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:

```
void main() {
  String val0 = "val0";
  int count = 0;
  int otherCount = 0;

  // PRETEST
  print("Pretest with while loop");

  // Values before the test
  print("Before the test, value of val0: $val0");
  print("Before the test, value of count: $count");

  // In pretest, before the instructions within the loop are performed,
  // the condition required to continue looping is verified
  while (count < 3) {
    print("val0: $val0, count: $count");
    count += 1;
    val0 = "val$count";
  }

  // Values after the test
  print("After the test, value of val0: $val0");
  print("After the test, value of count: $count");

  // Pretest completed.
  // The second while loop did not work.
  // Because the value is checked before execution.
  while (count < 3) {
    print("val0: $val0, count: $count");
    count += 1;
    val0 = "val$count";
  }

  if ( count <= 3 ) {
    print("Pretest completed. The second while loop did not work.");
    print("Since the value is checked before execution. \n");
  }

  // POST-TEST
  print("Post-test with do-while loop");

  // values are refreshed
  val0 = "val0";
  count = 0;

  // Values before the test
  print("Before the test, value of val0: $val0");
  print("Before the test, value of count: $count");

  // The condition is placed at the conclusion of the loop,
  // and if the condition is true, the loop is terminated with the keyword EXIT.
  do {
    print("val0: $val0, count: $count");
    count += 1;
    val0 = "val$count";
  } while (count < 3);

  // Values after the test
  print("After the test, value of val0: $val0");
  print("After the test, value of count: $count");

  // The condition is placed at the conclusion of the loop,
  // and if the condition is true, the loop is terminated with the keyword EXIT.
  do {
    print("val0: $val0, count: $count");
    count += 1;
    val0 = "val$count";
  } while (count < 3);

  print("After the second test, value of val0: $val0");
  print("After the second test, value of count: $count");

  // Post-test completed. The second while loop ran once. Since the value is checked after exec
  if ( count > 3 ) {
    print("Post-test completed. The second while loop ran once.");
    print("Since the value is checked after execution. \n");
  }

  // Unlabeled Loop Test
  print("Unlabeled test while loop");

  // values are refreshed
  val0 = "val0";
  count = 0;

  // Values before the test
  print("Before the test, value of val0: $val0");
  print("Before the test, value of count: $count");

  // The break statement ends the loop with exit.
  // Since this is unlabeled, it worked only in the closest loop.
  while (otherCount < 2) {
    print("Outer while loop otherCount:$otherCount");
    count = 0;
    while (count < 3) {
      print("val0: $val0, count: $count");
      count += 1;
      val0 = "val$count";
      if ( count == 2 ) {
        print("Break found, exited loop at 2");
        break;
      }
    }
    otherCount += 1;
  }

  // Values after the test
  print("After the test, value of val0: $val0");
  print("After the test, value of count: $count");

  // values are refreshed
  val0 = "val0";
  count = 0;
  otherCount = 0;
```

```
    // Values before the test
    print("Before the test, value of val0: $val0");
    print("Before the test, value of count: $count");

    // Break test completed
    print("Unlabeled Break test completed. \n");

    // The continue statement skips the iteration.
    // Since this is unlabeled, it worked only in the closest loop.
    while (otherCount < 2) {
      print("Outer while loop otherCount:$otherCount");
      count = 0;
      while (count < 3) {
        count += 1;
        val0 = "val$count";
        if ( count == 2 ) {
          print("Continue found, skipped loop at 2");
          continue;
        }
        print("val0: $val0, count: $count");
      }
      otherCount += 1;
    }

    // Values after the test
    print("After the test, value of val0: $val0");
    print("After the test, value of count: $count");

    // Continue test completed
    print("Unlabeled continue test completed. \n");

    // Labeled Loop Test
    print("Labeled test while loop");

    // values are refreshed
    val0 = "val0";
    count = 0;
    otherCount = 0;
```

```
    // Values before the test
    print("Before the test, value of val0: $val0");
    print("Before the test, value of count: $count");

    // The break statement ends the loop with exit.
    // Since this is labeled, it worked in the specified label's loop.
    outerBreak:
    while (otherCount <= 2) {
      print("Outer while loop otherCount:$otherCount");
      count = 0;
      otherCount += 1;
      while (count < 3) {
        print("val0: $val0, count: $count");
        count += 1;
        val0 = "val$count";
        if ( count == 2 ) {
          print("Break found, exited loop at 2");
          break outerBreak;
        }
      }
    }

    // Values after the test
    print("After the test, value of val0: $val0");
    print("After the test, value of count: $count");

    // values are refreshed
    val0 = "val0";
    count = 0;
    otherCount = 0;

    // Values before the test
    print("Before the test, value of val0: $val0");
    print("Before the test, value of count: $count");

    // Break test completed
    print("Labeled Break test completed. \n");

```

```
    // The continue statement skips the iteration.
    // Since this is labeled, it worked in the specified label's loop.
    outerContinue:
    while (otherCount <= 2) {
      print("Outer while loop otherCount:$otherCount");
      count = 0;
      otherCount += 1;
      while (count < 3) {
        count += 1;
        val0 = "val$count";
        if ( count == 2 ) {
          print("Continue found, skipped loop at 2");
          continue outerContinue;
        }
        print("val0: $val0, count: $count");
      }
    }

    // Values after the test
    print("After the test, value of val0: $val0");
    print("After the test, value of count: $count");

    // Continue test completed
    print("Labeled continue test completed. \n");
}
```

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

Run

```
var val0 = "val0";
var count = 0;
var otherCount = 0;

// PRETEST
console.log("Pretest with while loop");

// Values before the test
console.log("Before the test, value of val0: " + val0);
console.log("Before the test, value of count: " + count);

// In pretest, before the instructions within the loop are performed,
// the condition required to continue looping is verified
while (count < 3) {
    console.log("val0: " + val0 + " count: " + count);
    count += 1;
    val0 = "val" + count;
}

// Values after the test
console.log("After the test, value of val0: " + val0);
console.log("After the test, value of count: " + count);

// Pretest completed.
// The second while loop did not work.
// Because the value is checked before execution.
while (count < 3) {
    console.log("val0: " + val0 + " count: " + count);
    count += 1;
    val0 = "val" + count;
}

if ( count <= 3 ) {
    console.log("Pretest completed. The second while loop did not work.");
    console.log("Since the value is checked before execution. \n");
}

// POST-TEST
console.log("Post-test with do-while loop");

// values are refreshed
val0 = "val0";
count = 0;

// Values before the test
console.log("Before the test, value of val0: " + val0);
console.log("Before the test, value of count: " + count);

// The condition is placed at the conclusion of the loop,
// and if the condition is true, the loop is terminated with the keyword EXIT.
do {
    console.log("val0: " + val0 + " count: " + count);
    count += 1;
    val0 = "val" + count;
} while (count < 3);

// Values after the test
console.log("After the test, value of val0: " + val0);
console.log("After the test, value of count: " + count);

// The condition is placed at the conclusion of the loop,
// and if the condition is true, the loop is terminated with the keyword EXIT.
do {
    console.log("val0: " + val0 + " count: " + count);
    count += 1;
    val0 = "val" + count;
} while (count < 3);

console.log("After the second test, value of val0: " + val0);
console.log("After the second test, value of count: " + count);

// Post-test completed. The second while loop ran once. Since the value is checked after execution.
if ( count > 3 ) {
    console.log("Post-test completed. The second while loop ran once.");
    console.log("Since the value is checked after execution. \n");
}

// Unlabeled Loop Test
console.log("Unlabeled test while loop");

// values are refreshed
val0 = "val0";
count = 0;

// Values before the test
console.log("Before the test, value of val0: " + val0);
console.log("Before the test, value of count: " + count);

// The break statement ends the loop with exit.
// Since this is unlabeled, it worked only in the closest loop.
while (otherCount < 2) {
    console.log("Outer while loop otherCount: " + otherCount);
    count = 0;
    while (count < 3) {
      console.log("val0: " + val0 + " count: " + count);
      count += 1;
```

```
      val0 = "val" + count;
      if ( count == 2 ) {
        console.log("Break found, exited loop at 2");
        break;
      }
    }
    otherCount += 1;
}

// Values after the test
console.log("After the test, value of val0: " + val0);
console.log("After the test, value of count: " + count);

// values are refreshed
val0 = "val0";
count = 0;
otherCount = 0;

// Values before the test
console.log("Before the test, value of val0: " + val0);
console.log("Before the test, value of count: " + count);

// Break test completed
console.log("Unlabeled Break test completed. \n");

// The continue statement skips the iteration.
// Since this is unlabeled, it worked only in the closest loop.
while (otherCount < 2) {
    console.log("Outer while loop otherCount: " + otherCount);
    count = 0;
    while (count < 3) {
      count += 1;
      val0 = "val" + count;
      if ( count == 2 ) {
        console.log("Continue found, skipped loop at 2");
        continue;
      }
      console.log("val0: " + val0 + " count: " + count);
    }
    otherCount += 1;
}

// Values after the test
console.log("After the test, value of val0: " + val0);
console.log("After the test, value of count: " + count);

// Continue test completed
console.log("Unlabeled continue test completed. \n");
```

```
// Labeled Loop Test
console.log("Labeled test while loop");

// values are refreshed
val0 = "val0";
count = 0;
otherCount = 0;

// Values before the test
console.log("Before the test, value of val0: " + val0);
console.log("Before the test, value of count: " + count);

// The break statement ends the loop with exit.
// Since this is labeled, it worked in the specified label's loop.
outerBreak:
while (otherCount <= 2) {
    console.log("Outer while loop otherCount: " + otherCount);
    count = 0;
    otherCount += 1;
    while (count < 3) {
      console.log("val0: " + val0 + " count: " + count);
      count += 1;
      val0 = "val" + count;
      if ( count == 2 ) {
        console.log("Break found, exited loop at 2");
        break outerBreak;
      }
    }
}

// Values after the test
console.log("After the test, value of val0: " + val0);
console.log("After the test, value of count: " + count);

// Break test completed
console.log("Labeled Break test completed. \n");

// values are refreshed
val0 = "val0";
count = 0;
otherCount = 0;

// Values before the test
console.log("Before the test, value of val0: " + val0);
console.log("Before the test, value of count: " + count);
```

```
// The continue statement skips the iteration.
// Since this is labeled, it worked in the specified label's loop.
outerContinue:
while (otherCount <= 2) {
    console.log("Outer while loop otherCount: " + otherCount);
    count = 0;
    otherCount += 1;
    while (count < 3) {
      count += 1;
      val0 = "val" + count;
      if ( count == 2 ) {
        console.log("Continue found, skipped loop at 2");
        continue outerContinue;
      }
      console.log("val0: " + val0 + " count: " + count);
    }
}

// Values after the test
console.log("After the test, value of val0: " + val0);
console.log("After the test, value of count: " + count);

// Continue test completed
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.
```

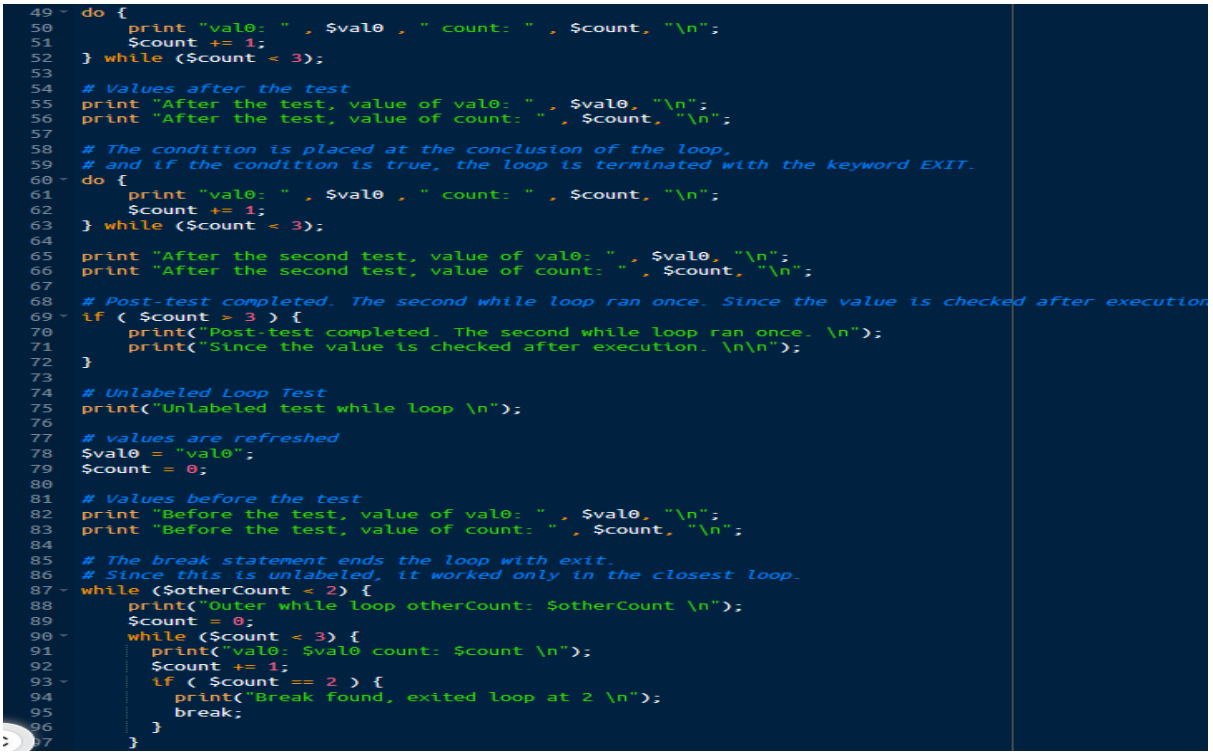
```
Outer while loop otherCount: 0
val0: val1 count: 1
Continue found, skipped loop at 2
val0: val3 count: 3
Outer while loop otherCount: 1
val0: val1 count: 1
Continue found, skipped loop at 2
val0: val3 count: 3
After the test, value of val0: val3
After the test, value of count: 3
Unlabeled continue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

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:

```
$val0 = "val0";
$count = 0;
$otherCount = 0;

# PRETEST
print("Pretest with while loop\n");

# Values before the test
print "Before the test, value of val0: " , $val0, "\n";
print "Before the test, value of count: " , $count, "\n";

# In pretest, before the instructions within the loop are performed,
# the condition required to continue looping is verified
while ($count < 3) {
    print "val0: " , $val0 , " count: " , $count, "\n";
    $count += 1;
}

# Values after the test
print "After the test, value of val0: " , $val0, "\n";
print "After the test, value of count: " , $count, "\n";

# Pretest completed.
# The second while loop did not work.
# Because the value is checked before execution.
while ($count < 3) {
    print "val0: " + $val0 + " count: " , $count, "\n";
    $count += 1;
}

if ( $count <= 3 ) {
    print "Pretest completed. The second while loop did not work.", "\n";
    print "Since the value is checked before execution. \n\n";
}

# POST-TEST
print("Post-test with do-while loop \n");

# values are refreshed
$val0 = "val0";
$count = 0;

# Values before the test
print "Before the test, value of val0: " , $val0, "\n";
print "Before the test, value of count: " , $count, "\n";

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



```
do {
    print "val0: " , $val0 , " count: " , $count, "\n";
    $count += 1;
} while ($count < 3);

# Values after the test
print "After the test, value of val0: " , $val0, "\n";
print "After the test, value of count: " , $count, "\n";

# The condition is placed at the conclusion of the loop,
# and if the condition is true, the loop is terminated with the keyword EXIT.
do {
    print "val0: " , $val0 , " count: " , $count, "\n";
    $count += 1;
} while ($count < 3);

print "After the second test, value of val0: " , $val0, "\n";
print "After the second test, value of count: " , $count, "\n";

# Post-test completed. The second while loop ran once. Since the value is checked after execution
if ( $count > 3 ) {
    print("Post-test completed. The second while loop ran once. \n");
    print("Since the value is checked after execution. \n\n");
}

# Unlabeled Loop Test
print("Unlabeled test while loop \n");

# values are refreshed
$val0 = "val0";
$count = 0;

# Values before the test
print "Before the test, value of val0: " , $val0, "\n";
print "Before the test, value of count: " , $count, "\n";

# The break statement ends the loop with exit.
# Since this is unlabeled, it worked only in the closest loop.
while ($otherCount < 2) {
    print("Outer while loop otherCount: $otherCount \n");
    $count = 0;
    while ($count < 3) {
      print("val0: $val0 count: $count \n");
      $count += 1;
      if ( $count == 2 ) {
        print("Break found, exited loop at 2 \n");
        break;
      }
    }
```

```
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:

```php
<?php
$val0 = "val0";
$count = 0;
$otherCount = 0;

// PRETEST
print("Pretest with while loop\n");

// Values before the test
echo "Before the test, value of val0: " , $val0, "\n";
echo "Before the test, value of count: " , $count, "\n";

// In pretest, before the instructions within the loop are performed,
// the condition required to continue looping is verified
while ($count < 3) {
    echo "val0: " , $val0 , " count: " , $count, "\n";
    $count += 1;
}

// Values after the test
echo "After the test, value of val0: " , $val0, "\n";
echo "After the test, value of count: " , $count, "\n";

// Pretest completed.
// The second while loop did not work.
// Because the value is checked before execution.
while ($count < 3) {
    echo "val0: " + $val0 + " count: " , $count, "\n";
    $count += 1;
}

if ( $count <= 3 ) {
    echo "Pretest completed. The second while loop did not work.", "\n";
    echo "Since the value is checked before execution. \n\n";
}

// POST-TEST
print("Post-test with do-while loop \n");

// values are refreshed
```

```php
$val0 = "val0";
$count = 0;

// Values before the test
echo "Before the test, value of val0: " , $val0, "\n";
echo "Before the test, value of count: " , $count, "\n";

// The condition is placed at the conclusion of the loop,
// and if the condition is true, the loop is terminated with the keyword EXIT.
do {
    echo "val0: " , $val0 , " count: " , $count, "\n";
    $count += 1;
} while ($count < 3);

// Values after the test
echo "After the test, value of val0: " , $val0, "\n";
echo "After the test, value of count: " , $count, "\n";

// The condition is placed at the conclusion of the loop,
// and if the condition is true, the loop is terminated with the keyword EXIT.
do {
    echo "val0: " , $val0 , " count: " , $count, "\n";
    $count += 1;
} while ($count < 3);

echo "After the second test, value of val0: " , $val0, "\n";
echo "After the second test, value of count: " , $count, "\n";

// Post-test completed. The second while loop ran once. Since the value is checked after execution.
if ( $count > 3 ) {
    print("Post-test completed. The second while loop ran once. \n");
    print("Since the value is checked after execution. \n\n");
}

// Unlabeled Loop Test
print("Unlabeled test while loop \n");

// values are refreshed
$val0 = "val0";
$count = 0;
```

```php

// Values before the test
echo "Before the test, value of val0: " , $val0, "\n";
echo "Before the test, value of count: " , $count, "\n";

// The break statement ends the loop with exit.
// Since this is unlabeled, it worked only in the closest loop.
while ($otherCount < 2) {
    print("Outer while loop otherCount: $otherCount \n");
    $count = 0;
    while ($count < 3) {
      print("val0: $val0 count: $count \n");
      $count += 1;
      if ( $count == 2 ) {
        print("Break found, exited loop at 2 \n");
        break;
      }
    }
    $otherCount += 1;
}

// Values after the test
echo "After the test, value of val0: " , $val0, "\n";
echo "After the test, value of count: " , $count, "\n";

// values are refreshed
$val0 = "val0";
$count = 0;
$otherCount = 0;

// Values before the test
echo "Before the test, value of val0: " , $val0, "\n";
echo "Before the test, value of count: " , $count, "\n";

// Break test completed
print("Unlabeled Break test completed. \n\n");

// The continue statement skips the iteration.
// Since this is unlabeled, it worked only in the closest loop.
while ($otherCount < 2) {
```

```
    print("Outer while loop otherCount: $otherCount \n");
    $count = 0;
    while ($count < 3) {
      $count += 1;
      if ( $count == 2 ) {
        print("Continue found, skipped loop at 2 \n");
        continue;
      }
      print("val0: $val0 count: $count \n");
    }
    $otherCount += 1;
}

// Values after the test
echo "After the test, value of val0: " , $val0, "\n";
echo "After the test, value of count: " , $count, "\n";

// Continue test completed
print("Unlabeled continue test completed. \n\n");

// Labeled Loop Test
print("Labeled test while loop \n");

// values are refreshed
$val0 = "val0";
$count = 0;
$otherCount = 0;

// Values before the test
echo "Before the test, value of val0: " , $val0, "\n";
echo "Before the test, value of count: " , $count, "\n";

// The goto used to send flow of the program to a certain location in the code.
outerBreak:
while ($otherCount <= 2) {
    print("Outer while loop otherCount: $otherCount \n");
    $count = 0;
    $otherCount += 1;
    while ($count < 3) {
      print("val0: $val0 count: $count \n");
```

```
      $count += 1;
      if ( $count == 2 ) {
        print("Break found, exited loop at 2 \n");
        goto outerBreak;
      }
    }
}

// Values after the test
echo "After the test, value of val0: " , $val0, "\n";
echo "After the test, value of count: " , $count, "\n";

// Break test completed
print("Labeled goto test completed. \n");

/*
This statements create an error. Since into loop or switch is disallowed in PHP.
goto loop;
for($i=0,$j=50; $i<100; $i++) {
  while($j--) {
    loop:
  }
}
*/

?>
```

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.

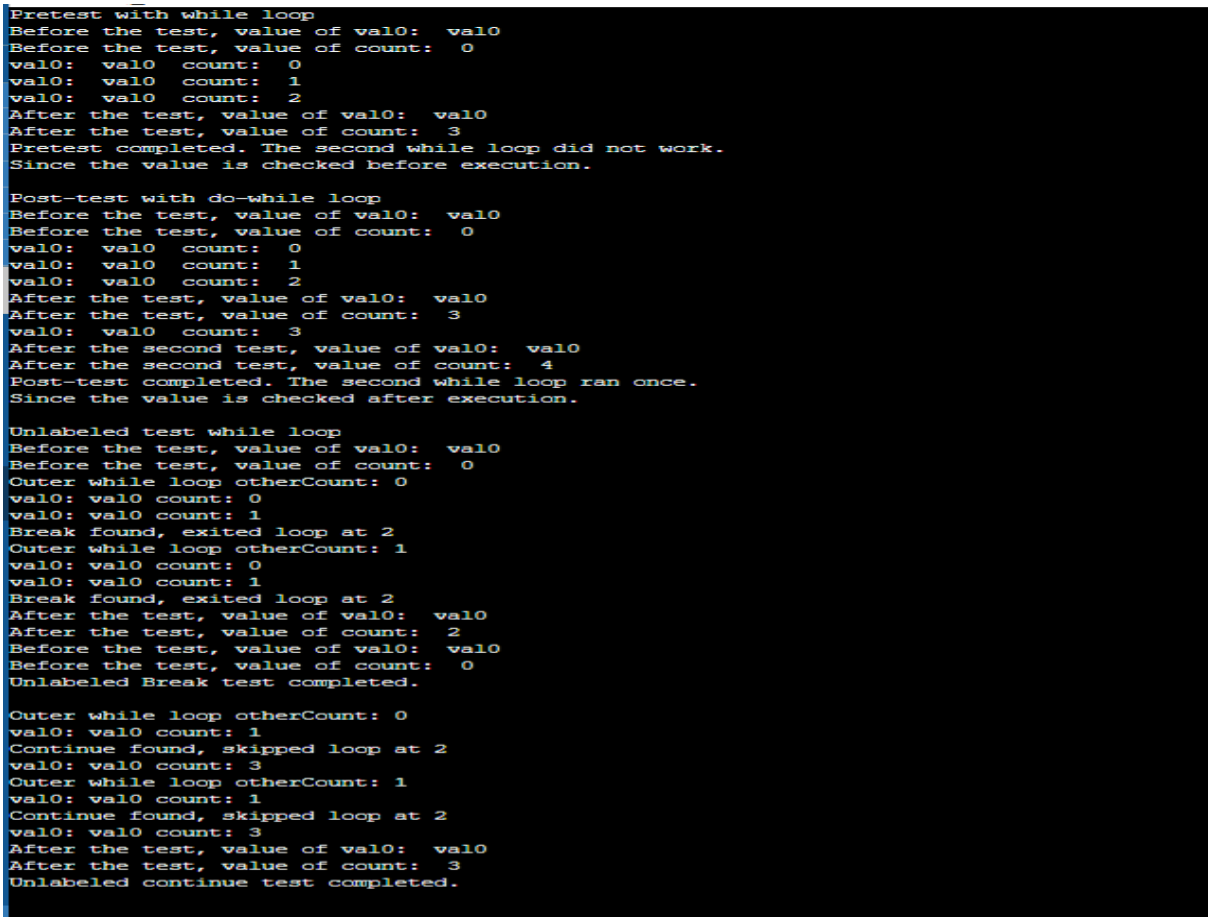
Full Code Example:

```
val0 = "val0";
count = 0;
otherCount = 0;
# PRETEST
print("Pretest with while loop\n",end = "");
# Values before the test
print("Before the test, value of val0: ", val0, "\n",end="");
print("Before the test, value of count: ", count, "\n",end="");
# In pretest, before the instructions within the loop are performed,
# the condition required to continue looping is verified
while (count < 3) :
    print("val0: ", val0, " count: ", count, "\n",end="");
    count += 1;

# Values after the test
print("After the test, value of val0: ", val0, "\n",end="");
print("After the test, value of count: ", count, "\n",end="");
# Pretest completed.
# The second while loop did not work.
# Because the value is checked before execution.
while (count < 3) :
    print("val0: " + val0 + " count: ", count, "\n",end="");
    count += 1;

if (count <= 3) :
    print("Pretest completed. The second while loop did not work.", "\n",end="");
    print("Since the value is checked before execution. \n\n",end="");

# POST-TEST
print("Post-test with do-while loop \n",end = "");
# values are refreshed
val0 = "val0";
count = 0;
# Values before the test
print("Before the test, value of val0: ", val0, "\n",end="");
print("Before the test, value of count: ", count, "\n",end="");
# The condition is placed at the conclusion of the loop,
# and if the condition is true, the loop is terminated with the keyword EXIT.
while(True) :
    print("val0: ", val0, " count: ", count, "\n",end="");
    count += 1;
    if((count < 3) == False) : break;

# Values after the test
print("After the test, value of val0: ", val0, "\n",end="");
```

```
print("After the test, value of count: ", count, "\n",end="");
# The condition is placed at the conclusion of the loop,
# and if the condition is true, the loop is terminated with the keyword EXIT.
while(True) :
    print("val0: ", val0, " count: ", count, "\n",end="");
    count += 1;
    if((count < 3) == False) : break;

print("After the second test, value of val0: ", val0, "\n",end="");
print("After the second test, value of count: ", count, "\n",end="");
# Post-test completed. The second while loop ran once. Since the value is checked after execution.
if (count > 3) :
    print("Post-test completed. The second while loop ran once. \n",end = "");
    print("Since the value is checked after execution. \n\n",end = "");

# Unlabeled Loop Test
print("Unlabeled test while loop \n",end = "");
# values are refreshed
val0 = "val0";
count = 0;
# Values before the test
print("Before the test, value of val0: ", val0, "\n",end="");
print("Before the test, value of count: ", count, "\n",end="");
# The break statement ends the loop with exit.
# Since this is unlabeled, it worked only in the closest loop.
while (otherCount < 2) :
    print("Outer while loop otherCount: " + str(otherCount) + " \n",end = "");
    count = 0;
    while (count < 3) :
        print("val0: " + str(val0) + " count: " + str(count) + " \n",end = "");
        count += 1;
        if (count == 2) :
            print("Break found, exited loop at 2 \n",end = "");
            break;


    otherCount += 1;

# Values after the test
print("After the test, value of val0: ", val0, "\n",end="");
print("After the test, value of count: ", count, "\n",end="");
# values are refreshed
val0 = "val0";
count = 0;
otherCount = 0;
```

```
# Values before the test
print("Before the test, value of val0: ", val0, "\n",end="");
print("Before the test, value of count: ", count, "\n",end="");
# Break test completed
print("Unlabeled Break test completed. \n\n",end = "");
# The continue statement skips the iteration.
# Since this is unlabeled, it worked only in the closest loop.
while (otherCount < 2) :
    print("Outer while loop otherCount: " + str(otherCount) + " \n",end = "");
    count = 0;
    while (count < 3) :
        count += 1;
        if (count == 2) :
            print("Continue found, skipped loop at 2 \n",end = "");
            continue;

        print("val0: " + str(val0) + " count: " + str(count) + " \n",end = "");

    otherCount += 1;

# Values after the test
print("After the test, value of val0: ", val0, "\n",end="");
print("After the test, value of count: ", count, "\n",end="");
# Continue test completed
print("Unlabeled continue test completed. \n\n",end = "");
```

Sample Output For Code:



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).