Lab Serie No. 0: Reminder on structures

Structures, often referred to as **structs**, offer a mechanism to **group several related variables** into one place. Each variable in the structure is known as a member of the structure. Unlike an array, a structure can contain **many different data types** (int, float, char, etc.).

You can create a structure by using the **struct** keyword and declare each of its members inside curly braces, Here's a simple example:

```
struct StructureName {
Type1 member1;
Type2 member2;
        ⋮
TypeN ChampN;
} ;
struct StructureName VariableName;
```

```
// Example
struct Student {// Structure declaration
  int age // Member1 (int variable)
  char name[50]; // Member2 (char variable)
}; // End the structure with a semicolon

struct Student std; // Std is a variable
```

*To avoid repeating the **struct** keyword in each variable declaration, you can use a keyword **typedef**. **typedef** allows us to create an alias for a data type, including a structure, so that you can use this alias instead of the struct keyword when declaring variables. We add the following line to the previous code:*

```
typedef struct Student Student;
```

The variable std is hence declared as follow:

```
Student std;
```

you can combine the typedef declaration with the structure definition in a single line:

```
// Example
Typedef struct {
  int id_number
  char name[50];
} Student;

Student std;
```

You can access the members of a structure using the dot (.) operator:

```
scanf("%d", &std.age); // Read the age from the user and assign it to the 'age 'member
strcpy(std.name, "Léa");// Access and modify the 'name' member of the student structure
printf("Name: %s\n", student1.name); // Display the 'name' member
printf("Age: %d\n", student1.age); // Display the 'age' member
```

Exercise 01

Write a C program that reads a set of N cities, each characterized by a name and its population number, sorts them, and displays them in ascending order of population.

Exercise 02

Write a C program that creates an array of N students, where each student is defined by his full name and average score. Then, update these averages by adding bonus points:

- 1 point for students with a score strictly below 10.
- 0.5 points for students with a score between 10 and 15, inclusive.

Exercise 03

In a library of books, each book is characterized by an **ISBN** code (int), its **category** (Algo, Archi, Algb,SE), its **title**, its **author** (First name and last name), the **publication year**, its state (**available/unvailable**) and the **number of copies**.

- Declare the necessary types, then write a subprogram that creates the set of **N** Books.
- Write a subprogram that displays the code, the title and the author of books whose year of publication is more recent than that given by the user.
- Write a subprogram that counts the number of books available in a given category.

Here is an example of declaring enumeration types :

```
// declaring enumeration types on days of the week.
enum    Days { MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY,
               SUNDAY };
```