CS315 HOMEWORK 2 REPORT

Onurcan Ataç
22002194
Section 2

Implementations

Explanations of the example programs are made by detailed comments on code parts. Different mini programs in codes are separated with slashes. Online compilers used are given right after the code segments of every language.

1.Dart

Code:

```dart
void main()
{
    // Part 1
    // should the conditional mechanism be an integral part of the
exit (conditional or unconditional exit)

    // in dart there are break statements, hence the conditional
mechanism does not have to be an integral part of the exit

    for( var i = 0; i < 5; i++)
    {
        print(i);
        break;
    }

    print("--------");

    // continue statement moves to the next iteration of the loop
while skipping the code before the next iteration

    for( var i = 0; i < 5; i++)
    {
        print(i);
        continue;
        print("something");
    }

    print("--------");

    // behaviour when not in a loop
    // when written outside of a loop, both break and continue
statements give an error that mentions those cannot be used outside
of a loop

    // Part 2
    // should only one loop body be exited or can enclosing loops
also be exited (labeled or unlabeled exit)

    // dart supports labeled exits, and they make it convenient to
exit multiple loops

    out:
    for( var i = 0; i < 5; i++)
    {
        for(var k = 10; k < 15; k++)
        {
            print(k);
            break out;
        }
    }

    print("--------");

    // in this kind of use, continue statement continues executing
from the next iteration of the
    // outer loop

    outside:
    for( var i = 0; i < 5; i++)
    {
        for(var k = 10; k < 15; k++)
        {
            print(k);
            continue outside;
            print("something");
        }
    }

    print("--------");

    // in dart, unlabeled exits still exit from only one loop

    for( var i = 0; i < 5; i++)
    {
        for(var k = 15; k < 20; k++)
        {
            print(k);
            break;
        }
```

```dart
    }

    print("--------");

    for( var i = 0; i < 2; i++)
    {
        for(var k = 15; k < 20; k++)
        {
            print(k);
            continue;
            print("something");
        }
    }
}
```

Code is ran by using https://dartpad.dev/

Output:

```
0

--------

0

1

2

3

4

--------

10

--------

10

10

10

10

10

--------

15
```

```
15

15

15

15

--------

15

16

17

18

19

15

16

17

18

19
```

2.Javascript

Code:

```
<!DOCTYPE html>
<html><head><script>//Part 1
  //should the conditional mechanism be an integral part of the exit
(conditional or unconditional exit)

  //javascript has break statement, hence conditional mechanism
shouldn't be an integral part of the exit

  for (let i = 0; i < 5; i++) {
    console.log(i);
    break;
  }

  console.log("--------");

  //continue statement passes to the next iteration of thhe loop
without considering the remaining code of that iteration

  for (let i = 0; i < 5; i++) {
    console.log(i);
    continue;
    console.log("something");
  }

  console.log("--------");

  //when used outside the loop
  //javascript displays errors for both break and continue when not
used in a loop

  //Part 2
  //should only one loop body be exited or can enclosing loops also
be exited (labeled or unlabeled exit)

  //javascript has labeled break statement, that enables programmer
to exit multiple loops at once

  out: for (let i = 0; i < 5; i++) {
    for (let k = 15; k < 20; k++) {
      console.log(k);
      break out;
    }
  }

  console.log("--------");

  // in this kind of use, continue statement continues executing
from the next iteration of the
  // outer loop

  outside: for (let i = 0; i < 2; i++) {
    for (let k = 15; k < 20; k++) {
      console.log(k);
      continue outside;
      console.log("something");
    }
  }

  console.log("--------");

  //unlabeled break statement works as expected, leaving only one
loop

  for (let i = 0; i < 5; i++) {
    for (let k = 10; k < 15; k++) {
      console.log(k);
      break;
    }
  }

  console.log("--------");

  //unlabeled continue works as expected, only exiting the inner
loop iteration

  for (let i = 0; i < 2; i++) {
    for (let k = 15; k < 17; k++) {
      console.log(k);
      continue;
      console.log("something");
    }
  }

</script></head><body>output is printed on the console</body></html>
```

Javascript version of the code was ran by https://onecompiler.com/javascript , browser version of html code was ran by live server feature of VS Code.

Output:

```
0
--------
0
1
2
3
4
--------
15
--------
15
15
--------
10
10
10
10
10
--------
15
16
15
16
```

3.Lua

Code:

```lua
--Part 1
--should the conditional mechanism be an integral part of the exit (conditional or unconditional exit)

--since lua has break statement, conditional mechanism does not have to be an integral part of the exit

for i = 0,5,1
do
   print(i)
   break
end

print("--------")

--lua does not include continue statement

--use of break outside the loop
--when used outide of a loop, lua gives an error indicating that break cannot be used outside a loop

--Part 2
--should only one loop body be exited or can enclosing loops also be exited (labeled or unlabeled exit)

--lua does not have labeled break statement, but it has "goto" which can bu used in similar fashion
--to exit multiple loops at the same time

for i = 0,5,1
do
   for k = 10,15,1 do
    print(k)
    goto done
   end
end
::done::

print("--------")

--unlabeled break exits only from one loop as expected

for i = 0,5,1
do
```

```
   for k = 5,10,1 do
    print(k)
    break
   end
end
```

Code is ran by using https://onecompiler.com/lua

Output:

```
0
--------
10
--------
5
5
5
5
5
5
```

4.PHP

Code:

```
<?php
// Part 1
// should the conditional mechanism be an integral part of the exit
(conditional or unconditional exit)

// there are break statements in PHP, so conditional mechanism isn't
always the integral part of the exit

for ($x = 0; $x < 5; $x++)
{
echo "$x \n";
break;
}

echo "-------- \n";

// continue statements exit the current iteration of the loop and
passes onto the next one, disregarding the
// remaining code in that iteration

for ($x = 0; $x < 5; $x++)
```

```php
{
echo "$x \n";
continue;
echo "something";
}

echo "-------- \n";

// when break or continue is used outside the loop
// PHP displays fatal error caused from break and continue not being
inside a loop

// Part 2
// should only one loop body be exited or can enclosing loops also
be exited (labeled or unlabeled exit)

// there are different practices in PHP for this problem, one of
them is putting the no of nested loops
// after the break statement, that way it can exit multiple loops at
once

for ($x = 0; $x < 5; $x++)
{
    for($y = 5; $y < 10; $y++)
    {
        echo "$y \n";
        break 2;
    }
}

echo "-------- \n";

// continue can be used in the same manner, getting out of the
iteration of the outer loop

for ($x = 0; $x < 5; $x++)
{
    for($y = 5; $y < 10; $y++)
    {
        echo "$y \n";
        continue 2;
    }
}

echo "-------- \n";

// another practice is using "goto" for exiting multiple loops

for ($x = 0; $x < 5; $x++)
```

```php
{
    for($y = 5; $y < 10; $y++)
    {
        echo "$y \n";
        goto out;
    }
}
out:

echo "-------- \n";

//break statements exit only one loop at a time as expected

for ($x = 0; $x < 5; $x++)
{
    for($y = 5; $x < 10; $x++)
    {
        echo "$y \n";
        break;
    }
}

echo "-------- \n";

//continue statements exit only the inner loop iteration as expected

for ($x = 0; $x < 2; $x++)
{
    for($y = 5; $y < 8; $y++)
    {
        echo "$y \n";
        continue;
        echo "something";
    }
}
?>
```

Code is ran by using https://onecompiler.com/php

Output:

```
0
--------
0
1
2
3
```

```
4
--------
5
--------
5
5
5
5
5
--------
5
--------
5
5
5
5
5
--------
5
6
7
5
6
7
```

5.Python

Code:

```python
# Part 1
# should the conditional mechanism be an integral part of the exit (conditional or unconditional exit)

# in python there are break statements, hence the conditional mechanism doesn't have to be an integral part of the exit

for x in range(0,5):
  print(x)
  break

print("--------")

# continue statement skips the part after it and passes onto next iteration in the loop

for x in range(5,10):
  print(x)
  continue
  print("something")

print("--------")

# behaviour when not in a loop
# when not used in a loop, both break and continue statement end up
with a syntax error, which mentions that they cannot
# be used outside of the loop

# Part 2
# should only one loop body be exited or can enclosing loops also be
exited (labeled or unlabeled exit)

# python does not support labeled exits, and in unlabeled exits
break statement only exits only one loop body

for x in range(0,5):
  for y in range(15,20):
    print(y)
    break

print("--------")

# continue statement skips the part after it and passes onto next
iteration in the loop, it only works for the inside loop in this
situation

for x in range(0,2):
  for y in range(15,18):
    print(y)
    continue
    print("something")
```

Code is ran by using https://onecompiler.com/python

Output:

```
0
--------
5
6
7
8
9
--------
15
15
```

```
15
15
15
--------
15
16
17
15
16
17
```

6.Ruby

Code:

```
#Part 1
#should the conditional mechanism be an integral part of the exit
(conditional or unconditional exit)

#conditional mechanism does not have to be an integral part of the
exit since ruby has break statement

for i in 0..3
    puts "#{i}"
    break
 end

puts "--------"

#ruby doesn't have continue, but it has next which has the same use

for i in 0..3
    puts "#{i}"
    next
    puts "something"
 end

puts "--------"

#when break and next are used outside of a loop
#ruby displays syntax errors, saying break and next are invalid

#Part 2
#should only one loop body be exited or can enclosing loops also be
exited (labeled or unlabeled exit)

#there aren't any labeled exits in ruby, or even goto statements, so
there isn't any way to get out of
```

```
#multiple loops at once

#break statement only exits from one loop as expected

for i in 0..1
    for k in 5..7
    puts "#{k}"
    break
    end
 end

puts "--------"

#next behaves like continue statements in other languages, ending
the specific iteration of the inner loop

 for i in 0..1
    for k in 5..7
    puts "#{k}"
    next
    puts "something"
    end
 end
```

Code is ran by using https://onecompiler.com/ruby

Output:

```
0
--------
0
1
2
3
--------
5
5
--------
5
6
7
5
6
7
```

7.Rust

Code:

```rust
fn main()
{
//Part 1
//should the conditional mechanism be an integral part of the exit
(conditional or unconditional exit)

//conditional mechanism doesn't have to be the integral part of the
exit since break statement exists
//in rust

for x in 1..3
{
    println!("{}", x);
    break;
}

println!("--------");

//continue expectedly end the current iteration of the loop and
proceeds to the next one,
//while disregarding remaining code in current iteration

for x in 1..3
{
    println!("{}", x);
    continue;
    println!("something");
}

println!("--------");

//when break or continue statements are used outside of a loop, rust
displays an error indicating that those
//cannot be used outside of a loop

//Part 2
//should only one loop body be exited or can enclosing loops also be
exited (labeled or unlabeled exit)

//labeled break statement can be used to exit multiple loops at once

'out:
for x in 1..5
{
  for y in 5..10
  {
    println!("{}", y);
    break 'out;
  }
}

println!("--------");

//by specifying the loop by a label, continue statement can work on
inner or outer loops
//in the following example, continue works on the outer loop which
is specified by a label
//labeled break can be used in the same way as well

'outside:
for x in 1..3
{
  for y in 5..8
  {
    println!("{}", y);
    continue 'outside;
  }
}

println!("--------");

//break and continue statements exit only one loop at once as
expected

for x in 1..3
{
  for y in 5..8
  {
    println!("{}", y);
    break;
  }
}

println!("--------");

for x in 1..3
{
  for y in 5..8
  {
    println!("{}", y);
    continue;
    println!("something");
  }
}
```

```
}
```

Code is ran by using https://onecompiler.com/rust

Output:

```
1
--------
1
2
--------
5
--------
5
5
--------
5
5
--------
5
6
7
5
6
7
```

Evaluation of Languages

My expectations for a language to be preferable to use, are that it provides necessary features for abstraction (which is correlated with writability) and makes those features easy to use as possible in terms of readability and writability.

Then, by looking at these criteria, a language must provide both break and continue, also give an easy option for exits in multiple loops. Python and Ruby does not comply with these criteria, since they don't provide ways to leave multiple loops at once.

Moreover, Lua does not provide a specific feature to exit loops, user must use a goto statement which is generally a bad practice. PHP provides both goto and multiple loop exits, by that manner language becomes incohesive. If PHP is used in a big project, different programmers would use different options and readability of the code would decrease.

Dart, Rust, Javascript provide features needed and cohesive at the same time. In Rust, simpler operations are hard to do in my opinion. That provides readability and writability issues. Between Dart and Javascript, I didn’t recognize a significant advantage of one against the other. Because of that, both those languages can be used without many problems in terms of the User-Located Loop Control Mechanisms.

Learning Strategy

For his task, I decided to start with Python as I already know some Python and determine my learning strategy after completing the program with the first language. After I completed Python, I realized that there might be enough explanation on the internet about unconditional exits in other languages as well. For every language, I first searched if there are break and continue statements (or statements with similar use but different name) in the language, I then tried to use them outside a loop and saw the output. Then, I searched for ways that I can used those statements in nested loops. I also examined with default break and continue statements in nested loops. I completed the homework repeating this routine for each language. At some parts, I also experimented on the online compilers to check if some syntax that might work, using my experience from other programming languages.

References

I used online compilers, which are added at the end of each code segment.

In terms of learning special cases for different programming languages, I searched for what I needed in Google and navigated through many tabs. I didn’t keep track of all the internet sites that I went through, they were usually forum sites such as https://stackoverflow.com/ .