Flow Control

Changing your program's behaviour

Flow control is all about getting your program to do:

- **Branching**: respond appropriately under different conditions. That is, you can program in alternatives your program can take depending on the present situation. Without branching, you can only write simple programs, as we have been up to this time. It's time to level up!
- **Iteration**: getting your program to do the same set of actions repeatedly. Iteration is what computers are good at. It's really what your program is doing most of the time.

Both Branching and Iteration are big topics: Smojo offers us powerful words to create new methods for branching. It also has many ways to perform iteration. So, we will revisit both these topics in greater detail later, once you have more Smojo under your belt.

Right now, let's start with just the basics.

Two Learning Tracks!

Before we begin, I'd like to offer you two learning tracks from this Chapter onwards:

A. The main track is for everyone.

B. The advanced track is if you are already proficient in programming.

I'll indicate the advanced sections with an asterisk * in the section headings and Quizzes. The number of asterisks will indicate the level of difficulty. If you're new to programming skip these sections and their quizzes. They won't affect your understanding. Come back to them later when you are more confident of your skills.

If you’re already a seasoned programmer, you should attempt as many of the asterisked sections as possible.

Branching

The workhorse of branching are the pair of words `->` (called IF) and `|.` (called THEN).

Consider the code below:

```
1 : temperature ( n — )
2     30 > -> “It’s hot!” . exit |.
3     “Cold” .
4 ;
5 35 temperature cr
6 12 temperature cr
```

Note that line numbers are shown in cyan, but are not part of the program. If this program is executed it displays:

```
It’s hot
Cold
ok
```

I suggest you type in this program and run it before proceeding with the discussion.

The word `->` examines the top of the stack.

- If it evaluates to **`true`**, then the code **between** `->` and `|.` is executed.
- Otherwise, the program resumes **after** the `|.`

In line 5 the phrase, `35 temperature` triggers the comparison:

```
35 30 >
```

which will push **`true`** onto the data stack. This is picked up by `->` and the code within, `“It’s hot” . exit` will be executed.

The action of `exit` is to get out of the current word, ie, the program resumes at a point after the enclosing word (ie, `temperature`) was last used. In this case, that would be line 6.

In line 6, the phrase `12 temperature` triggers the comparison

```
12 30 >
```

which pushes `false` onto the data stack. This is again picked up by `->` which then skips to the end of `|.` bringing us to line 3.

So `Cold` is displayed to the user and the temperature word is exited normally.

Be sure you understand these steps thoroughly before proceeding.

Quiz 3.0

3.0.0: Write a word `iabs` that returns the absolute value of an integer. Ie:

`24 iabs` returns `24`,

`-34 iabs` returns `34` and

`0 iabs` returns `0`.

3.0.1: Write a word `2dup ( a b -- a b a b )` that duplicates the top two items on the stack.

3.0.2: Write a word `imax ( n n -- n )` that determines the maximum of two integers. Write **three versions** - one using locals only another using stack words only and the third mixing them as appropriate. Of course you need to use the branching words in all three versions. Hint: You may need to use `2dup` from 3.0.1.

Answers to Quizzes

Quiz 3.0.0

3.0.0:

```
: iabs ( n — n )
   dup 0 >= —> exit |. -1 *
;
```

3.0.1:

```
: 2dup ( a b — a b a b )
   over over
;
```

3.0.2:

Version #1 with Locals only:

```
: imax ( n n — n ) { a b }
   a b < —> b exit |. a
;
```

Version #2 with Stack Words only:

```
: imax ( n n — n )
   2dup < —> nip exit |. drop
;
```

Version #3 (mixed):

```
: imax ( n n — n )
   2dup { a b }
   < —> b exit |. a
;
```

Appendix: Branching Words

Word	Action

`if … then`
 	
`if … else … then`
	A conditional block. The code within the `if` ... `then` is executed only if the top of the stack is true.
	
	The second form is used to execute code between `else` ... `then` if the top of stack is false.
	
	Note that `if` , `else` and `then` are separate words, often used together.

`-> | |.`	A multiway branch paired with `|` or `|.` Every multiway branch must be completed using `|.`

`_ -> … |.`
 	
`otherwise … |.`
	`_` (underscore) is a synonym for `true`.
	
	The combination `_ -> … |.` is equivalent to:
	
	`otherwise … |.`

`begin … again`	Represents an infinite loop. Break out of it using `exit`
`exit`	Exits a word.