CS315 - Programming Languages

Homework 2

Deniz Çalkan
21703994
Section 01

1.Design Issues

1.1 Iteration Statements Provided

C

Code Segment

```c
printf("--------------------------------\n");
printf("1. Iteration statements provided\n");
printf("--------------------------------\n");

// In C programming language for, while and do while statements are provided


// The for statement has three control expressions and executes the loop body until the
    second controlling expression evaluates to false.

for (int i = 0; i < 4; i++){

     printf("For ");
     printf("%d\n", i);
}

// The while statement checks the control expression before executing the loop body and
    executes the loop body until the control expression evaluates to false.
int a = 0;
while( a < 4) {

    printf("While ");
    printf("%d\n", a);
    a++;

}

// The do statement checks the contol expression after executing the loop body each
    time and also executes the loop body at least once.
int b = 4;
do {

    printf("Do While ");
    printf("%d\n", b);
    b++;

} while( b < 4);
```

Output

```
--------------------------------
1. Iteration statements provided
--------------------------------
For 0
For 1
For 2
For 3
While 0
While 1
While 2
While 3
Do While 4
```

GO

Code Segment

```
fmt.Print("---------------------------------\n")
fmt.Print("1. Iteration statements provided\n")
fmt.Print("---------------------------------\n")

/* In GO programming language only for statement is provided but
there are different ways to write for loop to satisfy different needs.*/


for i := 0; i < 4; i++ {

     fmt.Print("First For ")
     fmt.Println(i);
}

n := 1
for n < 5 {
     fmt.Print("Second For ")
     fmt.Println(n);
     n = n * 2;
}
```

Output

```
---------------------------------
1. Iteration statements provided
---------------------------------
First For 0
First For 1
First For 2
First For 3
Second For 1
Second For 2
Second For 4
```

Javascript

Code Segment

```javascript
console.log("--------------------------------\n");
console.log("1. Iteration statements provided\n");
console.log("--------------------------------\n");

  // In Javascript programming language for, for in, for of, while and do while
      statements are provided

  // The for statement has three control expressions and executes the loop body until
      the second controlling expression evaluates to false.

  for (let i = 0; i < 4; i++){
      console.log("For " + i);
  }

  // The for in statemnts contiunes until all proporties of the object are proccessed.
  let alphabet = { first : "A", second : "B",third : "C"};
  for (i in alphabet){
      console.log("For in " + alphabet[i] + "\n");
  }

  // The for of statment contiunes until all items of an iterable are processed
  let numbers = [10, 20,30,40,50];
  for(const i of numbers) {
     console.log("For of " + i);
  }

 // The while statement checks the control expression before executing the loop body
     and executes the loop body until the control expression evaluates to false.
  let a = 0;
  while( a < 4) {

      console.log("While " + a);
      a++;
  }

  // The do statement checks the contol expression after executing the loop body each
      time and also executes the loop body at least once.
  let b = 4;
  do {
      console.log("Do While " + b);
      b++;
  } while( b < 4);
```

Output

```
--------------------------------
1. Iteration statements provided
--------------------------------
For 0
For 1
For 2
For 3
For in A
For in B
For in C
For of 10
For of 20
For of 30
For of 40
For of 50
While 0
While 1
While 2
While 3
Do While 4
```

PHP

Code Segment

```php
echo "\n--------------------------------";
echo "\n1. Iteration statements provided";
echo "\n--------------------------------";

// In PHP programming language for, foreach, while and do while statements are provided

/* The for statement has three control expressions and executes the loop body
 until the second controlling expression evaluates to false.*/
for ($i = 0; $i < 4; $i++){

     echo "\nFor ". $i;
}

/* The foreach statment contiunes until all items of an iterable are processed.*/
$arr = array( 10, 20, 30, 40, 50);
foreach( $arr as $val ) {

    echo "\nForeach ". $val;
}
```

```php
/* The while statement checks the control expression before executing the loop body
and executes the loop body until the control expression evaluates to false.*/
$a = 0;
while( $a < 4) {

    echo "\nWhile ". $a;
    $a++;
}

/* The do statement checks the contol expression after executing the loop body
     each time and also executes the loop body at least once.*/
$b = 4;
do {

    echo("\nDo While ". $b);
    $b++;

} while( $b < 4);
```

Output

```
-------------------------------
1. Iteration statements provided
-------------------------------
For 0
For 1
For 2
For 3
Foreach 10
Foreach 20
Foreach 30
Foreach 40
Foreach 50
While 0
While 1
While 2
While 3
Do While 4
```

Python

Code Segment

```python
print("--------------------------------")
print("1. Iteration statements provided")
print("--------------------------------")

# In Python programming language for in and while statements are
    provided and both statements can be used with else clause.

#The for in statement continues until there is no iteration left
for i in range(1,5):
   print("For in" ),
   print(i)
else: #For loop can be used without this else.
    print("Inside else i = "),
    print(i)

a = 0
while ( a < 4):
    print("While "),
    print(a)
    a = a + 1
else: #While loop can be used without this else.
    print("Inside else a = "),
    print(a)
```

Output

```
--------------------------------
1. Iteration statements provided
--------------------------------
For in 1
For in 2
For in 3
For in 4
Inside else i =  4
While  0
While  1
While  2
While  3
Inside else a =  4
```

Rust

Code Segment

```rust
print!("--------------------------------\n");
print!("1. Iteration statements provided\n");
print!("--------------------------------\n");
// In Rust programming language for in, while and loop statements are provided.

//Loops in range [0,4)
for x in 0..4{
     println!("For in {}", x);
  }

let mut i = 0;
while i < 5{
    println!("While {}", i);
    i+=1;
}
//Acts like indefinite while loop.
let mut n = 0;
loop {
    if n == 5 {
      break;
    }
    println!("Loop {}", n);
    n+=1;
}
```

Output

```
--------------------------------
1. Iteration statements provided
--------------------------------
For in 0
For in 1
For in 2
For in 3
While 0
While 1
While 2
While 3
While 4
Loop 0
Loop 1
Loop 2
Loop 3
Loop 4
```

1.2 Data Structures Suitable for Iteration

C

Code Segment

```c
 printf("----------------------------------------\n");
 printf("2. Data Structures Suitable for Iteration\n");
 printf("----------------------------------------\n");

//Array and string

const int SIZE = 3;
int arr[SIZE] = { 1, 2, 3,};

for (int i = 0; i < SIZE; i++) {
  printf("%d ", arr[i]);
}

printf("\n");

char* str = "deniz";
int ch = 0;
while (str[ch] != '\0') {
    printf("%c ", str[ch]);
    ch++;
}
```

Output

```
----------------------------------------
2. Data Structures Suitable for Iteration
----------------------------------------
1 2 3
d e n i z
```

GO

Code Segment

```
fmt.Print("----------------------------------------\n")
fmt.Print("2. Data Structures Suitable for Iteration\n")
fmt.Print("----------------------------------------\n")


str := []string{"zero", "one"}
for i, s := range str {
    fmt.Println(i, s)
}

for i, ch := range "yes" {
    fmt.Printf("%#U %d\n", ch, i)
}


m := map[string]int{
"I":     1,
"am":    2,
"bored": 3,
}
for i, n := range m {
    fmt.Println(i, n)
}
```

```
channel := make(chan int)
go func(){
    channel <- 1
    channel <- 2
    channel <- 3
    channel <- 4
    close(channel)
}()
for i:= range channel {
   fmt.Println(i)
}
```

Output

```
----------------------------------------
2. Data Structures Suitable for Iteration
----------------------------------------
0 zero
1 one
U+0079 'y' 0
U+0065 'e' 1
U+0073 's' 2
I 1
am 2
bored 3
1
2
3
4
```

Javascript

Code Segment

```
console.log("----------------------------------------\n");
console.log("2. Data Structures Suitable for Iteration\n");
console.log("----------------------------------------\n");

//Iterate over an array
const arr = [5, 7, 9];
for (const i of arr) {
    console.log(i);
}

//Iterate over a string
const str = 'hey';
for (const i of str) {
    console.log(i);
}

//Iterate over a typed array
const typedArr = new Uint8Array([0x00, 0xff]);
for (const i of typedArr) {
   console.log(i);
 }

//Iterate over a map
const map = new Map([['d', 1], ['n', 2], ['z', 3]]);
for (const i of map) {
    console.log(i);
}

//Iterate over a set
const st = new Set(['s', 'e', 't']);
for (const i of st) {
   console.log(i);
}

//Iterate over an argument object
(function() {
  for (const i of arguments) {
    console.log(i);
  }
})(7, 11, 13);
```

```
//Iterate over a generator
function* generator(i) {
  yield i;
  yield i + 4;
}

const gen = generator(4);
for (const i of gen) {
  console.log(i);

}

//Iterate over an object(enumerable)
let myalphabet = { first : "A", second : "B",third : "C"};
for (i in alphabet){
    console.log("For in " + myalphabet[i] + "\n");
}
```

Output

```
2. Data Structures Suitable for Iteration
----------------------------------------
5
7
9
h
e
y
0
255
▼Array(2) ℹ
   0: "d"
   1: 1
   length: 2
  ▶__proto__: Array(0)
▼Array(2) ℹ
   0: "n"
   1: 2
   length: 2
  ▶__proto__: Array(0)
▼Array(2) ℹ
   0: "z"
   1: 3
   length: 2
  ▶__proto__: Array(0)
s
e
t
7
11
13
4
8
For in A
For in B
For in C
```

PHP

Code Segment

```php
echo "\n----------------------------------------";
echo "\n2. Data Structures Suitable for Iteration";
echo "\n----------------------------------------";

$arr2 = array( 10, 20, 30, 40, 50);
foreach( $arr2 as $vall ) {

    echo "\nForeach ". $vall;
}

//Dynamic array
foreach (array(1, 2, 3, 4, 5) as $v) {
    echo "\n$v\n";
}

//Multi dimensional array
$a = array();
  $a[0][0] = "d";
  $a[0][1] = "e";
  $a[1][0] = "n";
  $a[1][1] = "z";
```

```php
 foreach ($a as $v1) {
   foreach ($v1 as $v2) {
      echo "\n$v2\n";
    }
 }

 // Iterate over an array of arrys using list()
 $array = [
     [1, 2],
     [3, 4],
 ];

 foreach ($array as list($a, $b)) {
    echo "A: $a; B: $b\n";
}

$i = array(
"one" => 1,
"two" => 2,
"three" => 3,
);
```

```php
  foreach ($i as $key => $val) {
      echo "$key => $val\n";
  }

 for($c = 'A'; $c != 'D'; $c++) {
      echo $c.' ';
 }

 //strtotime
 for ($date = strtotime("2021-04-26"); $date < strtotime("2021-04-28"); $date = strtotime("+1 day", $date)) {
      echo date("\nY-m-d", $date);
}
```

Output

```
----------------------------------------
2. Data Structures Suitable for Iteration
----------------------------------------
Foreach 10
Foreach 20
Foreach 30
Foreach 40
Foreach 50
1

2

3

4

5

d

e

n

z
A: 1; B: 2
A: 3; B: 4
one => 1
two => 2
three => 3
A B C
2021-04-26
2021-04-27
```

Python

Code Segment

```python
print("----------------------------------------")
print("2. Data Structures Suitable for Iteration")
print("----------------------------------------")

#Dictionary
d = {"one": "1",  "two": "2",  "three": "3"}
for k in d:
    print(k) #There is no fixed order

#List
list = [8, 9, 10]

for i in list:
    print(i)

#Tuple
tuple = ("I", "am", "bored")
for i in tuple:
  print(i)

#Set
set = {"me", "myself", "I"}
for i in set:
  print(i) #There is no fixed order

#Array
arr = numpy.array([0, 1, 2])
for i in arr:
  print(i)

#String
str = "hey"
for i in range( len(str) ):
    print(str[i])
```

Output

```
----------------------------------------
2. Data Structures Suitable for Iteration
----------------------------------------
three
two
one
8
9
10
I
am
bored
me
I
myself
0
1
2
h
e
y
```

Rust

Code Segment

```
print!("--------------------------------\n");
print!("2. Data Structures Suitable for Iteration\n");
print!("--------------------------------\n");

//Vector
let v1 = vec![1,2,3];
for i in &v1 {
     print!("{} ", i);
}
println!();

//Vector using iter()
let v2 = vec![1, 2, 3];
let v2_iter = v1.iter();
for i in v2_iter {
     print!("{} ", i);
}
println!();

//Array using iter()
let arr = [1, 2, 3];
for i in arr.iter() {
    print!("{} ", i);
}
```

Output

```
----------------------------------------
2. Data Structures Suitable for Iteration
----------------------------------------
1 2 3
1 2 3
1 2 3
```

1.3 The Way the Next Item is Accessed

C

Code Segment

```c
printf("-----------------------------------\n");
printf("3. The Way the Next Item is Accessed\n");
printf("-----------------------------------\n");


const int S = 3;
int ar[S] = { 1, 2, 3,};

//Each item has the address =  start address + (i * 4).
//So, next item is accessed by this manner using pointers.
for (int i = 0; i < S; i++) {
  printf("%d ", (int)&ar[i]); //&ar[i] is the address of ar[i] (element at i'th index)
}
 printf("\n");

for (int i = 0; i < S; i++) {
  printf("%d ", *(ar + i)); // *(ar + i)) is the value of ar[i]
}
 printf("\n");
 printf("%d ", (int)ar); // ar is the poniter to the starting address of the array.
 printf("\n");
```

Output

```
-----------------------------------
3. The Way the Next Item is Accessed
-----------------------------------
-272632464 -272632460 -272632456
1 2 3
-272632464
```

GO

Code Segment

```go
fmt.Print("-------------------------------------\n")
fmt.Print("3. The Way the Next Item is Accessed \n")
fmt.Print("-------------------------------------\n")

//An array variable denotes the entire array. It is not a pointer to the first array element.
st := [3]int{2, 4, 6} //Just three integers laid out sequentially

fmt.Println(st) //Prints all of the array.

//Elemets are accessed using indexing
for i, s := range st {
    fmt.Println(i, &s) //All items are in the same memory location
}
```

Output

```
-------------------------------------
3. The Way the Next Item is Accessed
-------------------------------------
[2 4 6]
0 0xc0000b6020
1 0xc0000b6020
2 0xc0000b6020
```

Javascript

Code Segment

```javascript
console.log("-----------------------------------\n");
console.log("3. The Way the Next Item is Accessed\n");
console.log("-----------------------------------\n");

//In Javascript there are no pointers
//Elements are acessed with indexing
const ar = [5, 7, 9];
for (const i of ar) {
    console.log(i);
}

console.log(ar);
```

Output

```
-----------------------------------
3. The Way the Next Item is Accessed
-----------------------------------
5
7
9
▼Array(3) ℹ
    0: 5
    1: 7
    2: 9
    length: 3
  ▶ __proto__: Array(0)
```

PHP

Code Segment

```
echo "\n-----------------------------------";
echo "\n3. The Way the Next Item is Accessed ";
echo "\n-----------------------------------\n";

/*Array pointer is incremented by 1 to reach the next
element but the pointer is not like the pointer in C. It is more like an internal mechanism*/

$arr3 = array( 10, 20, 30, 40, 50);
foreach( $arr3 as $valll ) {

    echo " ". $valll;
}

echo "\n";
//next() can be used to go to next element
for ($i = 0; $i < 5; $i++){

     echo  next($arr3); //will start printing from 20 beacuse
    //the first time the next is called the value is 10 so it prints the next value.
     echo " ";
}
```

Output

```
-----------------------------------
3. The Way the Next Item is Accessed
-----------------------------------
 10 20 30 40 50
20 30 40 50
```

Python

Code Segment

```python
print("-----------------------------------")
print("3. The Way the Next Item is Accessed ")
print("-----------------------------------")

#There are no pointers in Python
#Elemets are accessed through indexing

list = [8, 9, 10]

#When looping through iterables iter() and next() built
    in functions are called.
for i in list:
    print(i)

#Doing manually what the above statement does
a = [8, 9, 10]

itr = iter(a)
itr
print(next(itr))
print(next(itr))
print(next(itr))
```

Output

```
-----------------------------------
3. The Way the Next Item is Accessed
-----------------------------------
8
9
10
8
9
10
```

Rust

Code Segment

```
print!("-----------------------------------\n");
print!("3. The Way the Next Item is Accessed\n");
print!("-----------------------------------\n");

//Rust uses built in iterator functions

let arr2 = [1, 2, 3];
for i in arr2.iter() { //without calling iter() this statement gives error
    print!("{} ", i); //calls next() built in function
}
```

Output

```
-----------------------------------
3. The Way the Next Item is Accessed
-----------------------------------
1 2 3
```

2.Evaluation of Languages

Best language in terms of readability and writability of iteration statements is Python because this programming language provides a variety of data structures and they are easy to implement. Also, there are iteration statements to go over these data structures easily. So, this situation gives lots of flexibility while using iterable data structures. On the other hand, C programming language was not very good in terms of iterable data structure. For example, there are not many choices other than arrays. Also, if users want to implement a data structure such as a linked list, they need to give a certain effort to complete this task because pointers are very critical. All of the other languages had more data structures available than C. Another thing that I noticed is that, in GO, only for loop exists but it can be used as a while loop. This situation increases simplicity but decreases readability.

3.Learning Strategy

First of all, I started with a skeleton program to give me an idea of what I will do to complete this task. I wrote my skeleton program in C programming language because it's the language that I have most knowledge about and also the one that I'm most comfortable with. I wrote a C program that will address all of the design issues specified in the homework description and separated all parts with meaningful comments. I also added information and clarifications on top of each part which I obtained while doing research about these parts. I wrote the C code using the Xcode development environment in MacOS. After I finished the C program I started writing

the same program in other languages: GO, Python, Rust and PHP respectively using online compilers (will be provided later) and I wrote the Javascript program in MacOS using a text editor while making the necessary syntax changes. Then, I started my report. For each part I went over all of the programs again one by one and looked for differences in each one of them and modified the programs to be able to show the important aspects. I did lots of research to be able to understand the dynamics of each language. Also, did lots of experiments with different data structures, iterators, and looping statements. Then, I added the code segments and outputs to the report.

URLs to online compilers:

GO: https://play.golang.org/
Python: https://www.tutorialspoint.com/execute_python_online.php
Rust: https://rextester.com/l/rust_online_compiler
PHP: https://rextester.com/l/php_online_compiler