Muhammed İkbal Doğan

21702990

CS315 HW2

1- DART

CODE

```
void main() {
  var x = 0;
  var y = 0;

 label1:
  while(y < 10){


     label2:
        while (true) {

            if (x == 5) {
              break label1;
            }

           if (x == 3) {
              x = x + 1;
              continue;
            }
           print('x = $x' + ' y = $y');
           x = x + 1;
        }
    y = y  + 1;

   }

  print('END');

}
```

OUTPUT

```
x = 0 y = 0
x = 1 y = 0
x = 2 y = 0
x = 4 y = 0
END
```

In the dart code I wrote two nested loops. When x equals to 3, it passes the print statements and continue from inner loop which is labeled as label2. When x equals 5 it breaks the not only inner loop but also outer loop. It is happened because dart supports unconditional exits(break and continue are shown) and it also supports labaled exits. Therefore when I wrote break label1 it exists from outer loop too therefore y is not incremented. So as it seen dart supports both unconditional exit and labeled exit.

2-JAVASCRIPT

CODE

index.js 3yqmbp83y

```javascript
label2:
for (let y = 0; y <= 5; y++) {
    console.log(y);

    label1:
        for (let i = 1; i <= 5; i++) {
          if (i  == 2) { continue; }
                console.log(i);
          if (i == 4) { break label2; }
        }
}
```

OUTPUT

STDIN

Input for the program (Optional)

Output:

```
0
1
3
4
```

In the javascript code, I wrote two nested loops and two labels which are named label1 and label2 as seen in the code. When i is equal to 2 it passes the label1 for loop and continues. When i equals to 4 it breaks the loop and also it breaks the outer loop as seen in the output it just prints 0 for y but it does not continue because it breaks the outer loop too with the help of labels. So javascript supports unconditional exists and also it supports labeled exits.

3-LUA

CODE

```
a = 10
b = 0
c = 0
while (c < 2)
do
::outerloop::
print("++++value of c:", c)
   while (b < 4)
        do
            print("----value of b:", b)
            while( a < 20 )
            do
                a=a+1
                print("value of a:", a)
                if( a == 15)
                then
                    break
                end
                if(a == 16)
                then
                goto outerloop
                end
            end
             b = b+1
            end
    c = c+1
end
print("program is ended")
```

OUTPUT

```
++++value of c: 0
----value of b: 0
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 15
----value of b: 1
value of a: 16
++++value of c: 0
----value of b: 1
value of a: 17
value of a: 18
value of a: 19
value of a: 20
----value of b: 2
----value of b: 3
++++value of c:1
program is ended
```

In the Lua code I wrote three nested loops. When a equals to 15 it breaks the most inner loop and goes to outer loop and b is incremented 1 by b = b+1. So it supports unconditional exits but there is no continue statement in dart. Rather than continue goto statements can be used. Goto statement supports labeled exits but when I try to jump to local label it gave an error because lua does not support jumps on local labels. To show we can break 2 nested loops I used goto outer loop and it jumped to outer loop label without incrementing c. So it supports labeled exits but labels should be in a loop. If label is in local it will give an error.

4-PHP

CODE

```php
<?php
   $x = 1 ;
   while($x)
   {
        for($j =10 ; ;$j ++)
        {
            if($j == 11)
            {
               continue;
            }
            if($j == 15)
            {
                break 2;
            }
            echo $j;
            echo "\n";
        }
      $x ++ ;
   }
   while($x)
   {
        echo $x;
        echo "\n";
        if($x == 5){
            goto end;
        }
        $x++;
   }
   end:
        echo "program is ended"
```

OUTPUT

```
10
12
13
14
1
2
3
4
5
program is ended
```

When the outputs are checked it can be seen that 11 is passed because of the continue statement. In php after break statements we can define how many loops will be exited. In the example when j equals 15 it executes break 2 and inner for and program exits from innet for and outer while. In the second loop when x equals 5 program moves to end label. So php support unconditional exits as it is mentioned and it also supports labeled exits as seen in the second example.

5-PYTHON

CODE

main.py

```python
no = 0
while no < 10:
    no = no +1
    if no == 5:
        continue
    if no == 7:
        break
    print ('Current No :' ,no)


print ("exit")
```

OUTPUT

```
Current No : 1
Current No : 2
Current No : 3
Current No : 4
Current No : 6
exit
```

In the python code above, I used continue statement when no equals to 5 and as it seen in the output it did not print 5 because it is skipped. When no equals 7 program unconditionally exit so python allows us to exit unconditionally. However there is no labeled exit option in python therefore there is no structure to exit enclosing loops. So it does not allow labeled exits.

6-RUBY

CODE

Execute | Beautify | Share Source Code

```
i = 1
while true

    i += 1
    if i  == 5
        next
    end
    if i  == 10
        break
    end
    puts i
end
```

OUTPUT

```
2
3
4
6
7
8
9
```

In the ruby code I incremented i in the loop every time and when i equals to 5, with next statement it passed the statements under it therefore it did not print 5. Next is working like continue in other languages. When i equals to 10, unconditionally exit from the loop. Therefore, ruby has unconditional exit option. However, in ruby there is no labeled jump structure therefore it could not be implemented with languages own structure.

7-RUST

CODE

Execute | Beautify | Share Source Code Help

```rust
fn main() {
   let mut num = 0;
   while num < 100 {
   num = num +1;
    if num == 3 {
        continue;
    }
    if num == 7{
        break;
    }
    println!("num:{}",num);
    }

    let mut no = 0;
    let mut no2 = 0;
    'outer: while no < 100{
        'inner: while no2 < 100 {
            no2 = no2 +1;
            if no2 == 11{
                break 'outer;
            }
        println!("no:{}",no2);
        }
        no = no +1;
    }
    println!("PROGRAM IS ENDED");
}
```

OUTPUT

```
Terminal
num:1
num:2
num:4
num:5
num:6
no:1
no:2
no:3
no:4
no:5
no:6
no:7
no:8
no:9
no:10
PROGRAM IS ENDED
```

In my rust code first I checked unconditional exits. In the first part I defined a num and in while loop I incremented it by one. Normally it should continue until it equals to 100 but when it equals 7 program exited from that loop unconditionally . In addition when num equals to 3 , program continued and did not print its value. So Rust allows unconditional exit. In the second part I wrote 2 nested loops and when no equals to 11, with break ‘outer statement, program exited from both nested loops with the help of label. In the output it can be seen that program printed until 10 end finished. So Rust allows labeled jumps too.

EVALUATION

In case of writability, ruby and python have negatives because these languages do not give opportunity to exit from nested loops. They do not have labeled exits also. In lua the nested loop exit problem could be solved with goto statements which allows us to labeled exit but in lua we cannot use goto statement to jump on a local block, label should be in a loop therefore it is not good In case of writability. In php we can use break with numbers to indicate how many loops will be exited, even if it is good for writability, it makes code harder to read so it affects readability negatively. Even if python is the best for readability, it is worst for writability. In my opinion, php is the best language in terms of user located loop control mechanisms because it both allows us to break loops as we want and also it has clear labeled exit structure. Break statement with numbers could affect readability in a bad way but it can be solved with label jumps with the help of goto.

LEARNING STRATEGY

Before starting to homework I readed the chapter 8 from the book. Then for dart structure, I used https://o7planning.org/13915/dart-loops and for compiling i used https://dartpad.dev/? .

For javascript structure,

https://www.udacity.com/blog/2021/06/javascript-break-and-continue.html

and for javascript compiling

https://onecompiler.com/javascript/3yqmbp83y

for lua structure,

https://www.educba.com/lua-goto/

https://www.tutorialspoint.com/lua/lua_break_statement.htm

for lua compiling

https://www.tutorialspoint.com/execute_lua_online.php

for php structure,

https://www.w3docs.com/learn-php/php-break-continue-and-goto.html

for php compiling,

https://www.tutorialspoint.com/execute_php_online.php

for python structure,

https://www.c-sharpcorner.com/article/python-unconditional-statements-and-string-operations/

for python compiling,

https://www.online-python.com/

for ruby structure,

https://www.geeksforgeeks.org/ruby-break-and-next-statement/amp/

for ruby compiling,

https://www.tutorialspoint.com/execute_ruby_online.php

for rust structure,

https://rustwiki.org/en/book/ch03-05-control-flow.html

https://doc.rust-lang.org/rust-by-example/flow_control/loop/nested.html

https://doc.rust-lang.org/std/keyword.continue.html

for rust compiling,

https://www.tutorialspoint.com/compile_rust_online.php

are the sources that I used.

REFERENCES

https://o7planning.org/13915/dart-loops

https://dartpad.dev/?

https://www.udacity.com/blog/2021/06/javascript-break-and-continue.html

https://onecompiler.com/javascript/3yqmbp83y

https://www.educba.com/lua-goto/

https://www.tutorialspoint.com/execute_lua_online.php

https://www.tutorialspoint.com/lua/lua_break_statement.htm

https://www.w3docs.com/learn-php/php-break-continue-and-goto.html

https://www.tutorialspoint.com/execute_php_online.php

https://www.c-sharpcorner.com/article/python-unconditional-statements-and-string-operations/

https://www.online-python.com/

https://www.tutorialspoint.com/execute_ruby_online.php

https://www.geeksforgeeks.org/ruby-break-and-next-statement/amp/

https://rustwiki.org/en/book/ch03-05-control-flow.html

https://doc.rust-lang.org/rust-by-example/flow_control/loop/nested.html

https://doc.rust-lang.org/std/keyword.continue.html

https://www.tutorialspoint.com/compile_rust_online.php

Sebesta, Robert W - Concepts of programming languages-Pearson_ W. Ross MacDonald School Resource Services Library (2016_2017)