EUPHORIA - FLOW CONTROL

Program execution flow refers to the order in which program statements get executed. By default statements get executed one after the another.

However, there are many times in which the order of execution needs to be different from the default order, to get the job done.

Euphoria has a number of *flow control* statements that you can use to arrange the execution order of statements.

The exit statement:

Exiting a loop is done with the keyword **exit**. This causes flow to immediately leave the current loop and recommence with the first statement after the end of the loop.

Syntax:

The syntax of an exit statement is:

```
exit [ "Label Name" ] [Number]
```

The **exit** statement would terminate the latest and inner most loop until an optional label name or number is specified.

A special form of **exit N** is **exit 0**. This leaves all levels of loop, regardless of the depth. Control continues after the outermost loop block. Likewise, exit -1 exits the second outermost loop, and so on.

Example:

```
#!/home/euphoria-4.0b2/bin/eui

integer b

for a = 1 to 16 do
   printf(1, "value of a %d\n", a)
   if a = 10 then
      b = a
      exit
   end if
end for

printf(1, "value of b %d\n", b)
```

This would produce following result:

```
value of a 1
value of a 2
value of a 3
value of a 4
value of a 5
value of a 6
value of a 7
value of a 8
value of a 9
value of a 10
value of b 10
```

The break statement:

The **break** statement works exactly like the **exit** statement, but applies to if statements or switch

statements rather than to loop statements of any kind.

Syntax:

The syntax of an break statement is:

```
break [ "Label Name" ] [Number]
```

The **break** statement would terminate the latest and inner most if or switch block until an optional label name or number is specified.

A special form of **break N** is **break 0**. This leaves the outer most if or switch block, regardless of the depth. Control continues after the outermost block. Likewise, break -1 breaks the second outermost if or switch block, and so on.

Example:

```
#!/home/euphoria-4.0b2/bin/eui

integer a, b
sequence s = {'E','u', 'p'}

if s[1] = 'E' then
   a = 3
   if s[2] = 'u' then
      b = 1
      if s[3] = 'p' then
         break 0 -- leave topmost if block
      end if
      a = 2
   else
      b = 4
   end if
else
   a = 0
   b = 0
end if
printf(1, "value of a %d\n", a)
printf(1, "value of b %d\n", b)
```

This would produce following result:

```
value of a 3
value of b 1
```

The continue statement:

The **continue** statement continues execution of the loop it applies to by going to the next iteration and skipping the rest of an iteration.

Going to the next iteration means testing a condition variable index and checking whether it is still within bounds.

Syntax:

The syntax of an continue statement is:

```
continue [ "Label Name" ] [Number]
```

The **continue** statement would re-iterate the latest and inner most loop until an optional label name or number is specified.

A special form of **continue N** is **continue 0**. This re-iterate the outer most loop, regardless of the depth. Likewise, continue -1 starts from the second outermost loop, and so on.

Example:

```
#!/home/euphoria-4.0b2/bin/eui

for a = 3 to 6 do
   printf(1, "value of a %d\n", a)
   if a = 4 then
      puts(1,"(2)\n")
      continue
   end if
   printf(1, "value of a %d\n", a*a)
end for
```

This would produce following result:

```
value of a 3
value of a 9
value of a 4
(2)
value of a 5
value of a 25
value of a 6
value of a 36
```

The retry statement:

The **retry** statement continues execution of the loop it applies to by going to the next iteration and skipping the rest of an iteration.

Syntax:

The syntax of an retry statement is:

```
retry [ "Label Name" ] [Number]
```

The **retry** statement retries executing the current iteration of the loop it applies to. The statement branches to the first statement of the designated loop, without testing anything nor incrementing the for loop index.

A special form of **retry N** is **retry 0**. This retries executing the outer most loop, regardless of the depth. Likewise, retry -1 retries the second outermost loop, and so on.

Normally, a sub-block which contains a retry statement also contains another flow control keyword like exit, continue or break, since otherwise the iteration would be endlessly executed.

Example:

```
#!/home/euphoria-4.0b2/bin/eui

integer errors = 0
integer files_to_open = 10

for i = 1 to length(files_to_open) do
   fh = open(files_to_open[i], "rb")
   if fh = -1 then
      if errors > 5 then
         exit
      else
         errors += 1
         retry
      end if
   end if
   file_handles[i] = fh
end for
```

Since retry does not change the value of i and tries again opening the same file, there has to be a way to break from the loop, which the exit statement provides.

The goto statement:

The **goto** statement instructs the computer to resume code execution at a labeled place.

The place to resume execution is called the target of the statement. It is restricted to lie in the current routine, or the current file if outside any routine.

Syntax:

The syntax of an goto statement is:

```
goto "Label Name"
```

The target of a goto statement can be any accessible **label** statement:

```
label "Label Name"
```

Label names must be double quoted constant strings. Characters that would be illegal in an Euphoria identifier may appear in a label name, since it is a regular string.

Example:

```
#!/home/euphoria-4.0b2/bin/eui

integer a = 0

label "FIRST"
printf(1, "value of a %d\n", a)
a += 10
if a < 50 then
   goto "FIRST"
end if
printf(1, "Final value of a %d\n", a)
```

This would produce following result:

```
value of a 0
value of a 10
value of a 20
value of a 30
value of a 40
Final value of a 50
```