← APG1 Bootcamp. Day08

Revie

Submit the project

Finish project

Survival camp

Show all

APG1 Bootcamp. Day00	100%
APG1 Bootcamp. Day01	100%
APG1 Bootcamp. Day02	100%
APG1 Bootcamp. Day03	0%

APG1 Bootcamp. Day06

APG1 Bootcamp. Day07

APG1 Bootcamp. Day08

APG1 Bootcamp. Day09

APG1 Bootcamp. Team01

Private Git project

ssh://git@repos-ssh.21-school.ru:2289/students/Go_Day08.ID_376231/mlarra_student...

Copy link Open

Task

Day 08 - Go Boot camp

Adventure, Danger and Cocoa

Contents

1. Chapter I
   1.1. General rules
2. Chapter II
   2.1. Rules of the day
3. Chapter III
   3.1. Intro
4. Chapter IV
   4.1. Exercise 00: Arithmetic
5. Chapter V
   5.1. Exercise 01: Botany
6. Chapter VI
   6.1. Exercise 02: Hot Chocolate

Chapter I

General rules

- Your programs should not quit unexpectedly (giving an error on a valid input). If this happens, your project will be considered non functional and will receive a 0 during the evaluation.
- We encourage you to create test programs for your project even though this work won't have to be submitted and won't be graded. It will give you a chance to easily test your work and your peers' work. You will find those tests especially useful during your defence. Indeed, during defence, you are free to use your tests and/or the tests of the peer you are evaluating.
- Submit your work to your assigned git repository. Only the work in the git repository will be graded.
- If your code is using external dependencies, it should use Go Modules for managing them

Chapter II

Rules of the day

- You should only turn in `*.go` files and (in case of external dependencies) `go.mod + go.sum`
- Your code for this task should be buildable with just `go build`

Chapter III

Intro

People tend to say that one of the main differences between Go and C is a pointer safety. That's partially true, and Go will try really hard to not let you shoot yourself in a foot when tangoing with pointers. But we are already far enough in a jungle to be able to play with danger just a bit, don't you think?

Chapter IV

Exercise 00: Arithmetic

Here in a jungle you can find some weird creatures that you need to treat in an unusual way. For this task you need to write a function `getElement(arr []int, idx int) (int, error)` that accepts and an index and gives you back the element with this index. Seems easy enough, eh? But here's one condition - you can't use lookup by this index (like `arr[idx]`), the only lookup allowed is a first element (`arr[0]`). You may need to remember some C to complete this exercise.

In case of any non-valid input (empty slice, negative index, index is out of bounds) the function should return an error with a text explanation of a problem.

Chapter V

Exercise 01: Botany

You're in luck! You've found some pretty rare plants:

```
type UnknownPlant struct {
    FlowerType  string
    LeafType    string
    Color       int `color_scheme:"rgb"`
}

type AnotherUnknownPlant struct {
    FlowerColor int
    LeafType    string
```

```
    Height      int `unit:"inches"`
}
```

Well, yeah, current representation is a bit of a mess. Your goal would be to write a single function `describePlant` that will accept any kind of plant (yes, it should work with structures of different types) and then print all fields as key-value pairs, separated by comma (mind the tags), like this:

```
FlowerColor:10
LeafType:lanceolate
Height(unit=inches):15
```

Chapter VI

Exercise 02: Hot Chocolate

Okay, now it's time to relax and have some cocoa. Cocoa usually comes in packages (see provided zip archive). You don't need to modify the code in packaged files in any way, the only thing you need to do is write a code (including cocoa files as part of your project) that will create default empty Mac OS GUI window (size 300x200) with title "School 21". It's easier than you think!