

Loops

Loop structures let you designate a set of instructions that the computer will perform a set number of times or until some condition changes. Often loops are used to manipulate several related items, using a variable whose value changes each time the computer works through the loop.

==> for

The for loop allows you to iterate through a variable for a specific range of values. You must supply three expressions in a for statement: a variable that is set to an initial value ($i = 0$), a conditional statement that determines when the looping ends ($i < 5$), and an expression that changes the value of the variable with each loop ($i++$ - which adds one to i). For example, the following code loops five times. The value of the variable i starts at 0 and ends at 4, and the output will be the numbers 0 through 4, each on its own line.

```
var i : int;
```

```
for (i = 0; i < 5; i++)  
{  
  trace(i);  
}
```

```
// output from the above code
```

```
0  
1  
2  
3  
4
```

==> for each..in

The for each..in loop iterates through the items of a collection, such as the elements of an array.

```
var myArray:Array = ["one", "two", "three"];
```

```
for each (var item in myArray)  
{  
  trace(item);  
}
```

```
// output:
```

```
one  
two  
three
```

==> **while**

The while loop is like an if statement that repeats as long as the condition is true.

```
var i : int = 0;
```

```
while (i < 3)  
{  
  trace(i);  
  i++;  
}
```

```
// output
```

```
0
```

```
1
```

```
2
```

==> **do..while**

The do..while loop is a while loop that guarantees that the code block is executed at least once, because the condition is checked after the code block is executed.

```
var i:int = 40;
```

```
do  
{  
  trace(i);  
  i++;  
}  
while (i < 5);
```

```
// output
```

```
40
```