



2022 Spring

CS315

Homework Assignment 2

Logically Controlled Loops in Dart, Javascript, Perl, PHP, and
Python

Name: Ahmet Faruk

Surname: Ulutaş

Section: 1

ID: 21803717

TA Name: Irmak Türköz

What is the Pretest Loop?

Before the instructions within the loop are performed, the condition required to continue looping is verified.

What is the Post-test Loop?

The condition is placed at the conclusion of the loop, and if the condition is true, the loop is terminated with the keyword EXIT.

What is conditional and unconditional exit?

Conditional execution determines whether or not an instruction is executed by the kernel. The condition property on most instructions decides whether the kernel executes it based on the condition flags set. The processor checks the condition attribute to the condition flags before executing. The instruction is performed if they match; else, the instruction is ignored. The non-comparative variant of conditional execution is unconditional execution.

What is a labelled and unlabeled loop?

The outer loop is terminated with a labelled break statement, whereas the satisfied loop is terminated with an unlabeled break statement.

Dart

A while or for loop can be used to create a pretest loop in Dart. A do-while loop can be used to create a post-test loop. There are continue and break labels that work as unlabeled in Dart. Continue to the next iteration. With break, the loop is terminated.

Full Code Example:

```
1 void main() {
2     String val0 = "val0";
3     int count = 0;
4     int otherCount = 0;
5
6     // PRETEST
7     print("Pretest with while loop");
8
9     // Values before the test
10    print("Before the test, value of val0: $val0");
11    print("Before the test, value of count: $count");
12
13    // In pretest, before the instructions within the loop are performed,
14    // the condition required to continue looping is verified
15    while (count < 3) {
16        print("val0: $val0, count: $count");
17        count += 1;
18        val0 = "val$count";
19    }
20
21    // Values after the test
22    print("After the test, value of val0: $val0");
23    print("After the test, value of count: $count");
24
25    // Pretest completed.
26    // The second while loop did not work.
27    // Because the value is checked before execution.
28    while (count < 3) {
29        print("val0: $val0, count: $count");
30        count += 1;
31        val0 = "val$count";
32    }
33
34    if ( count <= 3 ) {
35        print("Pretest completed. The second while loop did not work.");
36        print("Since the value is checked before execution. \n");
37    }
38
39    // POST-TEST
40    print("Post-test with do-while loop");
41
42    // values are refreshed
43    val0 = "val0";
44    count = 0;
45
46    // Values before the test
47    print("Before the test, value of val0: $val0");
48    print("Before the test, value of count: $count");
49
50    // The condition is placed at the conclusion of the loop,
51    // and if the condition is true, the loop is terminated with the keyword EXIT.
52    do {
53        print("val0: $val0, count: $count");
54        count += 1;
55        val0 = "val$count";
56    } while (count < 3);
57
58    // Values after the test
59    print("After the test, value of val0: $val0");
60    print("After the test, value of count: $count");
61
62    // The condition is placed at the conclusion of the loop,
63    // and if the condition is true, the loop is terminated with the keyword EXIT.
64    do {
65        print("val0: $val0, count: $count");
66        count += 1;
67        val0 = "val$count";
68    } while (count < 3);
69
70    print("After the second test, value of val0: $val0");
71    print("After the second test, value of count: $count");
72
73    // Post-test completed. The second while loop ran once. Since the value is checked after execution.
74    if ( count > 3 ) {
75        print("Post-test completed. The second while loop ran once.");
76        print("Since the value is checked after execution. \n");
77    }
78
79    // Unlabeled Loop Test
80    print("Unlabeled test while loop");
81
82    // values are refreshed
83    val0 = "val0";
84    count = 0;
85
86    // Values before the test
87    print("Before the test, value of val0: $val0");
88    print("Before the test, value of count: $count");
89
90    // The break statement ends the loop with exit.
91    // Since this is unlabeled, it worked only in the closest loop.
92    while (otherCount < 2) {
93        print("Outer while loop otherCount:$otherCount");
94        count = 0;
95        while (count < 3) {
96            print("val0: $val0, count: $count");
97            count += 1;
98            val0 = "val$count";
99            if ( count == 2 ) {
100                print("Break found, exited loop at 2");
101                break;
102            }
103        }
104        otherCount += 1;
105    }
106
107    // Values after the test
108    print("After the test, value of val0: $val0");
109    print("After the test, value of count: $count");
110
111    // values are refreshed
112    val0 = "val0";
113    count = 0;
114    otherCount = 0;
115}
```

```

116 // Values before the test
117 print("Before the test, value of val0: $val0");
118 print("Before the test, value of count: $count");
119
120 // Break test completed
121 print("Unlabeled Break test completed. \n");
122
123 // The continue statement skips the iteration.
124 // Since this is unlabeled, it worked only in the closest loop.
125 while (otherCount < 2) {
126     print("Outer while loop otherCount:$otherCount");
127     count = 0;
128     while (count < 3) {
129         count += 1;
130         val0 = "val$count";
131         if ( count == 2 ) {
132             print("Continue found, skipped loop at 2");
133             continue;
134         }
135         print("val0: $val0, count: $count");
136     }
137     otherCount += 1;
138 }
139
140 // Values after the test
141 print("After the test, value of val0: $val0");
142 print("After the test, value of count: $count");
143
144 // Continue test completed
145 print("Unlabeled continue test completed. \n");
146
147 // Labeled Loop Test
148 print("Labeled test while loop");
149
150 // values are refreshed
151 val0 = "val0";
152 count = 0;
153 otherCount = 0;

```

```

155 // Values before the test
156 print("Before the test, value of val0: $val0");
157 print("Before the test, value of count: $count");
158
159 // The break statement ends the loop with exit.
160 // Since this is labeled, it worked in the specified label's loop.
161 outerBreak:
162 while (otherCount <= 2) {
163     print("Outer while loop otherCount:$otherCount");
164     count = 0;
165     otherCount += 1;
166     while (count < 3) {
167         print("val0: $val0, count: $count");
168         count += 1;
169         val0 = "val$count";
170         if ( count == 2 ) {
171             print("Break found, exited loop at 2");
172             break outerBreak;
173         }
174     }
175 }
176
177 // Values after the test
178 print("After the test, value of val0: $val0");
179 print("After the test, value of count: $count");
180
181 // values are refreshed
182 val0 = "val0";
183 count = 0;
184 otherCount = 0;
185
186 // Values before the test
187 print("Before the test, value of val0: $val0");
188 print("Before the test, value of count: $count");
189
190 // Break test completed
191 print("Labeled Break test completed. \n");
192

```

```

193 // The continue statement skips the iteration.
194 // Since this is labeled, it worked in the specified label's loop.
195 outerContinue:
196 while (otherCount <= 2) {
197     print("Outer while loop otherCount:$otherCount");
198     count = 0;
199     otherCount += 1;
200     while (count < 3) {
201         count += 1;
202         val0 = "val$count";
203         if ( count == 2 ) {
204             print("Continue found, skipped loop at 2");
205             continue outerContinue;
206         }
207         print("val0: $val0, count: $count");
208     }
209 }
210
211 // Values after the test
212 print("After the test, value of val0: $val0");
213 print("After the test, value of count: $count");
214
215 // Continue test completed
216 print("Labeled continue test completed. \n");
217 }

```

Sample Output For Code:

Console

```
Pretest with while loop
Before the test, value of val0: val0
Before the test, value of count: 0
val0: val0, count: 0
val0: val1, count: 1
val0: val2, count: 2
After the test, value of val0: val3
After the test, value of count: 3
Pretest completed. The second while loop did not work.
Since the value is checked before execution.

Post-test with do-while loop
Before the test, value of val0: val0
Before the test, value of count: 0
val0: val0, count: 0
val0: val1, count: 1
val0: val2, count: 2
After the test, value of val0: val3
After the test, value of count: 3
val0: val3, count: 3
After the second test, value of val0: val4
After the second test, value of count: 4
Post-test completed. The second while loop ran once. Since the value is checked after execution.
```

```
Unlabeled test while loop
Before the test, value of val0: val0
Before the test, value of count: 0
Outer while loop otherCount:0
val0: val0, count: 0
val0: val1, count: 1
Break found, exited loop at 2
Outer while loop otherCount:1
val0: val2, count: 0
val0: val1, count: 1
Break found, exited loop at 2
After the test, value of val0: val2
After the test, value of count: 2
Before the test, value of val0: val0
Before the test, value of count: 0
Unlabeled Break test completed.
```

```
Outer while loop otherCount:0
val0: val1, count: 1
Continue found, skipped loop at 2
val0: val3, count: 3
Outer while loop otherCount:1
val0: val1, count: 1
Continue found, skipped loop at 2
val0: val3, count: 3
After the test, value of val0: val3
After the test, value of count: 3
Unlabeled continue test completed.
```

```
Labeled test while loop
Before the test, value of val0: val0
Before the test, value of count: 0
Outer while loop otherCount:0
val0: val0, count: 0
val0: val1, count: 1
Break found, exited loop at 2
After the test, value of val0: val2
After the test, value of count: 2
Before the test, value of val0: val0
Before the test, value of count: 0
Labeled Break test completed.
```

```
Outer while loop otherCount:0
val0: val1, count: 1
Continue found, skipped loop at 2
Outer while loop otherCount:1
val0: val1, count: 1
Continue found, skipped loop at 2
Outer while loop otherCount:2
val0: val1, count: 1
Continue found, skipped loop at 2
After the test, value of val0: val2
After the test, value of count: 2
Labeled continue test completed.
```

JavaScript

A while loop can be used to create a pretest loop in JavaScript. A do-while loop can be used to create a post-test loop. There are continue and break keywords that work in JavaScript. Continue to the next iteration. With break, the loop is terminated.

Full Code Example:

```
main.js
1 var val0 = "val0";
2 var count = 0;
3 var otherCount = 0;
4
5 // PRETEST
6 console.log("Pretest with while loop");
7
8 // Values before the test
9 console.log("Before the test, value of val0: " + val0);
10 console.log("Before the test, value of count: " + count);
11
12 // In pretest, before the instructions within the loop are performed,
13 // the condition required to continue looping is verified
14 while (count < 3) {
15     console.log("val0: " + val0 + " count: " + count);
16     count += 1;
17     val0 = "val" + count;
18 }
19
20 // Values after the test
21 console.log("After the test, value of val0: " + val0);
22 console.log("After the test, value of count: " + count);
23
24 // Pretest completed.
25 // The second while loop did not work.
26 // Because the value is checked before execution.
27 while (count < 3) {
28     console.log("val0: " + val0 + " count: " + count);
29     count += 1;
30     val0 = "val" + count;
31 }
32
33 if ( count <= 3 ) {
34     console.log("Pretest completed. The second while loop did not work.");
35     console.log("Since the value is checked before execution. \n");
36 }
37
38 // POST-TEST
39 console.log("Post-test with do-while loop");
40
41 // values are refreshed
42 val0 = "val0";
43 count = 0;
44
45 // Values before the test
46 console.log("Before the test, value of val0: " + val0);
47 console.log("Before the test, value of count: " + count);
48
49 // The condition is placed at the conclusion of the loop,
50 // and if the condition is true, the loop is terminated with the keyword EXIT.
51 do {
52     console.log("val0: " + val0 + " count: " + count);
53     count += 1;
54     val0 = "val" + count;
55 } while (count < 3);
56
57 // Values after the test
58 console.log("After the test, value of val0: " + val0);
59 console.log("After the test, value of count: " + count);
60
61 // The condition is placed at the conclusion of the loop,
62 // and if the condition is true, the loop is terminated with the keyword EXIT.
63 do {
64     console.log("val0: " + val0 + " count: " + count);
65     count += 1;
66     val0 = "val" + count;
67 } while (count < 3);
68
69 console.log("After the second test, value of val0: " + val0);
70 console.log("After the second test, value of count: " + count);
71
72 // Post-test completed. The second while loop ran once. Since the value is checked after execution.
73 if ( count > 3 ) {
74     console.log("Post-test completed. The second while loop ran once.");
75     console.log("Since the value is checked after execution. \n");
76 }
77
78 // Unlabeled Loop Test
79 console.log("Unlabeled test while loop");
80
81 // values are refreshed
82 val0 = "val0";
83 count = 0;
84
85 // Values before the test
86 console.log("Before the test, value of val0: " + val0);
87 console.log("Before the test, value of count: " + count);
88
89 // The break statement ends the loop with exit.
90 // Since this is unlabeled, it worked only in the closest loop.
91 while (otherCount < 2) {
92     console.log("Outer while loop otherCount: " + otherCount);
93     count = 0;
94     while (count < 3) {
95         console.log("val0: " + val0 + " count: " + count);
96         count += 1;
```

```

97     val0 = "val" + count;
98     if ( count == 2 ) {
99         console.log("Break found, exited loop at 2");
100         break;
101     }
102 }
103 otherCount += 1;
104 }
105
106 // Values after the test
107 console.log("After the test, value of val0: " + val0);
108 console.log("After the test, value of count: " + count);
109
110 // values are refreshed
111 val0 = "val0";
112 count = 0;
113 otherCount = 0;
114
115 // Values before the test
116 console.log("Before the test, value of val0: " + val0);
117 console.log("Before the test, value of count: " + count);
118
119 // Break test completed
120 console.log("Unlabeled Break test completed. \n");
121
122 // The continue statement skips the iteration.
123 // Since this is unlabeled, it worked only in the closest loop.
124 while (otherCount < 2) {
125     console.log("Outer while loop otherCount: " + otherCount);
126     count = 0;
127     while (count < 3) {
128         count += 1;
129         val0 = "val" + count;
130         if ( count == 2 ) {
131             console.log("Continue found, skipped loop at 2");
132             continue;
133         }
134         console.log("val0: " + val0 + " count: " + count);
135     }
136     otherCount += 1;
137 }
138
139 // Values after the test
140 console.log("After the test, value of val0: " + val0);
141 console.log("After the test, value of count: " + count);
142
143 // Continue test completed
144 console.log("Unlabeled continue test completed. \n");

```

```

146 // Labeled Loop Test
147 console.log("Labeled test while loop");
148
149 // values are refreshed
150 val0 = "val0";
151 count = 0;
152 otherCount = 0;
153
154 // Values before the test
155 console.log("Before the test, value of val0: " + val0);
156 console.log("Before the test, value of count: " + count);
157
158 // The break statement ends the loop with exit.
159 // Since this is labeled, it worked in the specified label's loop.
160 outerBreak:
161 while (otherCount <= 2) {
162     console.log("Outer while loop otherCount: " + otherCount);
163     count = 0;
164     otherCount += 1;
165     while (count < 3) {
166         console.log("val0: " + val0 + " count: " + count);
167         count += 1;
168         val0 = "val" + count;
169         if ( count == 2 ) {
170             console.log("Break found, exited loop at 2");
171             break outerBreak;
172         }
173     }
174 }
175
176 // Values after the test
177 console.log("After the test, value of val0: " + val0);
178 console.log("After the test, value of count: " + count);
179
180 // Break test completed
181 console.log("Labeled Break test completed. \n");
182
183 // values are refreshed
184 val0 = "val0";
185 count = 0;
186 otherCount = 0;
187
188 // Values before the test
189 console.log("Before the test, value of val0: " + val0);
190 console.log("Before the test, value of count: " + count);

```

```

192 // The continue statement skips the iteration.
193 // Since this is labeled, it worked in the specified label's loop.
194 outerContinue:
195 while (otherCount <= 2) {
196     console.log("Outer while loop otherCount: " + otherCount);
197     count = 0;
198     otherCount += 1;
199     while (count < 3) {
200         count += 1;
201         val0 = "val" + count;
202         if ( count == 2 ) {
203             console.log("Continue found, skipped loop at 2");
204             continue outerContinue;
205         }
206         console.log("val0: " + val0 + " count: " + count);
207     }
208 }
209
210 // Values after the test
211 console.log("After the test, value of val0: " + val0);
212 console.log("After the test, value of count: " + count);
213
214 // Continue test completed
215 console.log("Labeled continue test completed. \n");

```

Sample Output For Code:

Output Clear

```

node /tmp/xhrwLa6kEF-.js
Pretest with while loop
Before the test, value of val0: val0
Before the test, value of count: 0
val0: val0 count: 0
val0: val1 count: 1
val0: val2 count: 2
After the test, value of val0: val3
After the test, value of count: 3
Pretest completed. The second while loop did not work.
Since the value is checked before execution.

Post-test with do-while loop
Before the test, value of val0: val0
Before the test, value of count: 0
val0: val0 count: 0
val0: val1 count: 1
val0: val2 count: 2
After the test, value of val0: val3
After the test, value of count: 3
val0: val3 count: 3
After the second test, value of val0: val4
After the second test, value of count: 4
Post-test completed. The second while loop ran once.
Since the value is checked after execution.

Unlabeled test while loop
Before the test, value of val0: val0
Before the test, value of count: 0
Outer while loop otherCount: 0
val0: val0 count: 0
val0: val1 count: 1
Break found, exited loop at 2
Outer while loop otherCount: 1
val0: val2 count: 0
val0: val1 count: 1
Break found, exited loop at 2
After the test, value of val0: val2
After the test, value of count: 2
Before the test, value of val0: val0
Before the test, value of count: 0
Unlabeled Break test completed.

Outer while loop otherCount: 0
val0: val1 count: 1
Continue found, skipped loop at 2
val0: val3 count: 3
Outer while loop otherCount: 1
val0: val1 count: 1
Continue found, skipped loop at 2
val0: val3 count: 3
After the test, value of val0: val3
After the test, value of count: 3
Unlabeled continue test completed.

Labeled test while loop
Before the test, value of val0: val0
Before the test, value of count: 0
Outer while loop otherCount: 0
val0: val0 count: 0
val0: val1 count: 1
Break found, exited loop at 2
After the test, value of val0: val2
After the test, value of count: 2
Labeled Break test completed.

Before the test, value of val0: val0
Before the test, value of count: 0
Outer while loop otherCount: 0
val0: val1 count: 1
Continue found, skipped loop at 2
Outer while loop otherCount: 1
val0: val1 count: 1
Continue found, skipped loop at 2
Outer while loop otherCount: 2
val0: val1 count: 1
Continue found, skipped loop at 2
After the test, value of val0: val2
After the test, value of count: 2
Labeled continue test completed.

```

```

Outer while loop otherCount: 0
val0: val1 count: 1
Continue found, skipped loop at 2
val0: val3 count: 3
Outer while loop otherCount: 1
val0: val1 count: 1
Continue found, skipped loop at 2
val0: val3 count: 3
After the test, value of val0: val3
After the test, value of count: 3
Unlabeled continue test completed.

Labeled test while loop
Before the test, value of val0: val0
Before the test, value of count: 0
Outer while loop otherCount: 0
val0: val0 count: 0
val0: val1 count: 1
Break found, exited loop at 2
After the test, value of val0: val2
After the test, value of count: 2
Labeled Break test completed.

Before the test, value of val0: val0
Before the test, value of count: 0
Outer while loop otherCount: 0
val0: val1 count: 1
Continue found, skipped loop at 2
Outer while loop otherCount: 1
val0: val1 count: 1
Continue found, skipped loop at 2
Outer while loop otherCount: 2
val0: val1 count: 1
Continue found, skipped loop at 2
After the test, value of val0: val2
After the test, value of count: 2
Labeled continue test completed.

```


Perl

A while or until loop can be used to create a pretest loop in Perl. Do-while loop can be used to create a post-test loop. There are last, next, and redo labels that work on any enclosing block in Perl. With goto, the program jumps to the label.

Full Code Example:

```
1  $val0 = "val0";
2  $count = 0;
3  $otherCount = 0;
4
5  # PRETEST
6  print("Pretest with while loop\n");
7
8  # Values before the test
9  print "Before the test, value of val0: " , $val0, "\n";
10 print "Before the test, value of count: " , $count, "\n";
11
12 # In pretest, before the instructions within the loop are performed,
13 # the condition required to continue looping is verified
14 while ($count < 3) {
15     print "val0: " , $val0 , " count: " , $count, "\n";
16     $count += 1;
17 }
18
19 # Values after the test
20 print "After the test, value of val0: " , $val0, "\n";
21 print "After the test, value of count: " , $count, "\n";
22
23 # Pretest completed.
24 # The second while loop did not work.
25 # Because the value is checked before execution.
26 while ($count < 3) {
27     print "val0: " + $val0 + " count: " , $count, "\n";
28     $count += 1;
29 }
30
31 if ( $count <= 3 ) {
32     print "Pretest completed. The second while loop did not work.", "\n";
33     print "Since the value is checked before execution. \n\n";
34 }
35
36 # POST-TEST
37 print("Post-test with do-while loop \n");
38
39 # values are refreshed
40 $val0 = "val0";
41 $count = 0;
42
43 # Values before the test
44 print "Before the test, value of val0: " , $val0, "\n";
45 print "Before the test, value of count: " , $count, "\n";
46
47 # The condition is placed at the conclusion of the loop,
48 # and if the condition is true, the loop is terminated with the keyword EXIT.
49 do {
50     print "val0: " , $val0 , " count: " , $count, "\n";
51     $count += 1;
52 } while ($count < 3);
53
54 # Values after the test
55 print "After the test, value of val0: " , $val0, "\n";
56 print "After the test, value of count: " , $count, "\n";
57
58 # The condition is placed at the conclusion of the loop,
59 # and if the condition is true, the loop is terminated with the keyword EXIT.
60 do {
61     print "val0: " , $val0 , " count: " , $count, "\n";
62     $count += 1;
63 } while ($count < 3);
64
65 print "After the second test, value of val0: " , $val0, "\n";
66 print "After the second test, value of count: " , $count, "\n";
67
68 # Post-test completed. The second while loop ran once. Since the value is checked after execution
69 if ( $count > 3 ) {
70     print("Post-test completed. The second while loop ran once. \n");
71     print("Since the value is checked after execution. \n\n");
72 }
73
74 # Unlabeled Loop Test
75 print("Unlabeled test while loop \n");
76
77 # values are refreshed
78 $val0 = "val0";
79 $count = 0;
80
81 # Values before the test
82 print "Before the test, value of val0: " , $val0, "\n";
83 print "Before the test, value of count: " , $count, "\n";
84
85 # The break statement ends the loop with exit.
86 # Since this is unlabeled, it worked only in the closest loop.
87 while ($otherCount < 2) {
88     print("Outer while loop otherCount: $otherCount \n");
89     $count = 0;
90     while ($count < 3) {
91         print("val0: $val0 count: $count \n");
92         $count += 1;
93         if ( $count == 2 ) {
94             print("Break found, exited loop at 2 \n");
95             break;
96         }
97     }
98 }
```

```

98     $otherCount += 1;
99 }
100
101 # Values after the test
102 print "After the test, value of val0: " . $val0, "\n";
103 print "After the test, value of count: " . $count, "\n";
104
105 # values are refreshed
106 $val0 = "val0";
107 $count = 0;
108 $otherCount = 0;
109
110 # Break test completed
111 print("Unlabeled Break test completed. \n\n");
112
113 # Values before the test
114 print "Before the test, value of val0: " . $val0, "\n";
115 print "Before the test, value of count: " . $count, "\n";
116
117 # The next statement skips the iteration.
118 # Since this is unlabeled, it worked only in the closest loop.
119 while ($otherCount <= 2) {
120     print("Outer while loop otherCount: $otherCount \n");
121     $count = 0;
122     while ($count <= 3) {
123         $count += 1;
124         if ($count == 2) {
125             print("Continue found, skipped loop at 2 \n");
126             next;
127         }
128         print("val0: $val0 count: $count \n");
129     }
130     $otherCount += 1;
131 }
132
133 # Values after the test
134 print "After the test, value of val0: " . $val0, "\n";
135 print "After the test, value of count: " . $count, "\n";
136
137 # Continue test completed
138 print("Unlabeled continue test completed. \n\n");
139
140 # Labeled Loop Test
141 print("Labeled test while loop \n");
142
143 # values are refreshed
144 $val0 = "val0";
145 $count = 0;
146 $otherCount = 0;

```

```

148 # Values before the test
149 print "Before the test, value of val0: " . $val0, "\n";
150 print "Before the test, value of count: " . $count, "\n";
151
152 # The goto used to send flow of the program to a certain location in the code.
153 outerBreak:
154 while ($otherCount <= 2) {
155     print("Outer while loop otherCount: $otherCount \n");
156     $count = 0;
157     $otherCount += 1;
158     while ($count <= 3) {
159         print("val0: $val0 count: $count \n");
160         $count += 1;
161         if ($count == 2) {
162             print("Goto found, jumped loop at 2 \n");
163             goto outerBreak;
164         }
165     }
166 }
167
168 # Values after the test
169 print "After the test, value of val0: " . $val0, "\n";
170 print "After the test, value of count: " . $count, "\n";
171
172 # Break test completed
173 print("Labeled goto test completed. \n\n");
174
175 # values are refreshed
176 $val0 = "val0";
177 $count = 0;
178 $otherCount = 0;
179
180 # Values before the test
181 print "Before the test, value of val0: " . $val0, "\n";
182 print "Before the test, value of count: " . $count, "\n";
183
184 # The goto used to send flow of the program to a certain location in the code.
185 outerBreak2:
186 while ($otherCount <= 2) {
187     print("Outer while loop otherCount: $otherCount \n");
188     $count = 0;
189     $otherCount += 1;
190     while ($count <= 3) {
191         print("val0: $val0 count: $count \n");
192         $count += 1;
193         if ($count == 2) {
194             print("Next found, skipped loop at 2 \n");
195             next outerBreak2;
196         }
197     }
198 }
199
200 # Values after the test
201 print "After the test, value of val0: " . $val0, "\n";
202 print "After the test, value of count: " . $count, "\n";
203
204 # Break test completed
205 print("Labeled next test completed. \n");
206
207 # Redo, restarts to loop without checking condition.
208 # Last, executes this iteration lastly and EXIT.
209
210 # This statements create an error. Since into loop or switch is disallowed in perl.
211 # goto loop;
212 # for($i=0,$j=50; $i<100; $i++) {
213 #     while($j--) {
214 #         loop:
215 #     }
216 # }

```

Sample Output For Code:

```
$perl main.pl

Pretest with while loop
Before the test, value of val0: val0
Before the test, value of count: 0
val0: val0 count: 0
val0: val0 count: 1
val0: val0 count: 2
After the test, value of val0: val0
After the test, value of count: 3
Pretest completed. The second while loop did not work.
Since the value is checked before execution.

Post-test with do-while loop
Before the test, value of val0: val0
Before the test, value of count: 0
val0: val0 count: 0
val0: val0 count: 1
val0: val0 count: 2
After the test, value of val0: val0
After the test, value of count: 3
val0: val0 count: 3
After the second test, value of val0: val0
After the second test, value of count: 4
Post-test completed. The second while loop ran once.
Since the value is checked after execution.

Unlabeled test while loop
Before the test, value of val0: val0
Before the test, value of count: 0
Outer while loop otherCount: 0
val0: val0 count: 0
val0: val0 count: 1
Break found, exited loop at 2
val0: val0 count: 2
Outer while loop otherCount: 1
val0: val0 count: 0
val0: val0 count: 1
Break found, exited loop at 2
val0: val0 count: 2
After the test, value of val0: val0
After the test, value of count: 3
Unlabeled Break test completed.

Before the test, value of val0: val0
Before the test, value of count: 0
Outer while loop otherCount: 0
val0: val0 count: 1
Continue found, skipped loop at 2
val0: val0 count: 3
Outer while loop otherCount: 1
val0: val0 count: 1
Continue found, skipped loop at 2
val0: val0 count: 3
After the test, value of val0: val0
After the test, value of count: 3
Unlabeled continue test completed.

Labeled test while loop
Before the test, value of val0: val0
Before the test, value of count: 0
Outer while loop otherCount: 0
val0: val0 count: 0
val0: val0 count: 1
Goto found, jumped loop at 2
Outer while loop otherCount: 1
val0: val0 count: 0
val0: val0 count: 1
Goto found, jumped loop at 2
Outer while loop otherCount: 2
val0: val0 count: 0
val0: val0 count: 1
Goto found, jumped loop at 2
After the test, value of val0: val0
After the test, value of count: 2
Labeled goto test completed.

Before the test, value of val0: val0
Before the test, value of count: 0
Outer while loop otherCount: 0
val0: val0 count: 0
val0: val0 count: 1
Next found, skipped loop at 2
Outer while loop otherCount: 1
val0: val0 count: 0
val0: val0 count: 1
Next found, skipped loop at 2
Outer while loop otherCount: 2
val0: val0 count: 0
val0: val0 count: 1
Next found, skipped loop at 2
After the test, value of val0: val0
After the test, value of count: 2
Labeled next test completed.
```

PHP

A while or for loop can be used to create a pretest loop in PHP. A do-while loop can be used to create a post-test loop. There are continue and break keywords that work as unlabeled in PHP. Continue to the next iteration. With break, the loop is terminated. The goto statement directs the program's flow to a certain point in the code.

Full Code Example:

```
1 <?php
2 $val0 = "val0";
3 $count = 0;
4 $otherCount = 0;
5
6 // PRETEST
7 print("Pretest with while loop\n");
8 |
9 // Values before the test
10 echo "Before the test, value of val0: " , $val0, "\n";
11 echo "Before the test, value of count: " , $count, "\n";
12
13 // In pretest, before the instructions within the loop are performed,
14 // the condition required to continue looping is verified
15 while ($count < 3) {
16     echo "val0: " , $val0 , " count: " , $count, "\n";
17     $count += 1;
18 }
19
20 // Values after the test
21 echo "After the test, value of val0: " , $val0, "\n";
22 echo "After the test, value of count: " , $count, "\n";
23
24 // Pretest completed.
25 // The second while loop did not work.
26 // Because the value is checked before execution.
27 while ($count < 3) {
28     echo "val0: " + $val0 + " count: " , $count, "\n";
29     $count += 1;
30 }
31
32 if ( $count <= 3 ) {
33     echo "Pretest completed. The second while loop did not work.", "\n";
34     echo "Since the value is checked before execution. \n\n";
35 }
36
37 // POST-TEST
38 print("Post-test with do-while loop \n");
39
40 // values are refreshed
```

```

41 $val0 = "val0";
42 $count = 0;
43
44 // Values before the test
45 echo "Before the test, value of val0: " , $val0, "\n";
46 echo "Before the test, value of count: " , $count, "\n";
47
48 // The condition is placed at the conclusion of the loop,
49 // and if the condition is true, the loop is terminated with the keyword EXIT.
50 do {
51     echo "val0: " , $val0 , " count: " , $count, "\n";
52     $count += 1;
53 } while ($count < 3);
54
55 // Values after the test
56 echo "After the test, value of val0: " , $val0, "\n";
57 echo "After the test, value of count: " , $count, "\n";
58
59 // The condition is placed at the conclusion of the loop,
60 // and if the condition is true, the loop is terminated with the keyword EXIT.
61 do {
62     echo "val0: " , $val0 , " count: " , $count, "\n";
63     $count += 1;
64 } while ($count < 3);
65
66 echo "After the second test, value of val0: " , $val0, "\n";
67 echo "After the second test, value of count: " , $count, "\n";
68
69 // Post-test completed. The second while loop ran once. Since the value is checked after execution.
70 if ( $count > 3 ) {
71     print("Post-test completed. The second while loop ran once. \n");
72     print("Since the value is checked after execution. \n\n");
73 }
74
75 // Unlabeled Loop Test
76 print("Unlabeled test while loop \n");
77
78 // values are refreshed
79 $val0 = "val0";
80 $count = 0;

```

```

81
82 // Values before the test
83 echo "Before the test, value of val0: " , $val0, "\n";
84 echo "Before the test, value of count: " , $count, "\n";
85
86 // The break statement ends the loop with exit.
87 // Since this is unlabeled, it worked only in the closest loop.
88 while ($otherCount < 2) {
89     print("Outer while loop otherCount: $otherCount \n");
90     $count = 0;
91     while ($count < 3) {
92         print("val0: $val0 count: $count \n");
93         $count += 1;
94         if ( $count == 2 ) {
95             print("Break found, exited loop at 2 \n");
96             break;
97         }
98     }
99     $otherCount += 1;
100 }
101
102 // Values after the test
103 echo "After the test, value of val0: " , $val0, "\n";
104 echo "After the test, value of count: " , $count, "\n";
105
106 // values are refreshed
107 $val0 = "val0";
108 $count = 0;
109 $otherCount = 0;
110
111 // Values before the test
112 echo "Before the test, value of val0: " , $val0, "\n";
113 echo "Before the test, value of count: " , $count, "\n";
114
115 // Break test completed
116 print("Unlabeled Break test completed. \n\n");
117
118 // The continue statement skips the iteration.
119 // Since this is unlabeled, it worked only in the closest loop.
120 while ($otherCount < 2) {

```

```

121     print("Outer while loop otherCount: $otherCount \n");
122     $count = 0;
123     while ($count < 3) {
124         $count += 1;
125         if ( $count == 2 ) {
126             print("Continue found, skipped loop at 2 \n");
127             continue;
128         }
129         print("val0: $val0 count: $count \n");
130     }
131     $otherCount += 1;
132 }
133
134 // Values after the test
135 echo "After the test, value of val0: " , $val0, "\n";
136 echo "After the test, value of count: " , $count, "\n";
137
138 // Continue test completed
139 print("Unlabeled continue test completed. \n\n");
140
141 // Labeled Loop Test
142 print("Labeled test while loop \n");
143
144 // values are refreshed
145 $val0 = "val0";
146 $count = 0;
147 $otherCount = 0;
148
149 // Values before the test
150 echo "Before the test, value of val0: " , $val0, "\n";
151 echo "Before the test, value of count: " , $count, "\n";
152
153 // The goto used to send flow of the program to a certain location in the code.
154 outerBreak:
155 while ($otherCount <= 2) {
156     print("Outer while loop otherCount: $otherCount \n");
157     $count = 0;
158     $otherCount += 1;
159     while ($count < 3) {
160         print("val0: $val0 count: $count \n");

```

```

161         $count += 1;
162         if ( $count == 2 ) {
163             print("Break found, exited loop at 2 \n");
164             goto outerBreak;
165         }
166     }
167 }
168
169 // Values after the test
170 echo "After the test, value of val0: " , $val0, "\n";
171 echo "After the test, value of count: " , $count, "\n";
172
173 // Break test completed
174 print("Labeled goto test completed. \n");
175
176 /*
177 This statements create an error. Since into loop or switch is disallowed in PHP.
178 goto loop;
179 for($i=0,$j=50; $i<100; $i++) {
180     while($j--) {
181         loop:
182     }
183 }
184 */
185
186 ?>

```

Sample Output For Code:

```
Pretest with while loop
Before the test, value of val0: val0
Before the test, value of count: 0
val0: val0 count: 0
val0: val0 count: 1
val0: val0 count: 2
After the test, value of val0: val0
After the test, value of count: 3
Pretest completed. The second while loop did not work.
Since the value is checked before execution.
```

```
Post-test with do-while loop
Before the test, value of val0: val0
Before the test, value of count: 0
val0: val0 count: 0
val0: val0 count: 1
val0: val0 count: 2
After the test, value of val0: val0
After the test, value of count: 3
val0: val0 count: 3
After the second test, value of val0: val0
After the second test, value of count: 4
Post-test completed. The second while loop ran once.
Since the value is checked after execution.
```

```
Unlabeled test while loop
Before the test, value of val0: val0
Before the test, value of count: 0
Outer while loop otherCount: 0
val0: val0 count: 0
val0: val0 count: 1
Break found, exited loop at 2
Outer while loop otherCount: 1
val0: val0 count: 0
val0: val0 count: 1
Break found, exited loop at 2
After the test, value of val0: val0
After the test, value of count: 2
Before the test, value of val0: val0
Before the test, value of count: 0
Unlabeled Break test completed.
```

```
Outer while loop otherCount: 0
val0: val0 count: 1
Continue found, skipped loop at 2
val0: val0 count: 3
Outer while loop otherCount: 1
val0: val0 count: 1
Continue found, skipped loop at 2
val0: val0 count: 3
After the test, value of val0: val0
After the test, value of count: 3
Unlabeled continue test completed.
```

```

Labeled test while loop
Before the test, value of val0: val0
Before the test, value of count: 0
Outer while loop otherCount: 0
val0: val0 count: 0
val0: val0 count: 1
Break found, exited loop at 2
Outer while loop otherCount: 1
val0: val0 count: 0
val0: val0 count: 1
Break found, exited loop at 2
Outer while loop otherCount: 2
val0: val0 count: 0
val0: val0 count: 1
Break found, exited loop at 2
After the test, value of val0: val0
After the test, value of count: 2
Labeled goto test completed.

```

Python

A while or for loop can be used to create a pretest loop in Python. A do-while loop cannot be used to create a post-test loop. There are continue and break keywords that work as unlabeled in Python. Continue to the next iteration. With break, the loop is terminated.

Full Code Example:

```

1 val0 = "val0";
2 count = 0;
3 otherCount = 0;
4 # PRETEST
5 print("Pretest with while loop\n",end = "");
6 # Values before the test
7 print("Before the test, value of val0: ", val0, "\n",end="");
8 print("Before the test, value of count: ", count, "\n",end="");
9 # In pretest, before the instructions within the loop are performed,
10 # the condition required to continue looping is verified
11 while (count < 3) :
12     print("val0: ", val0, " count: ", count, "\n",end="");
13     count += 1;
14
15 # Values after the test
16 print("After the test, value of val0: ", val0, "\n",end="");
17 print("After the test, value of count: ", count, "\n",end="");
18 # Pretest completed.
19 # The second while loop did not work.
20 # Because the value is checked before execution.
21 while (count < 3) :
22     print("val0: " + val0 + " count: ", count, "\n",end="");
23     count += 1;
24
25 if (count <= 3) :
26     print("Pretest completed. The second while loop did not work.", "\n",end="");
27     print("Since the value is checked before execution. \n\n",end="");
28
29 # POST-TEST
30 print("Post-test with do-while loop \n",end = "");
31 # values are refreshed
32 val0 = "val0";
33 count = 0;
34 # Values before the test
35 print("Before the test, value of val0: ", val0, "\n",end="");
36 print("Before the test, value of count: ", count, "\n",end="");
37 # The condition is placed at the conclusion of the loop,
38 # and if the condition is true, the loop is terminated with the keyword EXIT.
39 while(True) :
40     print("val0: ", val0, " count: ", count, "\n",end="");
41     count += 1;
42     if((count < 3) == False) : break;
43
44 # Values after the test
45 print("After the test, value of val0: ", val0, "\n",end="");

```



```

46 print("After the test, value of count: ", count, "\n",end="");
47 # The condition is placed at the conclusion of the loop,
48 # and if the condition is true, the loop is terminated with the keyword EXIT.
49 while(True) :
50     print("val0: ", val0, " count: ", count, "\n",end="");
51     count += 1;
52     if((count < 3) == False) : break;
53
54 print("After the second test, value of val0: ", val0, "\n",end="");
55 print("After the second test, value of count: ", count, "\n",end="");
56 # Post-test completed. The second while loop ran once. Since the value is checked after execution.
57 if (count > 3) :
58     print("Post-test completed. The second while loop ran once. \n",end = "");
59     print("Since the value is checked after execution. \n\n",end = "");
60
61 # Unlabeled Loop Test
62 print("Unlabeled test while loop \n",end = "");
63 # values are refreshed
64 val0 = "val0";
65 count = 0;
66 # Values before the test
67 print("Before the test, value of val0: ", val0, "\n",end="");
68 print("Before the test, value of count: ", count, "\n",end="");
69 # The break statement ends the loop with exit.
70 # Since this is unlabeled, it worked only in the closest loop.
71 while (otherCount < 2) :
72     print("Outer while loop otherCount: " + str(otherCount) + " \n",end = "");
73     count = 0;
74     while (count < 3) :
75         print("val0: " + str(val0) + " count: " + str(count) + " \n",end = "");
76         count += 1;
77         if (count == 2) :
78             print("Break found, exited loop at 2 \n",end = "");
79             break;
80
81     otherCount += 1;
82
83 # Values after the test
84 print("After the test, value of val0: ", val0, "\n",end="");
85 print("After the test, value of count: ", count, "\n",end="");
86 # values are refreshed
87 val0 = "val0";
88 count = 0;
89 otherCount = 0;

```

```

91 # Values before the test
92 print("Before the test, value of val0: ", val0, "\n",end="");
93 print("Before the test, value of count: ", count, "\n",end="");
94 # Break test completed
95 print("Unlabeled Break test completed. \n\n",end = "");
96 # The continue statement skips the iteration.
97 # Since this is unlabeled, it worked only in the closest loop.
98 while (otherCount < 2) :
99     print("Outer while loop otherCount: " + str(otherCount) + " \n",end = "");
100     count = 0;
101     while (count < 3) :
102         count += 1;
103         if (count == 2) :
104             print("Continue found, skipped loop at 2 \n",end = "");
105             continue;
106
107         print("val0: " + str(val0) + " count: " + str(count) + " \n",end = "");
108
109     otherCount += 1;
110
111 # Values after the test
112 print("After the test, value of val0: ", val0, "\n",end="");
113 print("After the test, value of count: ", count, "\n",end="");
114 # Continue test completed
115 print("Unlabeled continue test completed. \n\n",end = "");

```

Sample Output For Code:

```
Pretest with while loop
Before the test, value of val0: val0
Before the test, value of count: 0
val0: val0 count: 0
val0: val0 count: 1
val0: val0 count: 2
After the test, value of val0: val0
After the test, value of count: 3
Pretest completed. The second while loop did not work.
Since the value is checked before execution.

Post-test with do-while loop
Before the test, value of val0: val0
Before the test, value of count: 0
val0: val0 count: 0
val0: val0 count: 1
val0: val0 count: 2
After the test, value of val0: val0
After the test, value of count: 3
val0: val0 count: 3
After the second test, value of val0: val0
After the second test, value of count: 4
Post-test completed. The second while loop ran once.
Since the value is checked after execution.

Unlabeled test while loop
Before the test, value of val0: val0
Before the test, value of count: 0
Outer while loop otherCount: 0
val0: val0 count: 0
val0: val0 count: 1
Break found, exited loop at 2
Outer while loop otherCount: 1
val0: val0 count: 0
val0: val0 count: 1
Break found, exited loop at 2
After the test, value of val0: val0
After the test, value of count: 2
Before the test, value of val0: val0
Before the test, value of count: 0
Unlabeled Break test completed.

Outer while loop otherCount: 0
val0: val0 count: 1
Continue found, skipped loop at 2
val0: val0 count: 3
Outer while loop otherCount: 1
val0: val0 count: 1
Continue found, skipped loop at 2
val0: val0 count: 3
After the test, value of val0: val0
After the test, value of count: 3
Unlabeled continue test completed.
```

Evaluation of These Languages in Terms of Readability and Writability of Logically-Controlled Loops

When the logical loops of these languages are examined in terms of readability and writability, I think that the most diversity is in Perl. Since it has more control functions than other languages and their ease of writing. In Perl, functions are named differently than in other languages. Although this may seem like confusion, there is no problem in terms of readability as there are quite meaningful keywords. The next command is used instead of continue and the last command is used instead of the break to call functions. Perl proved to be the most successful language for me in this regard, both in terms of readability and understanding when creating code. In particular, Perl comes first because it contains the label function, loop blocks such as while and do-while, as in other languages, and does not have a

deficiency. The scope of the Python language on this topic has been narrower than I expected. I just don't have access to a simple loop block like do-while, which allows for post-testing. Also, I cannot show the procedure to go to Label using this language. These situations limit Python's writability. Some PHP functions are also accessible in other programming languages. On the other hand, the goto function cannot be used as a labelled unconditional function. A created tag cannot be accessed from within the function. This reduces the potential writability. The structure of the Dart language is different from other languages. This language allows looping within the main function. Various loop blocks can be used to create test classes. Despite its efficiency, I don't think it's as useful as Perl. Finally, the same things are available in Javascript. Continue and Break routines can be used to give loop control. Pretest and posttest methods can be used to create and test loop blocks. This language allows for do-while content to be included.

My Learning Strategy

I conducted some preliminary study on the topic on the slides and the internet before beginning Assignment 2. I learned that different structures and keywords in different languages may be employed in loop control as a result of these subjects. The pretest and posttest methods can also be used to test the content of the loop blocks. I began writing similar codes after properly learning this material. Finding a comparable code in these 5 various languages mentioned in the task was a bit tricky for me. The most significant source of this flaw is the Python programming language. Since I began writing Python code, I've observed these flaws in the languages I've written.

Later, I discovered that this functionality does not exist in PHP or that it cannot be achieved with a single command. Using an online compiler, I addressed a single programming problem for five distinct languages, which was a challenging topic in my first project. I was able to rapidly transform the codes I generated to output using online compilers. As a result, I was able to more readily intervene in coding issues.

My task has not been significantly cut due to online compilers. Since the online compiler runs from the browser, the browser froze when I tried to write and run long tests. This caused both the fear of losing the code that I did not save and a waste of time.

While writing the scripts, I made several errors in the information I gathered. Some of the languages mentioned in the book's content and examples are not covered in the scope of this assignment. As a result, I conducted internet research to find out more about the programming languages mentioned in the assignment's content. As a consequence of my investigation, I discovered a lot of inconsistent data. As a result, I ran into several complications. Labelling in PHP language, for example, may be retrieved from within the loop, according to one source. When I tried to create the code and print it out, I ran into several issues.

This project improved my understanding of programming languages. One of the most significant aspects of completing this project is understanding the topic. Because there is so little knowledge about the Dart language, I was able to have restricted access to specific patterns and instructions. I believe that I should be able to get additional knowledge about a language that I do not fully understand.

In this assignment, I used online compilers directly. I can list them as follows: DartPad (<https://dartpad.dev/?>), Tutorialspoint Online Dart Compiler (https://www.tutorialspoint.com/execute_dart_online.php), Programiz Online JavaScript (<https://www.programiz.com/javascript/online-compiler/>) and Python Compiler (<https://www.programiz.com/python-programming/online-compiler/>), Tutorialspoint Online Perl Compiler (https://www.tutorialspoint.com/execute_perl_online.php) and Paize Online PHP Editor (<https://paiza.io/en/projects/new>). I also used the resources of these websites on the subject (<https://explorable.com/pretest-posttest-designs>, <https://www.geeksforgeeks.org/dart-loop-control-statements-break-and-continue/>, <https://www.javatpoint.com/javascript-label-statement>).