In many situations, program control can be accomplished by using either a `while` or a `for` statement. Which one gets used is often a matter of taste. One major advantage of a `for` loop is that control and indexing can both be kept right at the top. When loops are nested, this can facilitate the reading of the code. The program in the next section illustrates this.

4.10 An Example: Boolean Variables

Boolean algebra plays a major role in the design of computer circuits. In this algebra all variables have only the values zero or one. Transistors and memory technologies implement zero-one value schemes with currents, voltages, and magnetic orientations. Frequently, the circuit designer has a function in mind and needs to check whether, for all possible zero-one inputs, the output has the desired behavior.

We will use `int` variables `b1`, `b2`, ... , `b5` to represent five boolean variables. They will be allowed to take on only the values 0 and 1. A boolean function of these variables is one that returns only 0 or 1. A typical example of a boolean function is the majority function; it returns 1 if a majority of the variables have value 1, and 0 otherwise. We want to create a table of values for the functions

```
b1 || b3 || b5     and     b1 && b2 || b4 && b5
```

and the majority function. Recall that logical expressions always have the `int` value 0 or 1.

In file bool_vals.c

```
/* Print a table of values for some boolean functions. */
#include <stdio.h>

int main(void)
{
   int   b1, b2, b3, b4, b5;              /* boolean variables */
   int   cnt = 0;

   printf("\n%5s%5s%5s%5s%5s%5s%7s%7s%11s\n\n",    /* headings */
      "Cnt", "b1", "b2", "b3", "b4", "b5",
      "fct1", "fct2", "majority");
   for (b1 = 0; b1 <= 1; ++b1)
      for (b2 = 0; b2 <= 1; ++b2)
         for (b3 = 0; b3 <= 1; ++b3)
            for (b4 = 0; b4 <= 1; ++b4)
               for (b5 = 0; b5 <= 1; ++b5)
                  printf("%5d%5d%5d%5d%5d%5d%6d%7d%9d\n",
                     ++cnt, b1, b2, b3, b4, b5,
                     b1 || b3 || b5, b1 && b2 || b4 && b5,
                     b1 + b2 + b3 + b4 + b5 >= 3);
   putchar('\n');
   return 0;
}
```

The program prints a table of values for all possible inputs and corresponding outputs. It illustrates a typical use of nested `for` loops. Here is some output of the program:

```
  Cnt   b1   b2   b3   b4   b5   fct1   fct2   majority
    1    0    0    0    0    0      0      0          0
    2    0    0    0    0    1      1      0          0
    3    0    0    0    1    0      0      0          0
    .....
```