2021–2022 SPRING SEMESTER

CS315 – HW2

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

NAME: ABDULLAH RIAZ
ID : 22001296
COURSE:CS315 - PROGRAMMING LANGUAGES
SECTION: 01
DATE: 23/04/2022

Dart

- Supports both pretest and posttest
- Has unconditional labeled and unlabeled exit
- Control Mechanisms: Break, Continue statements

- Code Segment

```
bool whileCond = false;
int i = 0;
outerLoop:
while (i < 50) {
    while (whileCond != true) {
      print("Inner While Loop Iteration: $i");
      i++;
      if (i > 10) {
       print("Exiting Outer While Loop");
       break outerLoop;
         }
      if (i == 20) {
       whileCond = true;
        }
      }
     print ("Inner While Loop Finished Iteration");
}
```

- Output

```
Inner While Loop Iteration: 0
Inner While Loop Iteration: 1
Inner While Loop Iteration: 2
Inner While Loop Iteration: 3
Inner While Loop Iteration: 4
Inner While Loop Iteration: 5
Inner While Loop Iteration: 6
Inner While Loop Iteration: 7
Inner While Loop Iteration: 8
Inner While Loop Iteration: 9
Inner While Loop Iteration: 10
Exiting Outer While Loop
```

This code segment shows an example of a pretested logically controlled loop. Using an outer while loop, we can also show the functionality of dart to exit from outer loops as well.
In the execution result, we can see that when ‘i’ reaches 10 our inner while loop exits from both the loops as we used a label name after exit

- Code Segment

```
i = 0;
whileCond = false;

  do {
      print("Do While Loop Iteration: : $i");
      i++;
       if (i > 10) {
        print("Exiting Do While Loop");
         break;
       }
  } while (whileCond != true);
```

- Output

```
Do While Loop Iteration: : 0
Do While Loop Iteration: : 1
Do While Loop Iteration: : 2
Do While Loop Iteration: : 3
Do While Loop Iteration: : 4
Do While Loop Iteration: : 5
Do While Loop Iteration: : 6
Do While Loop Iteration: : 7
Do While Loop Iteration: : 8
Do While Loop Iteration: : 9
Do While Loop Iteration: : 10
Exiting Do While Loop
```

In this code segment, a posttest logically controlled loop is used. Furthermore a break statement without label is used to show that Dart supports both labeled and unlabeled loop exits.In case of unlabeled exit, it exits from the innermost loop only.

Javascript

- Supports both pretest and posttest
- Has unconditional labeled and unlabeled exit
- Control Mechanisms: Break, Continue statements

- Code Segment

```
let whileCond = false;
let i = 0;
outerLoop:
while (i < 50) {
  while (whileCond != true) {
    console.log(`Inner While Loop Iteration: ${i}`);
    i++;
    if (i > 10) {
      console.log('Exiting Outer While Loop');
      break outerLoop;
    }
    if (i == 20) {
      whileCond = true;
    }
  }
  console.log('Inner While Loop Finished Iteration');
}
```

- Output

```
Inner While Loop Iteration: 0
Inner While Loop Iteration: 1
Inner While Loop Iteration: 2
Inner While Loop Iteration: 3
Inner While Loop Iteration: 4
Inner While Loop Iteration: 5
Inner While Loop Iteration: 6
Inner While Loop Iteration: 7
Inner While Loop Iteration: 8
Inner While Loop Iteration: 9
Inner While Loop Iteration: 10
Exiting Outer While Loop
```

In the example above, we can see a pretest loop in Javascript language. It also has a break inside a if statement as Javascript does not support conditional exits natively. Using a label with exit, we can exit the outermost loop from inside the nested loop. In the output we can see only '*Exiting Outer While Loop*' and no other console log from the code segment since the exit in inner while loop exits out of both the loops.

- Code Segment

```
i = 0;
whileCond = false;

do {
  do {
    console.log(`Do While Loop Iteration: ${i}`);
    i++;
    if (i > 10) {
      console.log('Exiting Do While Loop');
      break;
    }
  } while (whileCond != true);
  console.log('Inner Do While Loop Finished Iteration');
} while (i < 1);
```

- Output

```
Do While Loop Iteration: 0
Do While Loop Iteration: 1
Do While Loop Iteration: 2
Do While Loop Iteration: 3
Do While Loop Iteration: 4
Do While Loop Iteration: 5
Do While Loop Iteration: 6
Do While Loop Iteration: 7
Do While Loop Iteration: 8
Do While Loop Iteration: 9
Do While Loop Iteration: 10
Exiting Do While Loop
Inner Do While Loop Finished Iteration
```

This code segment tests the posttest loop in javascript. It means that any statement within the loop would run once at least. It also has an unlabeled exit which means it can only escape the innermost loop from where it is called. This can be noticed by the presence of 'Inner Do While Loop Finished Iteration' output . If it had exited both loops like in the previous example, this output would not be there.

Perl

- Supports both pretest and posttest
- Has both conditional and unconditional labeled and unlabeled exit
- Control Mechanisms: next, last statements
- Code Segment

```perl
$whileCond = 0;
$i = 0;

outerLoop:
while ($i < 50) {
    while ($whileCond != 1) {
    print("Inner While Loop Iteration: $i \n");
    $i++;
    if ($i > 10) {
          print("Exiting Outer While Loop");
          last outerLoop;
          }
    if ($i == 20) {
          $whileCond = 1;
          }
    }
    print ("Inner While Loop Finished Iteration");
}
```

```perl
$whileCond = 0;
$i = 0;

outerLoop:
while ($i < 50) {
    while ($whileCond != 1) {
    print("Inner While Loop Iteration: $i \n");
    $i++;
    if ($i > 10) {
          print("Exiting Outer While Loop");
          }
    last outerLoop if ($i > 10) ;
    if ($i == 20) {
          $whileCond = 1;
          }
    }
    print ("Inner While Loop Finished Iteration");
```

- Output(same for both code segments above)

```
Inner While Loop Iteration: 0
Inner While Loop Iteration: 1
Inner While Loop Iteration: 2
Inner While Loop Iteration: 3
Inner While Loop Iteration: 4
Inner While Loop Iteration: 5
Inner While Loop Iteration: 6
Inner While Loop Iteration: 7
Inner While Loop Iteration: 8
Inner While Loop Iteration: 9
Inner While Loop Iteration: 10
Exiting Outer While Loop
```

In the code examples above we can see a pretested logically controlled loop. The key difference between both the examples is the exit conditional mechanism. Unlike other languages, conditional exit is part of the Perl language and an if statement is not needed to make a conditional exit. The first code segment is using an if statement to control the exit while the next one uses the native conditional exit mechanism. The native conditional exit mechanism also supports labeled exits.The outputs are the same for both for these code segments.

- Code Segment

```
$whileCond = 0;
$i = 0;
  do {
      print("Do While Loop Iteration: $i \n");
      $i++;
      if ($i > 10) {$whileCond = 1};
  } while ($whileCond != 1);
```

- Output

```
Do While Loop Iteration: 0
Do While Loop Iteration: 1
Do While Loop Iteration: 2
Do While Loop Iteration: 3
Do While Loop Iteration: 4
Do While Loop Iteration: 5
Do While Loop Iteration: 6
Do While Loop Iteration: 7
Do While Loop Iteration: 8
Do While Loop Iteration: 9
Do While Loop Iteration: 10
```

A Do While loop(posttest) in Perl is not considered as loop as far as control mechanism statements such as next or last are concerned so they cannot be used inside a do while loop .

PHP

- Supports both pretest and posttest
- Has unconditional unlabeled exit only.
- Control Mechanisms: Break, Continue statements

- Code Segment

```php
$whileCond = false;
$i = 0;

while ($i < 50) {
  while ($whileCond != true) {
    echo "Inner While Loop Iteration: ${i} <br />";
    $i++;
    if ($i > 10) {
      echo "Exiting Outer While Loop <br />";
      break 2;
    }
    if ($i == 20) {
      $whileCond = true;
    }
  }
  echo "Inner While Loop Finished Iteration <br />";
}
```

- Output

```
Inner While Loop Iteration: 0
Inner While Loop Iteration: 1
Inner While Loop Iteration: 2
Inner While Loop Iteration: 3
Inner While Loop Iteration: 4
Inner While Loop Iteration: 5
Inner While Loop Iteration: 6
Inner While Loop Iteration: 7
Inner While Loop Iteration: 8
Inner While Loop Iteration: 9
Inner While Loop Iteration: 10
Exiting Outer While Loop
```

In the code segment above, a pre tested loop can be seen in PHP language. However, compared to other languages PHP does not use labels and instead uses a number corresponding to the loops..The innermost loop is considered number 1 and and the next outermost loop is considered 2 and so on.

- Code Segment

```
$i = 0;
$whileCond = false;

do {
  do {
    echo "Do While Loop Iteration: ${i} <br />";
    $i++;
    if ($i > 10) {
      echo "Exiting Do While Loop <br />";
      break 1;
    }
  } while ($whileCond != true);

  echo "Inner Do While Loop Finished Iteration <br />";

} while ($i < 1)
```

- Output

```
Do While Loop Iteration: 0
Do While Loop Iteration: 1
Do While Loop Iteration: 2
Do While Loop Iteration: 3
Do While Loop Iteration: 4
Do While Loop Iteration: 5
Do While Loop Iteration: 6
Do While Loop Iteration: 7
Do While Loop Iteration: 8
Do While Loop Iteration: 9
Do While Loop Iteration: 10
Exiting Do While Loop
Inner Do While Loop Finished Iteration
```

PHP also supports post tested logically controlled loops as seen in the code segment above.

Python

- Supports only pretest
- Has unconditional unlabeled exit only.
- Control Mechanisms: Break, Continue, Else statements

- Code Segment

```python
whileCond = False
i = 0

while whileCond != True :
      print("Inner While Loop Iteration: " , i );
      i = i + 1
      if i > 10 :
      print('Exiting inner While Loop');
      break
```

- Output

```
Inner While Loop Iteration:  0
Inner While Loop Iteration:  1
Inner While Loop Iteration:  2
Inner While Loop Iteration:  3
Inner While Loop Iteration:  4
Inner While Loop Iteration:  5
Inner While Loop Iteration:  6
Inner While Loop Iteration:  7
Inner While Loop Iteration:  8
Inner While Loop Iteration:  9
Inner While Loop Iteration:  10
Exiting inner While Loop
```

Python only supports pretested logically controlled loops. Furthermore Python also only supports unconditional unlabeled exits.

Readability and Writability

In the langues compared above, Python would be the language with the least writability and readability in terms of logically-controlled loops. The language doesn't support do while loops at all and has no labeled exits meaning exiting outerloops is very hard. The lack of integrated control mechanism for exits also decreases the writability of Python. PHP is better in both writability and readability compared to python as it supports both pretests and protest controlled loops. However it falls short in terms of readability as there are no labeled exits meaning it can be hard to identify which loop is being excited. Dart and JavaScript have similar level of readability and writability in terms of logically-controlled loops as they both provide pretest and posttest controlled loops as well as provided labeled exits which improves the readability of these two languages. However the best in terms of both readability and writability in these languages is perl as it provides conditional mechanism as part of exit in loops which reduces the need for writing extra if statements and improves the writability and readability of Perl as a language

Learning Strategy

I started with the course book “Concepts of Programming Languages” under Logically controlled loops. After getting an idea of it, I moved on to the official documentation of the languages in the assignment to understand how they handle do..while and while loop as well as control statements associated with them. Then I used online compilers to write code for each of the languages(online compilers are listed in references) and gave my analysis regarding the difference of these languages.

References:

Python online compiler. Online Python Compiler (Interpreter). (n.d.). Retrieved April 24, 2022, from https://www.programiz.com/python-programming/online-compiler/

Execute perl online. Online Perl Compiler. (n.d.). Retrieved April 24, 2022, from https://www.tutorialspoint.com/execute_perl_online.php

Dartpad. DartPad. (n.d.). Retrieved April 24, 2022, from https://dartpad.dev/

PHP Online compiler. PHP Online Compiler (Editor / Interpreter). (n.d.). Retrieved April 24, 2022, from https://www.w3schools.com/php/php_compiler.asp

PERL 5.34.1 documentation. Perl Documentation - Perldoc Browser. (n.d.). Retrieved April 24, 2022, from https://perldoc.perl.org/

Python 3.10.4 documentation. 3.10.4 Documentation. (n.d.). Retrieved April 24, 2022, from https://docs.python.org/3/

While - manual. php. (n.d.). Retrieved April 24, 2022, from https://www.php.net/manual/en/control-structures.while.php

While - javascript: MDN. JavaScript | MDN. (n.d.). Retrieved April 24, 2022, from https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/while