

BA272 Conditionals and Loops - Software as Story?

So you now understand some of the building blocks of programs, e.g., source code, classes, variables, objects, compilers, and executables. Next we will focus more on how we connect those blocks to accomplish things.

In addition to the “straight line” code we have seen so far, we can make choices depending on conditions and we can repeat sets of instructions (p. 45).

First consider `if` syntax. The basic idea is simple: only do certain steps in certain conditions. In a simple form, here is the syntax:

```
if ( conditional statement ) { do_something; }
```

A conditional statement is either true or false. When we deal with Boolean variables we can just say: `if (boolMyBooleanVariable)`, but most conditional statements employ relational operators that compare two values. The most important ones are listed on page 47. Especially note equals which is `==`. Remember someone decided to use an equals sign `=` as a gozzinta instruction so we have to use two equals signs together to ask a question. `==` is a lot like the equal sign in mathematical equations. Generally, if `a==b` then `b==a` etc. We can accomplish even more using And, Or, Not and Parentheses.

Make up some questions comparing several numbers in different conditions and practice. Write them on a paper and pass them up so we can all try them.

Can you write a program which says “Bigger than a bread box” when either the area of a box is greater than 3 cubic feet or its weight is greater than 5 pounds? Declare the needed variables, populate them with test values, create the code and see if it works.

Many lines of code can be lumped together between the braces `{ }` after an `if` statement. This will cause them all to be executed if the conditional evaluates to a value of true. There is a short cut (no braces) if only one statement is to be executed – but this can create readability issues and maintenance problems so I generally frown on the practice.

You can also use `else` as shown here:

```
val = Console.ReadLine();
if (val == "A")
    { Console.WriteLine("That was an A"); }
else
    { Console.WriteLine("That was NOT an A"); }
```

Pay attention to Rob’s comments about constants on page 49. Why might it be better to use a constant defined at the top of a program, e.g., `const double PI = 3.14159;` as compared with putting 3.14159 in several place in the program? Or, why not just use a regular variable – how does it being a constant help? *Hint: take note of the answer to these questions, you could see them on a quiz/exam.*

Another important control we have for program flow is use of a loop. Consider:

```
string val;
do
    {
        val = Console.ReadLine();
```

```

        if (val == "A")
            { Console.WriteLine("That was an A"); }
        else
            { Console.WriteLine("That was NOT an A"); }
    } while (!val.Equals("z"));

```

We will find lots of reasons to perform looping (iteration) in programs.

To begin with, you need to be able to create, compare, and contrast several kinds of loops: Do While, While, and For.

The main difference between Do While and While is that the test is performed after the code is executed in Do While, whereas in While the test is done before the code is executed. Thus the code in Do While is always executed once.

Instead of basing repetition on a conditional statement as in Do While and While, For loops are executed some particular number of times. For loops can be nested. So, for example, if you were counting the days in 3 weeks you could do the following:

```

int iCountDays = 0;
for (int iWeekNum = 1; iWeekNum <=3; iWeekNum++) {
    for (int iDayOfWeekNum = 1; iDayOfWeekNum <=7; iDayOfWeekNum++) {
        iCountDays++;
        Console.Write(iCountDays);
    }
}

```

Producing: 123456789101112131415161718192021

With a bit more effort you can make it show the day of the week. This example combines `else` and `if`.

```

int iCountDays = 0;
for (int iWeekNum = 1; iWeekNum <= 3; iWeekNum++) {
    for (int iDayOfWeekNum = 1; iDayOfWeekNum <= 7; iDayOfWeekNum++) {
        iCountDays++;
        if (iDayOfWeekNum == 1) {
            Console.Write(" Monday:");
        } else if (iDayOfWeekNum == 2) {
            Console.Write(" Tuesday:");
        } else if (iDayOfWeekNum == 3) {
            Console.Write(" Wednesday:");
        } else if (iDayOfWeekNum == 4) {
            Console.Write(" Thursday:");
        } else if (iDayOfWeekNum == 5) {
            Console.Write(" Friday:");
        } else if (iDayOfWeekNum == 6) {
            Console.Write(" Saturday:");
        } else if (iDayOfWeekNum == 7) {
            Console.Write(" Sunday:");
        }
        Console.Write(iCountDays);
    }
}

```

That code produces:

```

Monday:1 Tuesday:2 Wednesday:3 Thursday:4 Friday:5 Saturday:6 Sunday:7 Monday:8
Tuesday:9 Wednesday:10 Thursday:11 Friday:12 Saturday:13 Sunday:14 Monday:15 Tu
esday:16 Wednesday:17 Thursday:18 Friday:19 Saturday:20 Sunday:21

```

Can you make a Do While loop with a While loop nested inside that is equivalent to this one?

```

for (int iWeekNum = 1; iWeekNum <=3; iWeekNum++) {
    for (int iDayOfWeekNum = 1; iDayOfWeekNum <=7; iDayOfWeekNum++) {
        iCountDays++;
        Console.WriteLine(iCountDays);
    }
}

```

Really: you need to be able to do that yourself! But here is one answer:

```

int iCountDays = 0;
int iWeekNum = 1;
do {
    int iDayOfWeekNum = 1;
    while (iDayOfWeekNum <= 7) {
        iCountDays++;
        Console.WriteLine(iCountDays);
        iDayOfWeekNum++;
    }
    iWeekNum++;
} while (iWeekNum <= 3);

```

The point is that anything you can do in a `for` loop can be done in a `do` loop. Which of those two code blocks is better? Why?

Check out `break` and `continue` in the text. Try using these to stop after Tuesday or skip weekends in the looping code above - see pages 53 and 54.

Try this exercise:

- 1) Make a loop that accepts only A, B, or C (can you make it change an a into an A?). Be able to do it with both Do While and While.
- 2) In front of that loop in your code, make the console display some options (perhaps like this):
Please Choose from the following list of options
A - Compute Circumference
B - Compute distance in 10 rotations
C - Compute # of rotations in 100 yards
- 3) Create variables for PI and diameter – hard code their values. Would you use constants?
- 4) Make your program display the result of the chosen computation.
- 5) Now make another loop “wrapped” around your menu code. When the user chooses one of the letters, clear the screen (`Console.Clear()`), show the computation, then go back and display the menu again.
- 6) Add an option “Z” quit. If they choose this one don’t do anything, but get out of the loop.

Instructive variations:

- Use `break` – don’t use `break`.
- Use the placeholder method to round the numbers different ways.
- Write code using some or (|) parameters in addition to and (&) to test the letter values.
- Create a boolean variable called `isAMenuOption` and set it to true or false using code. Then use that value to control a loop.

There are lots of ways to do this.... Here is one.

```
const double dWHEELDIAMTER_IN_INCHES = 32;
double PI2 = Math.Round(Math.PI, 2);
string sLetter = "";
while (sLetter != "Z") { // Support repeated visits to the menu

    // Display a menu
    Console.WriteLine(" Please Choose from the following list of options ");
    Console.WriteLine(" A - Compute Circumference ");
    Console.WriteLine(" B - Compute distance in 10 rotations ");
    Console.WriteLine(" C - Compute # of rotations in 100 yards ");
    Console.WriteLine(" Z - Quit");

    do { sLetter = Console.ReadLine().ToUpper(); // Loop till you recognize a letter
    } while (sLetter != "A" && sLetter != "B" && sLetter != "C" && sLetter != "Z");

    if (sLetter == "Z") { break; } // {} needed? No Helpful? - maybe yes?

    Console.Clear(); // Carefully placed clearing of the screen
    Console.WriteLine(" "); // Just to make it pretty

    double Circum = 2 * PI2 * (dWHEELDIAMTER_IN_INCHES / 2);
    if (sLetter == "A")
        {Console.WriteLine(" Circumference is {0,7:####.00} ft", (Circum / 12));}
    if (sLetter == "B")
    { Console.WriteLine(" 10 Rotations would go {0,7:####.00} ft", (Circum / 12) * 10); }
    if (sLetter == "C")
    { Console.WriteLine(" 100 yards takes {0,7:####.00} rotations ", (3600f / Circum)); }

    Console.WriteLine(" "); // Just to make it pretty
}
```

Later this term you will see how arrays and the switch command can be used with looping and condition testing in a variety of simple but powerful applications.