Melat Ali
20 Oct

CS330: Programming Language Project (PLP)

Assignment 5: Functions and parameter passing

1. **What is the syntax for declaring a function in your language?**

   func funcname(parameters) -> return type {

   statements

   return statement}[1]

   To call the function, follow the format: funcname(parameters)

   For example:

   ```
   func multiplyandadd(no1:Int, no2:Int) -> (Int, Int) {
     no3 = no1 * no2
     no4 = no1 + no2
     return (no3, no4)}
   multiplyandadd(no1:12, no2:13)
   #each parameter needs to have its type after it
   #you need to declare the data type for each variable that's being returned
   ```

2. **Are there any rules about where the function has to be placed in your code file so that it can run?**

   No, the function can be placed anywhere in the code.

3. **Does your language support recursive functions?**

   Recursive functions are functions that call on themselves. In order to create one in Swift, we have to include a function call within the function itself:

   ```
   func recurse() {
      statements
      recurse()}
   recurse()
   ```
   [2]

[1] "Swift - Functions." *Tutorialspoint*, www.tutorialspoint.com/swift/swift_functions.htm.
[2] https://www.programiz.com/swift-programming/recursion

4. **Can functions in your language accept multiple parameters? Can they be of different data types?**

   Yes, Swift functions can accept multiple parameters and they can be of different data types.

5. **Can functions in your language return multiple values at the same time? How is that implemented?**

   Yes, Swift functions can return multiple values. In order to do this, we can include the multiple values within the same return statement separated by commas:
   return(value1, value2), and as shown in Number 1, you have to include the return types on the first line of the function declaration:

```
func multiplyandadd(no1:Int, no2:Int) -> (Int, Int) {
  no3 = no1 * no2
  no4 = no1 + no2
  return (no3, no4)}
multiplyandadd(no1:12, no2:13)
```
[3]

6. **Is your language pass-by reference or value?**

   A language is pass by value if it makes a copy of a parameter's value when that value is assigned to another parameter. Thus, if the original value changes, the second parameter remains unchanged. If a language is pass by reference, then the second parameter is assigned the first parameter's value's address, so if the original value changes, the second parameter's value also changes.
   Swift is pass by reference.[4]

7. **Are there any other aspects of functions in your language that aren't specifically asked about here, but that are important to know in order to write one? What are they?**

- Swift functions can have optional return types. This can be implemented by adding a question mark after the return parameters in the first line.
  func multiplyandadd(no1:Int, no2:Int) -> (Int, Int)? {

[3] "How to Return Multiple Values from a Function in Swift." *Simple Swift Guide*, 8 May 2020, www.simpleswiftguide.com/how-to-return-multiple-values-from-function-in-swift/.
[4] Deitel, Paul J., and Harvey M. Deitel. *Swift for Programmers*. Pearson Education, Inc., 2015.

Melat Ali
20 Oct

- **Write a function that takes in two numbers, multiplies them, and returns the output**

```
print("Takes in two numbers, multiplies them, and returns the output\n")
func multiply(no1: Int, no2: Int) -> Int {
   return no1*no2
}
print("No1 = 5")
print("No2 = 6")
print("Result: ")
print(multiply(no1: 5, no2: 6))
print("\n")
```

OUTPUT:

```
Takes in two numbers, multiplies them, and returns the output

No1 = 5
No2 = 6
Result:
30
```

- **Write a recursive function (if possible)**

```
print("Recursive Function\n")
func recursiveEX(val: Int) -> Int {
    print(val)
    while (val>10) {
      recursiveEX(val: val - 1)}
    return val}
let result1 = recursiveEX(val:10)
print(result1)
print("\n")
```

OUTPUT:

```
Recursive Function

10
10
```

- **Write a function that takes in a string (or your language's equivalent) and splits it into two strings, then returns both strings**

```
print("Takes in a string and splits it into two strings, then returns both strings\n")
func stringEX(st1: String) -> (String, String) {
    let splitarray = st1.components(separatedBy: " ")
    let newst1 = splitarray[0]
    let newst2 = splitarray[1]
    return(newst1, newst2)
    }
print("Before Split:")
print("Melat Ali")
print("After Split:")
let result = stringEX(st1:"Melat Ali")
var st1 = result.0
var st2 = result.1
print("First Name: " + st1)
print("Last Name: " + st2)
print("\n")
```

OUTPUT:

```
Takes in a string and splits it into two strings, then returns both strings

Before Split:
Melat Ali
After Split:
First Name: Melat
Last Name: Ali
```

- **Call your functions from main, and save the results of the function calls in variables.**

```
print("Call your functions from main, and save the results of the function calls in var
func main() {
    let multiplyfunction = multiply(no1: 5, no2: 6)
    print(multiplyfunction)
    let recursivefunc = recursiveEX(val:10)
    print(recursivefunc)
    let result = stringEX(st1:"Melat Ali")
    print(result)}

main()
print("\n")
```

OUTPUT:

```
Call your functions from main, and save the results of the function calls in variab

30
10
10
("Melat", "Ali")
```

Melat Ali
20 Oct

- **Write a function that tests whether your language is pass-by-reference or pass-by-value.**

```
print("Test whether your language is pass-by-reference or pass-by-value\n")

func PassbyValue(value1: String, value2: Int){
    var value1 = "changed"
    var value2 = 12}
print("Pass by Value\n")
PassbyValue(value1:"", value2:2)
print(val1)
print(val2)


func PassbyReference(_ value1: inout String, _ value2: inout Int){
    value1 = "changed"
    value2 = 12}
var val1: String = ""
var val2: Int = -1
PassbyReference(&val1, &val2)
print("Pass by Reference\n")
print(val1)
print(val2)
```

OUTPUT:

```
Test whether your language is pass-by-reference or pass-by-value

Pass by Value


0
Pass by Reference

changed
12
```

As you can see, the first function did not work, but the second function successfully changed the values. In order to pass by reference in Swift, you need to include the term inout before the variable type and add an underscore before the variable name. When calling the function, you also need to include an & before each parameter name.