```
         break
      case "are":
      case "you":
         print "a is are or you"
         break
      default:
         print "a is none of those"
         break
   endswitch
End
```

Loops

Igor implements two basic types of looping structures: do-while and for loops.

The do-while loop iterates through the loop code and tests an exit condition at the end of each iteration.

The for loop is more complex. The beginning of a for loop includes expressions for initializing and updating variables as well as testing the loop's exit condition at the start of each iteration.

Do-While Loop

The form of the do-while loop structure is:

```
do
   <loop body>
while(<expression>)
```

This loop runs until the expression evaluates to zero or until a break statement is executed.

This example will always execute the body of the loop at least once, like the do-while loop in C.

```
Function Test(lim)
   Variable lim      // We use this parameter as the loop limit.

   Variable sum=0
   Variable i=0      // We use i as the loop variable.
   do
      sum += i       // This is the body; equivalent to sum=sum+i.
      i += 1         // Increment the loop variable.
   while(i < lim)
   return sum
End
```

Nested Do-While Loops

A nested loop is a loop within a loop. Here is an example:

```
Function NestedLoopTest(numOuterLoops, numInnerLoops)
   Variable numOuterLoops, numInnerLoops

   Variable i, j

   i = 0
   do
      j = 0
      do
         <inner loop body>
         j += 1
      while (j < numInnerLoops)
      i += 1
   while (i < numOuterLoops)
End
```