C++ Programming Basics.") The program should then add the two amounts and display the answer, again in pounds, shillings, and pence. Use a `do` loop that asks the user whether the program should be terminated. Typical interaction might be

   ```
   Enter first amount: £5.10.6
   Enter second amount: £3.2.6
   Total is £8.13.0
   Do you wish to continue (y/n)?
   ```

   To add the two amounts, you'll need to carry 1 shilling when the pence value is greater than 11, and carry 1 pound when there are more than 19 shillings.

9. Suppose you give a dinner party for six guests, but your table seats only four. In how many ways can four of the six guests arrange themselves at the table? Any of the six guests can sit in the first chair. Any of the remaining five can sit in the second chair. Any of the remaining four can sit in the third chair, and any of the remaining three can sit in the fourth chair. (The last two will have to stand.) So the number of possible arrangements of six guests in four chairs is 6*5*4*3, which is 360. Write a program that calculates the number of possible arrangements for any number of guests and any number of chairs. (Assume there will never be fewer guests than chairs.) Don't let this get too complicated. A simple `for` loop should do it.

10. Write another version of the program from Exercise 7 so that, instead of finding the final amount of your investment, you tell the program the final amount and it figures out how many years it will take, at a fixed rate of interest compounded yearly, to reach this amount. What sort of loop is appropriate for this problem? (Don't worry about fractional years; use an integer value for the year.)

11. Create a three-function calculator for old-style English currency, where money amounts are specified in pounds, shillings, and pence. (See Exercises 10 and 12 in Chapter 2.) The calculator should allow the user to add or subtract two money amounts, or to multiply a money amount by a floating-point number. (It doesn't make sense to multiply two money amounts; there is no such thing as square money. We'll ignore division. Use the general style of the ordinary four-function calculator in Exercise 4 in this chapter.)

12. Create a four-function calculator for fractions. (See Exercise 9 in Chapter 2, and Exercise 4 in this chapter.) Here are the formulas for the four arithmetic operations applied to fractions:

| | |
|---|---|
| Addition: | `a/b + c/d = (a*d + b*c) / (b*d)` |
| Subtraction: | `a/b - c/d = (a*d - b*c) / (b*d)` |
| Multiplication: | `a/b * c/d = (a*c) / (b*d)` |
| Division: | `a/b / c/d = (a*d) / (b*c)` |

The user should type the first fraction, an operator, and a second fraction. The program should then display the result and ask whether the user wants to continue.