HomeWork C Language

Ali ZAINOUL
ali.zainoul.az@gmail.com

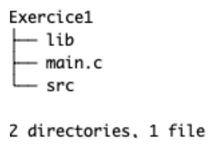
March 25, 2023

General informations

Particular attention will be paid to:

- your understanding of the question and what is being asked
- your code quality and **compilation of your program**.
- work must be returned before 24 april at midnight.
- Exceptation: 5 directories/files for the 5 exercices, zipped into one zip, named like this: **firstName_lastName**_C.zip, every directory/file must contain the source files (.c) and/or the header files (.h).

- Suppose we want to create a directory for Exercice1:
- First, let's create Exercice1 directory, getting into it, create inside of it src and lib directories, then create a file named main.c ; the following command do all of this in a single line. Open a command prompt and type in:
  mkdir Exercice1 && cd Exercice1 && mkdir src lib && touch main.c
  This tree illustrates the excpected output:



```
Exercice1
├── lib
├── main.c
└── src

2 directories, 1 file
```

Figure 1: tree

- **Compilation of a single file:** Assume we are into Exercice1 directory, then open a command prompt and type in:
  gcc -o myExecutable main.c && ./myExecutable

- **Compilation of multiple files:** Assume we are into Exercice1 directory, and there is several files to be executed before the main; then open a command prompt and type in:
  gcc -o myExecutable src/*.c main.c && ./myExecutable

  Where: ***.c** : represents all the src files .c contained in the src directory.

We recall that:

- pwd: prints the actual directory where you are
- cd myDirectory: change directory into myDirectory
- mkdir myDirectory: create a directory named myDirectory
- touch myFile: creating a file named myFile
- ls: listing all the files and directories present into your actual directory (you can know where you are by typing pwd)
- &&: stands for the binary operator applied into several commands. E.g: command1 && command2 is equivalent to: command 1 then command2.

Example: helloWorld.c source code

```
#include <stdio.h>

int main() {
    printf("Hello, world!\n");
    return 0;
}
```

Compile with: gcc -o myExe helloWorld.c && ./myExe

C Language Exercises

Exercise 1: Structures

Create a structure called ”Employee” that contains the following information:

- Name (string)
- ID number (integer)
- Salary (float)
- Address (string)

- Date of Birth (string)

Write a program that asks the user to enter information for 5 employees and stores it in an array of Employee structures. Then, write a function that takes the array of Employee structures as an argument and sorts it by ID number in ascending order. Finally, print out the information for each employee, in order of increasing ID number.

Exercise 2: Arrays

Write a program that generates an array of 10,000 random integers between 1 and 100 (inclusive). Then, write a function that takes the array of integers as an argument and finds the two integers with the smallest difference between them. Finally, print out the two integers with their indices in the array.

Exercise 3: Loops

Write a program that asks the user to enter 5 integers and stores them in an array. Then, use a loop to find the sum of all the integers in the array and print out the result.

Exercise 4: Pointers

Write a program that initializes an integer variable ”num” to 10 and a pointer ”ptr” to point to it. Use the pointer to change the value of ”num” to 20. Print out the value of ”num” before and after the change.

Exercise 5: Pointers to Structures

Create a structure called ”Person” that contains the following information:

- Name (string)
- Age (integer)
- Address (string)

Write a program that creates a pointer to a Person structure and allocates memory for it using the ‘malloc()‘ function. Then, prompt the user to enter information for the Person and store it in the allocated memory using the pointer. Finally, print out the information for the Person using the pointer.

Good luck ♥