**BILKENT UNIVERSITY**

**ENGINEERING FACULTY**

**DEPARTMENT OF COMPUTER ENGINEERING**


# CS315

# Homework Assignment 2

# Logically Controlled Loops in Dart, Javascript, Lua, PHP, Python, Ruby, and Rust

**Bulut Gözübüyük 21702771**

# 1.0 Dart

## 1.1 Pretest

Pretest in Dart can be done using the while loop.

### 1.1.1 Code

```
var i = 0;
while (i < 5) {
  print(i);
  i++;
}
```

### 1.1.2 Output

```
0
1
2
3
4
```

## 1.2 Posttest

Posttest in Dart can be done using do while loop.

### 1.2.1 Code

```
i = 0;
do {
  print(i);
  i++;
} while (i < 5);
```

### 1.2.2 Output

```
0
1
2
3
4
```

## 1.3 Unlabeled break

There is an unlabeled break in Dart to terminate loops.

### 1.3.1 Code

```
i = 0;
  while (i < 5) {
    i++;
    print(i);
    break;
  }
```

### 1.3.2 Output

```
1
```

## 1.4 Unlabeled Continue

In Dart, continue can be used to skip over the current iteration in the loop. In the below code, loop skips printing 2 using the continue.

### 1.4.1 Code

```
 i = 0;
  while (i < 5) {
    i++;
    if (i == 2) { continue; }
    print(i);
  }
```

### 1.4.2 Output

```
1
3
4
5
```

## 1.5 Labeled Break

In Dart, break can be done using labels as can be seen in the example below. The break statement terminated the outer loop.

### 1.5.1 Code

```
i = 0;
 label:
 while (true) {
  while (true) {
   print(i);
   if (i == 1) break label;
   i++;
  }
 }
```

### 1.5.2 Output

```
0
1
```

## 1.6 Labeled Continue

In Dart, continue statements can be used with labels as can be seen in the example below. The continue statement skips the current iteration of outer loop.

### 1.6.1 Code

```
 i = 0;
 var j = 0;
 anotherlabel:
 while (i < 2) {
  while (j < 2) {
   print("$i,$j");
   j++;
   if (i == j) continue anotherlabel;
  }
  i++;
  print(i);
 }
```

### 1.6.2 Output

```
0,0
0,1
1
2
```

### 1.7 Evaluation of Dart programming language

Dart offers strong readability since curly braces {} are used to evaluate scopes. In addition, because of the large number of English identifiers, one can read Dart quickly. I have found that it mainly uses English keywords to handle them while programming logically controlled loops using the Dart programming language. Plus it has labels for continue and break. Therefore those properties make Dart a language that smooths the way the logically controlled loops are programmed.

# 2.0 Javascript

## 2.1 Pretest

Pretest in Javascript can be done using a while loop. Test condition is written in parentheses after the while statement.

### 2.1.1 Code

```
var i = 0;

while (i < 3) {
   console.log(i);
   i++;
}
```

### 2.1.2 Output

```
0
1
2
```

## 2.2 Posttest

Using do while loop, posttest can be done in Javascript.

### 2.2.1 Code

```
i = 0

do {
   console.log(i);
   i++;
} while (i < 3);
```

### 2.2.2 Output

```
0
1
2
```

## 2.3 Unlabeled Break

Break can be used in Javascript to terminate loops. 2 cant be printed since loop is terminated.

### 2.3.1 Code

```
var i = 0;

while (i < 3) {
   if (i == 2 ) break;
   console.log(i);
   i++;
}
```

### 2.3.2 Output

0
1

## 2.4 Unlabeled Continue

Continue in Javascript is used to skip current iteration in the loop. Printing 2 is skipped with a continue statement.

### 2.4.1 Code

```
var i = 0;

while (i < 3) {
   i++;
   if (i == 2 ) continue;
   console.log(i);
}
```

### 2.4.2 Output

1
3

## 2.5 Labeled Break

In Javascript, break can be done using labels as can be seen in the example below. The break statement terminated the outer loop.

### 2.5.1 Code

```
i = 0;
label:
while (true) {
   while (true) {
     console.log(i);
     if (i == 1) break label;
     i++;
   }
}
```

### 2.5.2 Output

0
1

## 2.6 Labeled Continue

In Javascript, continue statements can be used with labels as can be seen in the example below. The continue statement skips the current iteration of outer loop.

### 2.6.1 Code

```
i = 0;
var j = 0;
anotherlabel:
while (i < 2) {
   while (j < 2) {
     console.log(i + ", ", j);
     j++;
     if (i == j) continue anotherlabel;
   }
i++;
console.log(i);
}
```

### 2.6.2 Output

0, 0
0, 1
1
2

## 2.7 Evaluation of Javascript programming language

Javascript uses curly braces ({}) to evaluate scope, as in the Dart language. This makes it easier to read Javascript. In addition, the readability of the Javascript programming language is much higher because it is similar enough to the English language. Javascript uses a considerable number of English identifiers to handle loops, labeled continue and break statements, as in the Dart programming language. Therefore it can be inferred that the programming language of Javascript encourages the work of programmers with loops.

# 3.0 Lua

## 3.1 Pretest

Pretest condition in Lua can be done using while (condition) do (stmt) end.

### 3.1.1 Code

```
i = 0
while (i < 3)
do
   print(i)
   i = 1 + i
end
```

### 3.1.2 Output

```
0
1
2
```

## 3.2 Posttest

Posttest in Lua can be done using repeat and until statements.

### 3.2.1 Code

```
i = 0
repeat
   print(i)
   i = 1 + i
until (i > 2)
```

### 3.2.2 Output

```
0
1
2
```

## 3.3 Unlabeled Break

Ints.

### 3.3.1 Code

```
i = 0
while (i < 3)
do
   if (i == 2) then break end
   print(i)
   i = 1 + i
end
```

### 3.3.2 Output

0
1

## 3.4 Break with goto

In Lua programming language, the loop can be terminated using goto statement.

### 3.4.1 Code

```
i = 0
while (i < 3)
do
   if (i == 2) then goto breakwithgoto end
   print(i)
   i = 1 + i
end
::breakwithgoto::
```

### 3.4.2 Output

0
1

### 3.5 Continue with goto

In the Lua programming language, continue can be done using the goto statement as can be seen below. The program skipped printing 1 using the goto statement.

#### 3.5.1 Code

```
i = 0
while (i < 3)
do
   if (i == 1) then goto continuewithgoto end
   print(i)
   ::continuewithgoto::
   i = 1 + i
end
```

#### 3.5.2 Output

```
0
2
```

### 3.7 Evaluation of Lua programming language

As in the programming languages Dart and Javascript, the programming language Lua has curly braces to define scopes. Lua gives about the same readability as the others compared to Dart and the Javascript languages, because Lua still uses a large number of keywords in English. Lua language also uses a large amount of english words to handle the loops. It can be inferred that since they present similar features, Lua displays the same writability as Javascript and Dart. On the other hand, the goto statement makes Lua harder to read and write.

# 4.0 PHP

## 4.1 Pretest

In PHP programming language pretest loop can be done using while loop.

### 4.1.1 Code

```
$i = 0;
while ($i < 3){
    echo ("$i\n");
    $i++;
}
```

### 4.1.2 Output

```
0
1
2
```

## 4.2 Posttest

In PHP programming language, posttest can be done using do while loop.

### 4.2.1 Code

```
$i = 0;
do {
    echo ("$i\n");
    $i++;
} while($i < 3);
```

### 4.2.2 Output

```
0
1
2
```

## 4.3 Unlabeled Break

In the PHP programming language, a loop can be terminated using the break statement. In the example below, 2 cannot be printed because loop is terminated.

### 4.3.1 Code

```
$i = 0;
while ($i < 3){
    if ($i == 2) break;
    echo ("$i\n");
    $i++;
}
```

### 4.3.2 Output

```
0
1
```

## 4.4 Unlabeled Continue

In PHP programming language, current loop iteration can be skipped using continue statement. In the example below, 2 is not printed because it is skipped.

### 4.4.1 Code

```
$i = 0;
while ($i < 3){
    $i++;
    if ($i == 2) continue;
    echo ("$i\n");
}
```

### 4.4.2 Output

```
1
3
```

## 4.5 Break outer loop

In PHP the outer loop can be terminated using break 2 statement.

### 4.5.1 Code

```
$i = 0;
$j = 0;
while ($i < 2) {
    while ($j < 2) {
        echo("$i, $j\n");
        $j++;
        if ($i == $j) {break 2;}
    }
    $i++;
    echo("$i\n");
}
```

### 4.5.2 Output

```
0, 0
0, 1
1
2
```

## 4.6 Evaluation of PHP programming language

Since PHP requires a dollar sign before variables, this makes it more difficult to read and write the PHP programming language. In addition, at the start of the program, it includes a php tag.

# 5.0 Python3

## 5.1 Pretest

In the python3 programming language, pretest can be done using a while loop.

### 5.1.1 Code

```
i = 0
while i < 3:
    print(i)
    i += 1
```

### 5.1.2 Output

```
0
1
2
```

## 5.2 Posttest

There is no exact do while or similar statement in python3. But it can be simulated using the break statement as can be seen below.

### 5.2.1 Code

```
i = 0
while True:
    print(i)
    i += 1
    if not i < 3:
        break
```

### 5.2.2 Output

```
0
1
2
```

## 5.3 Unlabeled Break

The Break statement can be used to terminate current iteration of the loop in python. As can be seen below, 2 is not printed because the loop is terminated.

### 5.3.1 Code

```
i = 0
while i < 3:
    print(i)
    if i == 1:
        break
    i += 1
```

### 5.3.2 Output

0
1

## 5.4 Unlabeled Continue

In the python3 programming language, the continue statement is used to skip current iteration in the loop. 2 is skipped using continue statement.

### 5.4.1 Code

```
i = 0
while i < 3:
    i += 1
    if i == 2:
        continue
    print(i)
```

### 5.4.2 Output

1
3

### 5.5 Evaluation of Python programming language

Until now, in the sense of readability, it can be argued that Python and Dart is the simplest programming language since it is so similar to the English language. Python requires fewer symbols because it uses spaces instead of curly braces for indentation. It can therefore be inferred that the language of python programming is effective in the sense of writing. But there are no do while and labels for continue and break. Labels were proposed before for python but it was rejected due to the chance of reducing readability in some cases.

Link: https://www.python.org/dev/peps/pep-3136/#rejection-notice

# 6.0 Ruby

## 6.1 Pretest

In the Ruby programming language, pretest can be implemented using while loop.

### 6.1.1 Code

```
$i = 0
while $i <3 do
   puts $i
   $i = 1 + $i
end
```

### 6.1.2 Output

```
0
1
2
```

## 6.2 Posttest

In the Ruby programming language, posttest can be done using begin end while statement.

### 6.2.1 Code

```
$i = 0
begin
   puts $i
   $i = 1 + $i
end while $i < 3
```

### 6.2.2 Output

```
0
1
2
```

## 6.3 Unlabeled Break

In the ruby programming language, the loop can be terminated using the break statement. As can be seen below, 3rd iteration is not done because the loop is terminated.

### 6.3.1 Code

```
$i = 0
while $i <3 do
   if $i ==2
      break
   end
   puts $i
   $i = 1 + $i
end
```

### 6.3.2 Output

```
0
1
```

## 6.4 Unlabeled Continue

In the ruby programming language, the current iteration of the loop can be skipped by the next statement. This is similar to the continue statement. As can be seen in the output, printing 2 is skipped.

### 6.4.1 Code

```
$i = 0
while $i <3 do
   $i = 1 + $i
   if $i ==2
      next
   end
   puts $i
end
```

### 6.4.2 Output

```
1
3
```

### 6.5 Evaluation of Ruby programming language

As in most of the languages I have analyzed, because the programming language of Ruby uses English keywords to handle logically managed loops, Ruby seems to have similar readability and writability.

# 7.0 Rust

## 7.1 Pretest

In the Rust programming language pretest loop can be done using the while statement.

### 7.1.1 Code

```
let mut i = 0;
while i < 3 {
    println!("{}", i);
    i = i + 1
}
```

### 7.1.2 Output

```
0
1
2
```

## 7.2 Posttest

In the rust programming language, to posttest, infinite loop and break at the end can be used.

### 7.2.1 Code

```
i = 0;

loop {
    println!("{}", i);
    i = i + 1;
    if i > 2 { break; }
}
```

### 7.2.2 Output

```
0
1
2
```

## 7.3 Unlabeled Break

Td.

### 7.3.1 Code

```
i = 0;
while i < 3 {
    if i == 2 {break;}
    println!("{}", i);
    i = i + 1
}
```

### 7.3.2 Output

```
0
1
```

## 7.4 Unlabeled Continue

To skip current iteration in a loop in Rust programming language, the continue statement can be used as below. 2 is not printed because it is skipped.

### 7.4.1 Code

```
i = 0;
while i < 3 {
    i = i + 1;
    if i == 2 {continue;}
    println!("{}", i);
}
```

### 7.4.2 Output

```
1
3
```

## 7.5 Labeled Break

In the Rust programming language, the break statement can be used with a label.

### 7.5.1 Code

```
'label: loop {
    loop {
        println!("test");
        break 'label;
    }
}
```

### 7.5.2 Output

test

## 7.6 Labeled Continue

In Rust programming language, the continue statement can be used with labels so that outer loop iterations can be skipped.

### 7.6.1 Code

```
i = 0;
let mut j = 0;
'anotherlabel: while i < 2 {
    while j < 2 {
      println!("{}{}", i, j);
      j = j + 1;
      if i == j {continue 'anotherlabel;}
    }
i = i + 1;
println!("{}",i);
}
```

### 7.6.2 Output

00
01
1
2

## 7.7 Return value with break

In the Rust programming language, with the break statement, a variable can be returned from the loop as can be seen below.

### 7.7.1 Code

```
i = 0;
   let return_value = loop {
      i = i + 2;
      if i == 6 {break i;}
   };

   println!("{}", return_value);
```

### 7.7.2 Output

```
6
```

## 7.8 Evaluation of Rust programming language

Since the programming language of Rust has an enormous number of additional symbols and functions required to perform some operations compared to other languages, it can be argued that in terms of readability, Rust is the toughest programming language. This also makes the programming language of Rust more difficult to write. But in terms of logically controlled loops, one can benefit from it a lot since it is too powerful with its features in that context.

## 8.0 Learning Strategy

I tried to stick to the official documents of the programming languages when studying all of the 7 languages. I check for stack overflow if I have problems after looking at the official documents, then I find solutions to my problems. These two methods help me to complete this homework so that it can be said that they are adequate.

**URLs of the official documentations**

1. Dart https://api.dart.dev/stable/2.10.4/index.html
2. Javascript by Mozilla https://developer.mozilla.org/en-US/docs/Web/JavaScript
3. Lua https://www.lua.org/
4. PHP https://www.php.net/docs.php
5. Python3 https://docs.python.org/3/
6. Ruby https://www.ruby-lang.org/en/documentation/
7. Rust https://doc.rust-lang.org/

**URLs of the online compiler/interpreters**

8. Dart https://dartpad.dev/
9. Javascript https://repl.it/languages/javascript
10. Lua https://www.tutorialspoint.com/execute_lua_online.php
11. PHP https://sandbox.onlinephpfunctions.com/
12. Python3 https://www.onlinegdb.com/online_python_compiler
13. Ruby https://repl.it/languages/ruby
14. Rust https://repl.it/languages/rust

## 9.0 Conclusion

In my opinion, Dart is the best language for me in terms of readability, and Ruby is the worst. Since Python and Dart are too similar to the English language, their readability is almost the same. But Dart has better features in terms of writing logically controlled loops. In addition, the Rust language is too powerful in terms of writing logically controlled loops. In brief, Dart is the best programming language since it is too close to english while reading, and while writing since it has a lot of features for logically controlled loops.