

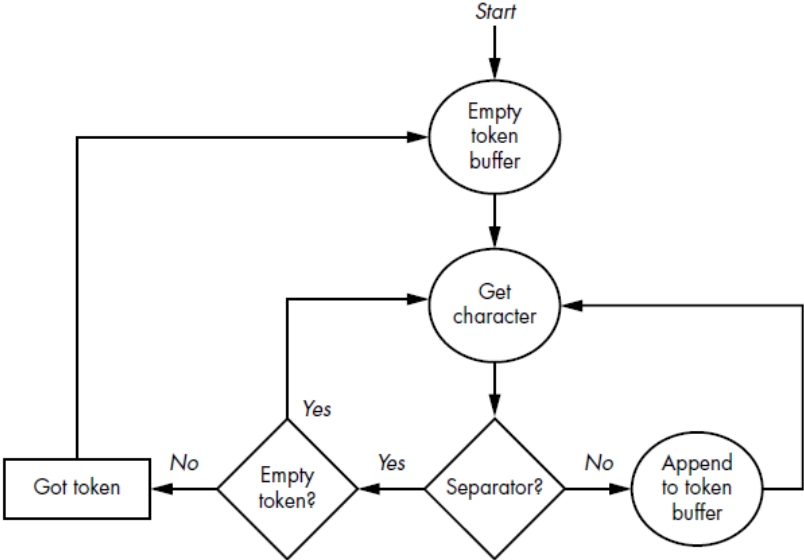
The Secret Life of Programs

Understand Computers—Craft Better Code

by Jonathan E. Steinhart

errata updated to print 4

Page	Error	Correction	Print corrected																																																
13	Figure replacement	$ \begin{array}{r} +1 \quad '0 \quad '0 \quad '0 \quad '0 \\ -1 \quad \underline{1 \quad 1 \quad 1 \quad 1} \\ \hline 0 \quad 0 \quad 0 \quad 0 \end{array} $ <p>Figure 1-12: Finding -1</p>	Print 3																																																
22	<table border="1"> <thead> <tr> <th>Dec</th> <th>Hex</th> <th>Char</th> <th>Dec</th> <th>Hex</th> <th>Char</th> <th>Dec</th> <th>Hex</th> <th>Char</th> <th>Dec</th> <th>Hex</th> <th>Char</th> </tr> </thead> <tbody> <tr> <td>16</td> <td>10</td> <td>DLE</td> <td>48</td> <td>30</td> <td>0</td> <td>80</td> <td>5</td> <td>P</td> <td>112</td> <td>70</td> <td>p</td> </tr> </tbody> </table>	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	16	10	DLE	48	30	0	80	5	P	112	70	p	<table border="1"> <thead> <tr> <th>Dec</th> <th>Hex</th> <th>Char</th> <th>Dec</th> <th>Hex</th> <th>Char</th> <th>Dec</th> <th>Hex</th> <th>Char</th> <th>Dec</th> <th>Hex</th> <th>Char</th> </tr> </thead> <tbody> <tr> <td>16</td> <td>10</td> <td>DLE</td> <td>48</td> <td>30</td> <td>0</td> <td>80</td> <td>50</td> <td>P</td> <td>112</td> <td>70</td> <td>p</td> </tr> </tbody> </table>	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	16	10	DLE	48	30	0	80	50	P	112	70	p	Print 2
Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char																																								
16	10	DLE	48	30	0	80	5	P	112	70	p																																								
Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char																																								
16	10	DLE	48	30	0	80	50	P	112	70	p																																								
25	Figure replacement	<p>The diagram illustrates the bit-level conversion of Unicode characters to UTF-8. It shows four examples:</p> <ul style="list-style-type: none"> Unicode A (0x0041): A 16-bit binary sequence (0000000001000001) is mapped to a single 8-bit UTF-8 byte (01000001). Unicode π (0x03C0): A 16-bit binary sequence (0000001111000000) is mapped to two 8-bit UTF-8 bytes (11001111 and 10000000). Unicode ♣ (0x2663): A 16-bit binary sequence (0001001100110000) is mapped to three 8-bit UTF-8 bytes (11100110, 10011001, and 10100011). 	Print 3																																																

Page	Error	Correction	Print corrected
95	This means we can't do things like form a long word from bytes 5, 6, 7, and 8, because that would mean that the bus would have to make two trips: one to building 0 and one to building 1 .	This means we can't do things like form a long word from bytes 5, 6, 7, and 8, because that would mean that the bus would have to make two trips: one to building 1 and one to building 2 .	Print 4
158	One of Metcalfe's big innovations was <i>random back-off-and-retry</i>.	Metcalfe used <i>random back-off-and-retry</i>, an innovation pioneered by ALOHAnet, a packetswitched radio network developed at the University of Hawaii.	Print 3
164	Deletion	In Figure 6-29, you can see that we're using a comparator to test the sampled value in the holding tank against the value of the DAC. Once cleared, the counter counts up until the DAC value hits the sampled value, at which time the counter is disabled and we're done.	Print 4
177	And the concept of a <i>user</i> appeared so that machines could tell what belonged to who .	And the concept of a <i>user</i> appeared so that machines could tell what belonged to whom .	Print 3
209	Figure 7-3 6	Figure 7-3 5	Print 4
221	Figure replacement	 <pre> graph TD Start([Start]) --> EmptyTokenBuffer([Empty token buffer]) EmptyTokenBuffer --> GetCharacter([Get character]) GetCharacter --> Separator{Separator?} Separator -- Yes --> AppendTokenBuffer([Append to token buffer]) AppendTokenBuffer --> GetCharacter Separator -- No --> EmptyToken{Empty token?} EmptyToken -- Yes --> GetCharacter EmptyToken -- No --> GotToken[Got token] GotToken --> EmptyTokenBuffer </pre> <p data-bbox="1058 1243 1381 1273"><i>Figure 8-1: Simple lexical analysis</i></p>	Print 3