



```
balance *= interestRate;
```

Writing same code 10 times seems a bit wasteful, and what if you change the duration from 10 to 40 years?

You have to manually copy that code 40 times. So same code you can replace with following code..

```
class Program
```

```
{  
    static void Main(string[] args)  
    {  
        double balance, interestRate, targetBalance;  
        Console.WriteLine("What is your current balance?");  
        balance = Convert.ToDouble(Console.ReadLine());  
        Console.WriteLine("What is your current annual interest rate (in %)?");  
        interestRate = 1 + Convert.ToDouble(Console.ReadLine()) / 100.0;  
        Console.WriteLine("What balance would you like to have?");  
        targetBalance = Convert.ToDouble(Console.ReadLine());  
  
        int totalYears = 0;  
        while (balance < targetBalance)  
        {  
            balance *= interestRate;  
            ++totalYears;  
        }  
  
        Console.WriteLine("In {0} year{1} you'll have a balance of {2}.",  
            totalYears, totalYears == 1 ? "" : "s", balance);  
        Console.ReadKey();  
    }  
}
```

Execute the code and enter some values...

```
file:///C:/Users/Administrator/AppData/Local/Temporary Projects/ConsoleApplication1/bin/Debug...
What is your current balance?
10000
What is your current annual interest rate (in %)?
5
What balance would you like to have?
100000
In 48 years you'll have a balance of 10401.2696469421.
```

In the preceding code we have created double variable balance, interestRate, and targetBalance.

```
double balance, interestRate, targetBalance;
Console.WriteLine("What is your current balance?");
balance = Convert.ToDouble(Console.ReadLine());
Console.WriteLine("What is your current annual interest rate (in %)?");
interestRate = 1 + Convert.ToDouble(Console.ReadLine()) / 100.0;
Console.WriteLine("What balance would you like to have?");
targetBalance = Convert.ToDouble(Console.ReadLine());
```

With the help of above code we are just accepting value for the variable balance, interestRate and targetBalance.

```
int totalYears = 0;
while (balance < targetBalance)
{
    balance *= interestRate;
    ++totalYears;
}
```

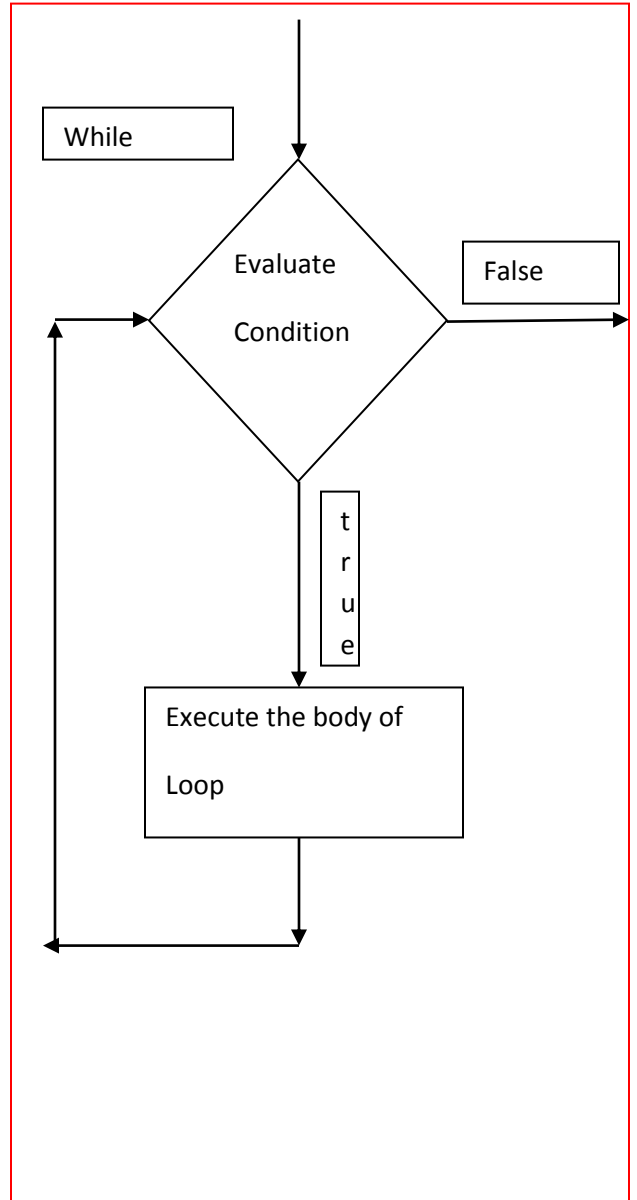
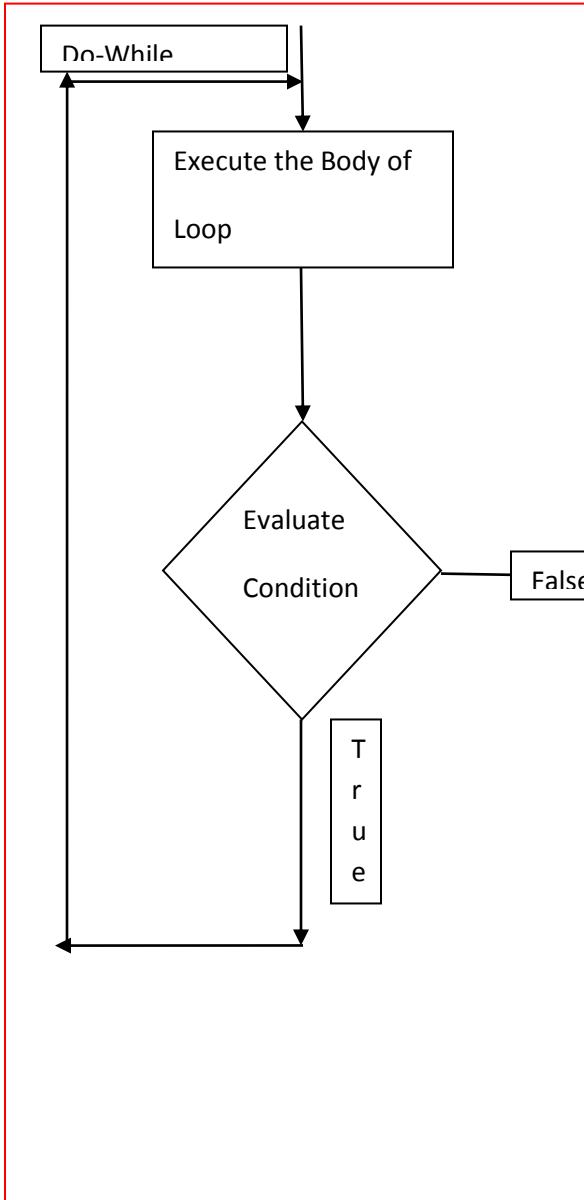
This code is simply repeats the simple annual calculation of the balance with a fixed interest rate as many time as necessary for the balance to satisfy the terminating condition. We are keeping count of how many years have been accounted for by incrementing the totalyears as counter variable with each loop in cycle.

### **Do-While Loop**

Do-While loop is similar to While Loop construct. Both iterate until the specified condition become the false.

However, in the do-While loop, the body of the loop is executed at least once and condition is evaluated for subsequent iteration. And the reason behind that is, the Boolean test in the while loop is take place at the start of loop, not at end. But in While-Loop the Boolean test in the while loop is take place at the end of loop.

The difference between the While and Do-While loop shown in following figure.



Code:

class Program

{

static void Main(string[] args)

{

double balance, interestRate, targetBalance;

Console.WriteLine("What is your current balance?");

balance = Convert.ToDouble(Console.ReadLine());

Console.WriteLine("What is your current annual interest rate (in %)?");

interestRate = 1 + Convert.ToDouble(Console.ReadLine()) / 100.0;

Console.WriteLine("What balance would you like to have?");

targetBalance = Convert.ToDouble(Console.ReadLine());

```

int totalYears = 0;
do
{
    balance *= interestRate;
    ++totalYears;
}
while (balance < targetBalance);
Console.WriteLine("In {0} year{1} you'll have a balance of {2}.",
    totalYears, totalYears == 1 ? "" : "s", balance);
Console.ReadKey();
}
}

```

Execute the code again but this time we are evaluating target balance at bottom.

The screenshot shows a console window titled "file:///C:/Users/Administrator/AppData/Local/Temporary Projects/ConsoleApplication1/bin/Debug...". The text inside the window is as follows:

```

What is your current balance?
1000
What is your current annual interest rate <in %>?
5
What balance would you like to have?
10000
In 48 years you'll have a balance of 10401.2696469421.
_

```

## The For Loop

The for loop structure use to execute the block of statements for a specific number of times. To define a for loop

You need the following information.

- A starting value to initialize the counter variable
- A condition for continuing the loop, involving the counter variable
- An operation to perform on the counter variable at the end of each loop cycle.

The following code is the syntax of the for loop construct.

For (initialization; Test Expression; Increment/Decrement)

```
{
```

Statements.

```
}
```

**Initialization:** The initialization expression initialize the for loop construct. It is executed once at the beginning of the for loop.

**Test Expression:** The test expression determines when to terminate the loop execution. When the expression is evaluated to false, the loop terminates.

**Increment/Decrement:** The increment/decrement component gets invoked after each iteration.

All this components are optional.

You can create infinite loop by omitting all the three expression as shown below...

```
For ( ; ; )
```

```
{
```

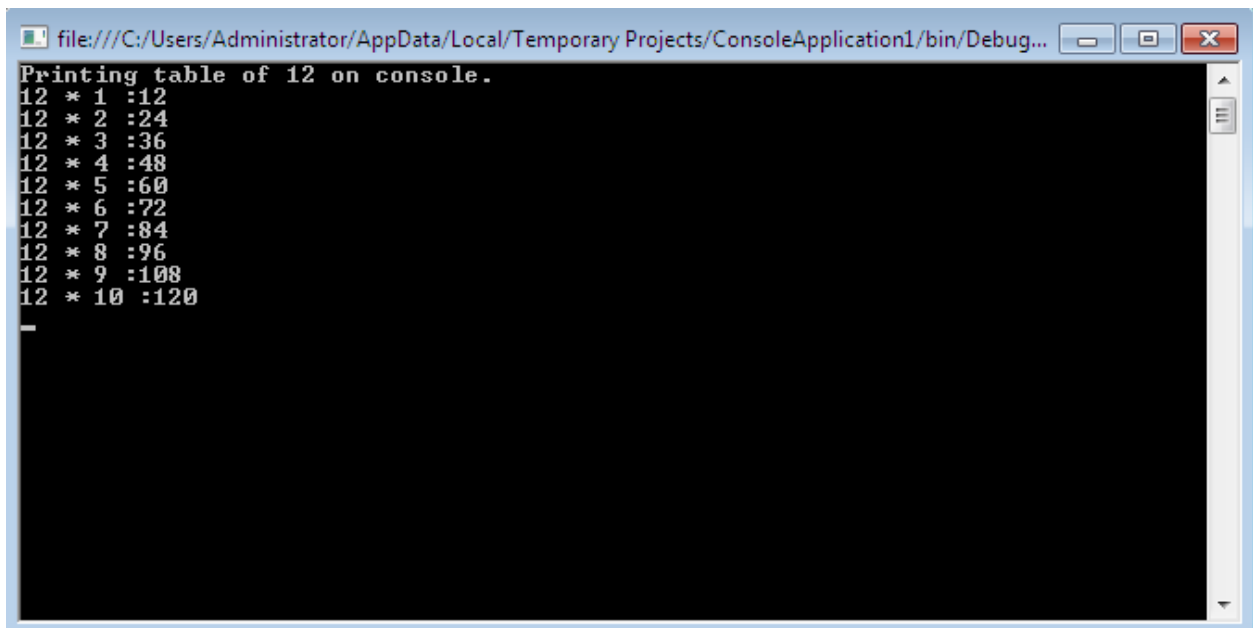
```
... .
```

```
}
```

Let's see one example on for loop. In this example we will print table of 12 on console.

```
class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine("Printing table of 12 on console.");
        for (int x = 1; x <= 10; x++)
        {
            Console.WriteLine("12 * " + x + " : " + x * 12);
        }
        Console.ReadLine();
    }
}
```

Output of preceding code is as follows.

A screenshot of a Windows console application window. The title bar shows the file path: file:///C:/Users/Administrator/AppData/Local/Temporary Projects/ConsoleApplication1/bin/Debug... The console output displays the text "Printing table of 12 on console." followed by ten lines of multiplication results: "12 \* 1 :12", "12 \* 2 :24", "12 \* 3 :36", "12 \* 4 :48", "12 \* 5 :60", "12 \* 6 :72", "12 \* 7 :84", "12 \* 8 :96", "12 \* 9 :108", and "12 \* 10 :120". A cursor is visible on the line following the last output.

```
file:///C:/Users/Administrator/AppData/Local/Temporary Projects/ConsoleApplication1/bin/Debug...
Printing table of 12 on console.
12 * 1 :12
12 * 2 :24
12 * 3 :36
12 * 4 :48
12 * 5 :60
12 * 6 :72
12 * 7 :84
12 * 8 :96
12 * 9 :108
12 * 10 :120
-
```

## Interrupting Loops

You can interrupt the loop in c# by applying some commands...

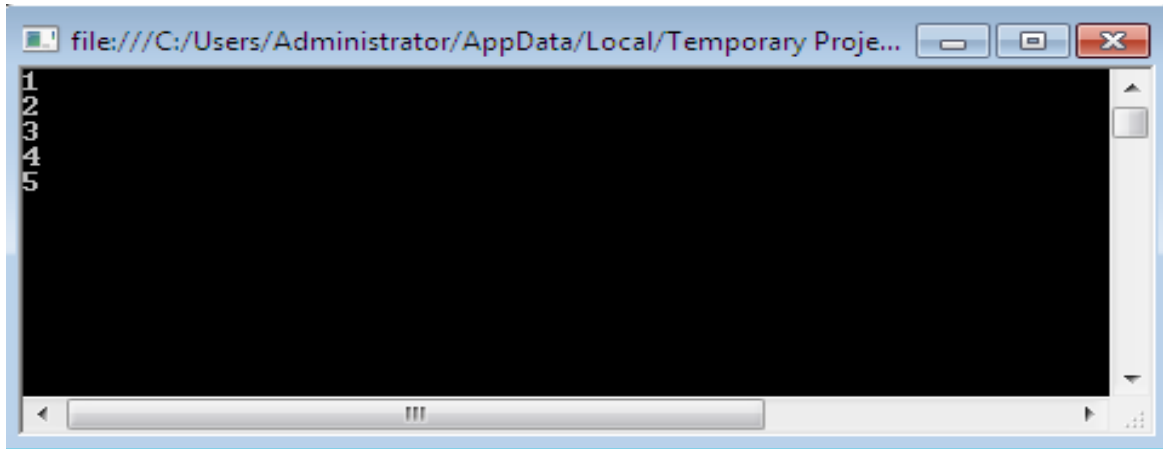
**1) Break:** The Break command simply exit the loop, and execution continues at the first line of code after the loop as shown in the following example

```
static void Main(string[] args)
{
    int i = 1;
    while (i <= 10)
    {
        if (i == 6)
            break;
        Console.WriteLine("{0}", i++);
    }
}
```



```
}  
Console.ReadLine();  
}
```

This code writes out the numbers from 1 to 5 because the break command causes the loop to exit when `i` reaches 6.

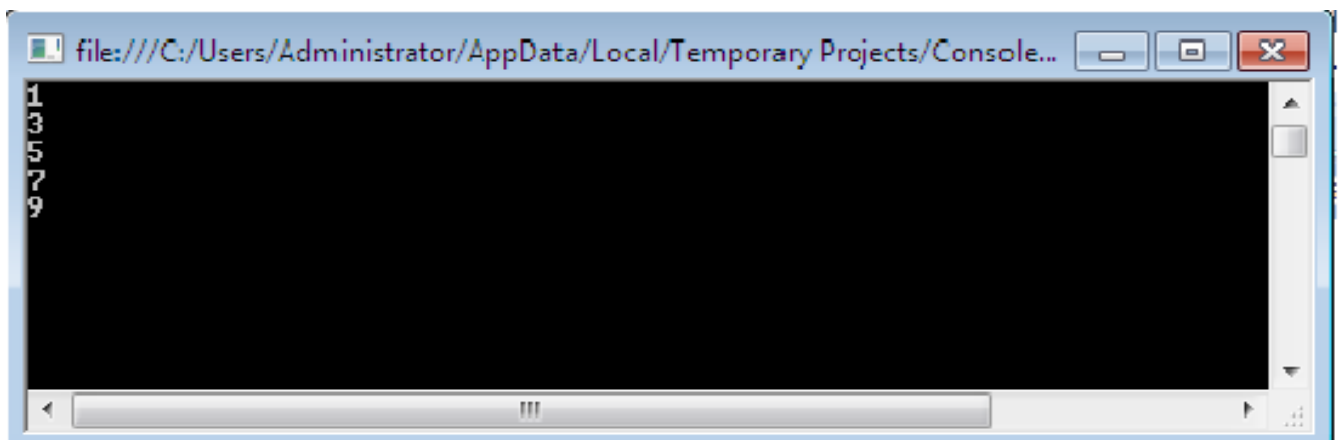


```
file:///C:/Users/Administrator/AppData/Local/Temporary Proje...  
1  
2  
3  
4  
5
```

**2)Continue:** continue only stop the current cycle, not the whole loop, as shown here..

```
int i;  
for (i = 1; i <= 10; i++)  
{  
    if ((i % 2 == 0))  
        continue;  
    Console.WriteLine(i);  
}
```

In the preceding example, whenever the remainder of `i` divide by 2 is Zero, the continue statement stop the execution of current cycle, so output will be....



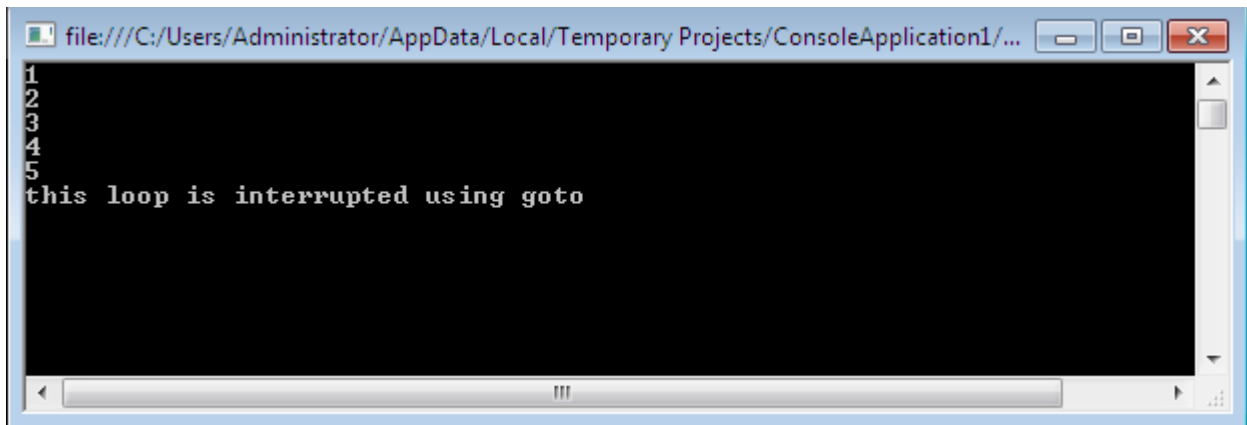
```
file:///C:/Users/Administrator/AppData/Local/Temporary Projects/Console...  
1  
3  
5  
7  
9
```

**3)Goto:** The third method of interrupting the loop is to use goto statement

```
int i = 1;
while (i <= 10)
{
    if (i == 6)
        goto exitPoint;
    Console.WriteLine("{0}", i++);
}
exitPoint:
    Console.WriteLine("this loop is interrupted using goto");
    Console.ReadLine();
```

This code writes out the numbers from 1 to 5 because the goto command causes the loop to exit when i reaches 6.

Output:



```
file:///C:/Users/Administrator/AppData/Local/Temporary Projects/ConsoleApplication1/...
1
2
3
4
5
this loop is interrupted using goto
```

**Summary:**

Looping allows you to execute block of code a number of times according to conditions you specify. You can use do and while loops to execute code while a Boolean expression evaluates to true, and for loop to include the counter in your loop. The loop can be interrupted by continue, break or with the goto statements.

