EXERCISE 7-6 NESTED if STRUCTURES

1. Open *FINAL.CPP*.
2. Compile, link, and run the program. Run the program several times, testing different combinations of values.
3. When you have verified that the output is correct, close the source code file.

Extra for Experts

The nested if structure in Figure 7-12 is used to set the `exempt_from_final` *variable to true or false. Any type of statements could appear in the nested if statement. However, because the code in this case simply sets the value of the* `exempt_from_final` *variable, the entire nested if structure can be replaced with the statement below.*

```
exempt_from_final = (((my_average >= 90) && (my_days_absent <= 3)) ||
                     ((my_average >= 80) && (my_days_absent <= 1)));
```

Using logical operators, the statement above combines the expressions used in the nested if structure. The result is the value desired for `exempt_from_final`*. To prove it, replace the nested if structure in FINAL.CPP with the statement above and run the program again.*

Do not be fooled, however, into thinking that statements like the one above make if structures unnecessary. Ordinarily, a selection structure cannot be replaced with a sequence structure and achieve the same result.

Figure 7-13 shows a simple program that includes a nested if structure. The program asks the user to input the amount of money he or she wishes to deposit in order to open a new checking account. Based on the value provided by the user, the program recommends a type of account.

```
#include <iostream.h>

main()
{
  float amount_to_deposit;

  cout << "How much do you want to deposit to open the account? ";
  cin  >> amount_to_deposit;

  if(amount_to_deposit < 1000.00 )
   {
    if(amount_to_deposit < 100.00 )
     { cout << "You should consider the EconoCheck account.\n"; }
    else
     { cout << "You should consider the FreeCheck account.\n"; }
   }
  else
   { cout << "You should consider an interest-bearing account.\n"; }
  return 0;
}
```

FIGURE 7-13
The nested if structure is used to make a recommendation.

EXERCISE 7-7 MORE NESTED if

1. Open *DEPOSIT.CPP*. The program in Figure 7-13 appears.
2. Compile, link, and run the program. Run the program several times using values which are less than $100, between $100 and $1000, and greater than $1000.
3. Close the source code file.

THE switch STRUCTURE

Note

A **menu** *is a set of options presented to the user of a program.*

You have studied one-way (if) and two-way (if-else) selection structures. C++ has another method of handling multiple options known as the ***switch structure***. The switch structure has many uses, but may be most often used when working with menus. Figure 7-14 is a code segment that displays a menu of choices and asks the user to enter a number that corresponds to one of the choices. Then a case statement is used to handle each of the options.

Let's analyze the switch structure in Figure 7-14. It begins with the keyword *switch*, followed by the control expression (the variable **`shipping_method`**) to be compared in the structure. Within the braces of the structure are a series of *case* keywords. Each one provides the code that is to be executed in the event that **`shipping_method`** matches the value that follows case. The *default* keyword tells the compiler that if nothing else matches, execute the statements that follow.

The *break* keyword, which appears at the end of each case segment, causes the flow of logic to jump to the first executable statement after the switch structure.

```
cout << "How do you want the order shipped?\n";
cout << "1 - Ground\n";
cout << "2 - 2-day air\n";
cout << "3 - Overnight air\n";
cout << "Enter the number of the shipping method you want: ";
cin >> shipping_method;

switch(shipping_method)
 {
   case 1:
    shipping_cost = 5.00;
    break;
   case 2:
    shipping_cost = 7.50;
    break;
   case 3:
    shipping_cost = 10.00;
    break;
   default:
    shipping_cost = 0.00;
    break;
 }
```

FIGURE 7-14
The switch structure takes action based on the user's input.