Code Walkthroughs

Beginning programmers often write code that they don't understand. As you learn to program, you will be tempted to find code in your textbook, or in examples given in class, that seem to accomplish what you need done in your program. If you simply copy code that you don't understand, you will find it very difficult to fix the code if your program does not work correctly.

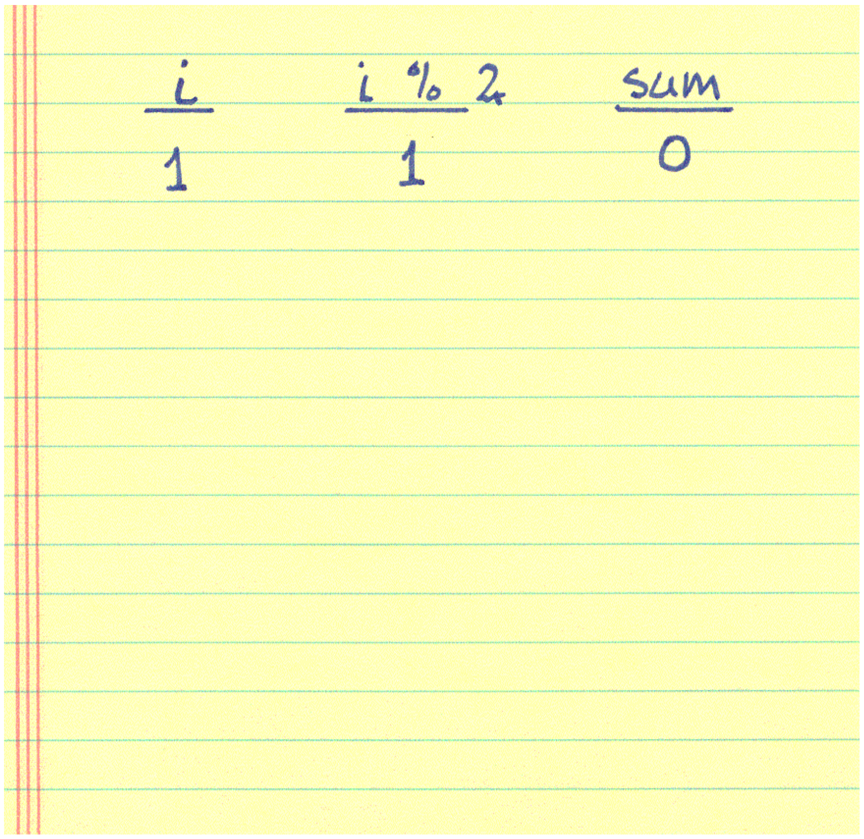
Before you ever compile your code, a useful technique for testing your understanding of the code is to do a **code walkthrough**. If you truly understand how your code works, you will find it much easier to spot errors when your program doesn't work like you expected it to. No doubt, you will find code walkthroughs tedious, but the understanding that comes from doing a code walkthrough is well worth the effort. To do a walkthrough, take out a blank piece of paper. Across the top of the sheet of paper, write down the names of the variables involved in the section of code that you are walking through. Now, trace through the code one line at a time, pretending in your mind that you are the computer that is executing the code. As you note changes in the values of variables, write them down on the paper.

An Example

Suppose that you have written some code to add together the even numbers between 1 and 10, inclusive. While not too complex, this example will serve to illustrate a code walkthrough. Your code might look like the following:

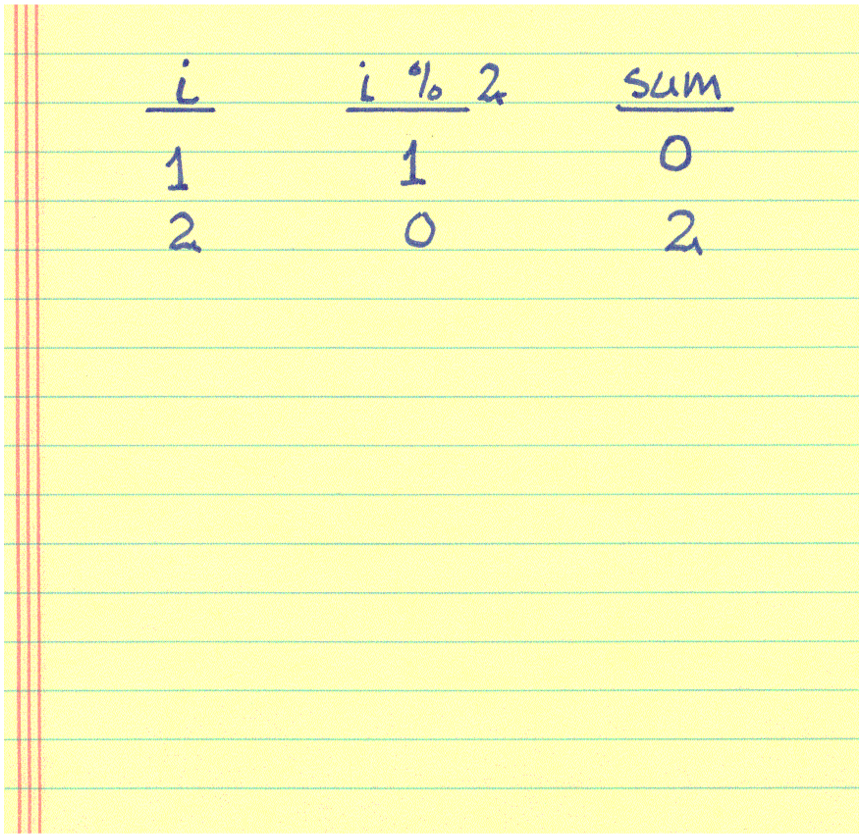
```
using System;

class Program
{
   static void Main ( )
   {
       // sum the even integers from 1 to 10
       int sum = 0;

        for (int i = 1; i < 10; i++)
       {
           if ( (i % 2) == 0)
               sum += i;
       }

       Console.WriteLine("The sum is {0}", sum);
       Console.ReadLine( );
   }
 }
```

To begin the walkthrough of this code, write down the names of the key variables on a sheet of paper. In this case, we probably want to keep track of what happens to the variables `sum` and `i`. However, since the computation depends on the value of `i` being even, it will also be useful to keep track of `i % 2`. To begin with, the value of `sum` is equal to zero. Now start through the loop. The first time we go through the loop, `i` will be equal to one. The value of `1 % 2` is also one. The logic in our program says that if `i % 2` is zero, then we should add the value of `i` to `sum`. Otherwise, we do nothing. So, at this point the value of `sum` is still zero. Our sheet of paper should look something like



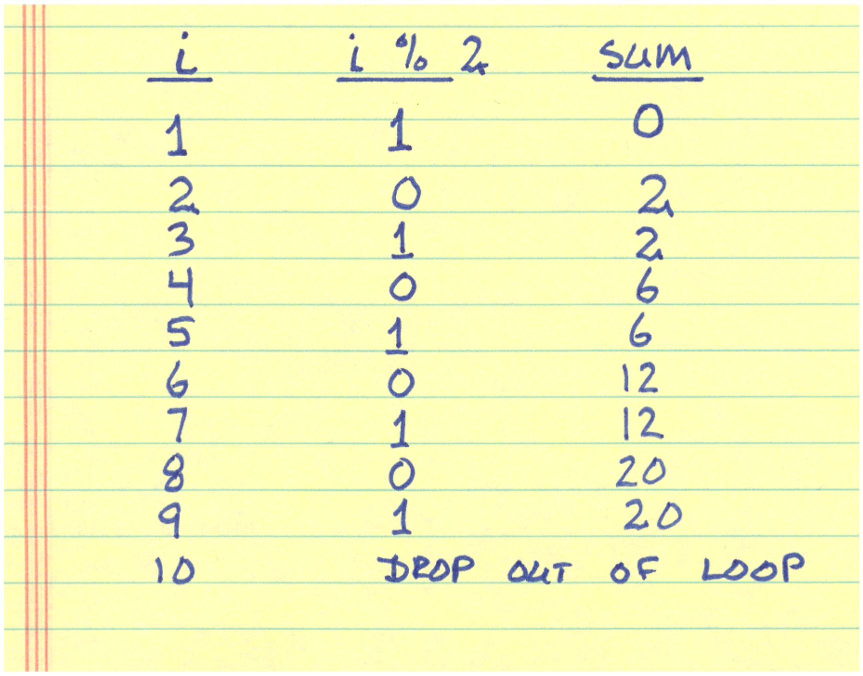
i	i % 2	sum
1	1	0

Increment the value of `i`. The value of `i` is now 2. Since 2 < 10, we will execute the body of the loop another time. With `i` equal to two, we have `2 % 2` equal to zero, so we will execute the statement `sum += 1;`. The value of `sum` is now equal to two. This is shown in the image below.



i	i % 2	sum
1	1	0
2	0	2

Continue in this fashion until the value of `i` is incremented to 10. Now, since `i` is no longer less than 10, we drop out of the loop. Our walkthrough sheet will look something like



i	i % 2	sum
1	1	0
2	0	2
3	1	2
4	0	6
5	1	6
6	0	12
7	1	12
8	0	20
9	1	20
10	DROP OUT OF LOOP	

If you have added up the even values between 1 and 10, you will notice that the value shown on our walkthrough is not correct. The correct answer is 30, ten more than the result we got in our walkthrough. It should be obvious from the walkthrough that we should have executed the body of the loop one more time, so that the value of ten was added to `sum`. We can correct this by changing line 9 to read

```
        for (int i = 1; i <= 10; i++)
```

Using a Walkthrough

After designing your code, type in the code for your program, or the part of the program that you want to test. Oftentimes this will be a single method. Code walkthroughs are extremely useful when testing a single method. Now walkthrough your code as described in the paragraphs above. Make any corrections to errors that you notice as you do your walkthrough.

Now, build and execute your program. See if it produces the correct results. If it does not, use the debugger to set watches on the variables you used in your walkthrough and step through your code one line at a time. The results should match what you wrote down in your walkthrough. If at some step the results do not match what you did in the walkthrough, you know that you need to examine this part of the code again. The error is usually obvious, **if you understand how the code is supposed to work**. This understanding comes from the code walkthrough. If what you see in the debugger and what you did in your walkthrough still do not match, you may not have understood some important programming principle. Watching what happens in the debugger is oftentimes very helpful in clarifying programming principles that you do not fully understand.