Go Piscine

Go 06

Summary: THIS document is the subject for the Go 06 module of the Go Piscine @ 42Tokyo.

Contents

Chapter I

Instructions

- Only this page will serve as reference; do not trust rumors.
- Watch out! This document could potentially change up to an hour before submission.
- These exercises are carefully laid out by order of difficulty - from easiest to hardest. We `will not` take into account a successfully completed harder exercise if an easier one is not perfectly functional.
- Make sure you have the appropriate permissions on your files and directories.
- You have to follow the submission procedures for every exercise.
- Your exercises will be checked and graded by your fellow classmates.
- You cannot leave any additional file in your directory than those specified in the subject.
- Got a question? Ask your peer on the right. Otherwise, try your peer on the left.
- Your reference guide is called `Google / man / the Internet / ....`
- Examine the examples thoroughly. They could very well call for details that are not explicitly mentioned in the subject...
- If no other explicit information is displayed, you must use the latest versions of Go.

Chapter II

Exercise 00 : boolean

	Exercise 00
boolean	
Turn-in directory : *ex*00/	
Files to turn in : `*`	
Allowed packages : `github.com/42tokyo/ft, os`	
Allowed builtin functions : `None`	

Create a new directory called boolean.

- The code below must be copied into a file called `main.go` inside of the `boolean` directory.
- The necessary changes must be applied for the program to work.
- Code to be copied

```
func printStr(s string) {
        for _, r := range s {
                ft.PrintRune(r)
        }
        ft.PrintRune(’\n’)
}

func isEven(nbr int) boolean {
        if even(nbr) == 1 {
                return yes
        } else {
                return no
        }
}

func main() {
        if isEven(lengthOfArg) == 1 {
                printStr(EvenMsg)
        } else {
                printStr(OddMsg)
        }
}
```

- Usage

```
$ go run . "not" "odd"
I have an even number of arguments
$ go run . "not even"
I have an odd number of arguments
```

Chapter III

Exercise 01 : point

	Exercise 01
point	
Turn-in directory : *ex*01/	
Files to turn in : `*`	
Allowed packages : `github.com/42tokyo/ft, fmt`	
Allowed builtin functions : `None`	

Create a new directory called point.

- The code below must be copied into a file called `main.go` inside the `point` directory.
- The necessary changes must be applied so that the program works.
- Code to be copied

```
func setPoint(ptr *point) {
        ptr.x = 42
        ptr.y = 21
}

func main() {
        points := &point{}

        setPoint(points)

        fmt.Printf("x = %d, y = %d\n",points.x, points.y)
}
```

- Usage

```
$ go run .
x = 42, y = 21
$
```

Chapter IV

Exercise 02 : displayfile

Exercise 02
displayfile
Turn-in directory : *ex*02/
Files to turn in : `*`
Allowed packages : `github.com/42tokyo/ft, os`
Allowed builtin functions : `None`

Write a program that displays, on the standard output, the content of a file given as argument.

- Usage

```
$ go run .
File name missing
$ echo "Ultimate Question of Life, the Universe, and Everything" > 42.txt
$ go run . 42.txt main.go
Too many arguments
$ go run . 42.txt
Ultimate Question of Life, the Universe, and Everything
$
```

Chapter V

Exercise 03 : cat

Exercise 03
cat
Turn-in directory : *ex*03/
Files to turn in : `*`
Allowed packages : `github.com/42tokyo/ft, os`
Allowed builtin functions : `None`

Write a program that behaves like a simplified cat command.

- The options do not have to be handled.
- If the program is called without arguments it should take the standard input (stdin) and print it back on the standard output (stdout).
- Usage

```
$ echo "Born2Code" > quote.txt
$ cat <<EOF> 42.txt
"Ultimate Question of Life, the Universe, and Everything"
EOF
$ go run . abc
ERROR: open abc: no such file or directory
exit status 1
$ go run . quote.txt
Born2Code
$ go run . quote.txt abc
Born2Code
ERROR: abc: No such file or directory
$ cat quote.txt | ./cat
Born2Code
$ go run .
Hello
Hello
^C
$ go run . quote.txt alan_turing.txt
Born2Code
"Ultimate Question of Life, the Universe, and Everything"
$
```

Chapter VI

Exercise 04 : ztail

	Exercise 04
ztail	
Turn-in directory : *ex*04/	
Files to turn in : `*`	
Allowed packages : `github.com/42tokyo/ft, os`	
Allowed builtin functions : `None`	

Write a program that behaves like a simplified `tail` command that takes at least one file as an argument.

- The only option to be handled is `-c` and will be used in all the tests as the first argument, with positive values.
- Handle the errors by returning a non-zero exit status but process all the files.
- If several files are given, print a newline and the file name between each one of them (see below).
- Usage
- If file1.txt & file2.txt contains :

```
abcdefghijklmnopqrstuvwxyz
```

- Normal cases :

```
$ go run . -c 4 file1.txt
xyz
$ go run . -c 4 file1.txt file2.txt
==> file1.txt <==
xyz

==> file2.txt <==
xyz
$
```

- Error cases :

```
$ go run . -c 4 file1.txt nonexisting1.txt file2.txt nonexisting2.txt
==> file1.txt <==
xyz
open nonexisting1.txt: no such file or directory

==> file2.txt <==
xyz
open nonexisting2.txt: no such file or directory
$ echo $?
1
$
```