

Chapter 22 – Do-While Loops

The third loop control structure is *do-while loop*, which is very similar to the *while-loop*. The *do-while* has the following form:

```
do {
    statement(s);
} while (Boolean-expression);
```

where *do* is the beginning of the loop body
statement(s) are the statements in the body of the loop
while is the end of the body of the loop
Boolean-expression turns false to halt the loop

Since the Boolean expression is positioned last in the loop, the body of a *do-while* loop will always be performed before the Boolean expression is evaluated. *This means that the body of the loop will be performed at least once.* (This is unlike the *while-loop* where the body of the loop may not be performed at all.)

This feature makes *do-while* loops unwieldy and complex in many looping situations, but perfect for other uses. For example, *do-while* loops are ideal for controlling user input from a menu prompt. The following example program will demand that the user input a “correct” or valid value. In fact, the program will not continue until the user does input a valid character.

```
#include <iostream>
#include <conio.h>
using namespace std;

void main( ) {
    char beeps, a_beep = 7;
    cout << "Press the digit to the left of your choice:" << endl
         << " 1 - 1 beep" << endl
         << " 2 - 2 beeps" << endl
         << " 3 - 3 beeps" << endl
         << " 4 - 4 beeps" << endl;
    do {
        beeps = getch( );
    } while (beeps < '1' || beeps > '4');

    switch(beeps){
    case '4':
        cout << a_beep;
    case '3':
        cout << a_beep;
```

```

        case '2':
            cout << a_beep;
        case '1':
            cout << a_beep;
    }
}

```

When run, the *do-while* will reject all input that is not the characters '1', '2', '3', or '4'. Once the user selects one of these valid inputs, the program will proceed and produce the correct number of beeps. (Note that the *switch* statement does not need a *default* option.)

Programming Assignment 22.1

Add a *do-while* loop to the interface function(s) of the class from programming assignment 21.4 so that any incorrect selections from the menu(s) of choices of CD products to buy should be ignored. Use a *getch* function as part of this addition to the program.

Programming Assignment 22.2

This program will use the class *Car* from chapter 12.

Write a program that accepts two characters the user until the user decides to halt. Each character represents a vehicle of class *Car*. Possible vehicle symbols are 'A', 'B', 'C', 'D' and 'E'. Each symbol is associated with a counter.

Each time two characters are read in, they are to become the symbols of two objects of class *Car*. The two *Car* objects then race each other by being moved 10 times each. Based on their locations, a winner or a tie is determined. If a *Car* object wins the race, its counter (based on the current symbol of the *Car* object) is incremented by +2. If there is a tie, the counters of both *Car* objects are incremented by +1. The finish of each race should be displayed on the monitor screen.

The results of each race are to be output to the monitor screen. In the cases where there is a winner, this will include the name of the winner and the distance the winner has gone. In the case where there is a tie, the output will consist of the name of both contestants and the distance that they have traveled.

When the end-of-file is reached, the program should output to the monitor screen the symbol and score of each vehicle. For example:

```

A – 59
B – 37
C – 63
D – 27
E – 84

```