Practical work nr. 8 – Structs

Tópicos

- Data structures (**struct**s)

Exercises

1. Write a program that computes the Euclidean distance between points. In particular:
    a. Define a data structure, named **`Point2D`**, to represent a 2-dimensional point with real coordinates **`x`** and **`y`**.
    b. Write a function that asks the user for the two coordinates of a point.
    c. Write a function that prints the contents of a point in the format **`(+#.###, +#.###)`**. Note that **`+`** represents here a sign (**`+`** or **`-`**) and **`#`** represents here a digit (**`0`** to **`9`**).
    d. Write a function that computes the Euclidean distance between two points.
    e. Implement a program that asks the user for points until a point with coordinates **`(0,0)`** is entered. The program should count the number of points that were entered (excluding the **`(0,0)`** point) and should compute the sum of their distances to the origin. The program should also determine which point is further away from the origin. The program should interact with the user as follows (try to respect how things are formatted; use the following example as a guide):

```
One point? 1 0
One point? 0 -4
One point? -2 0
One point? 3 4
One point? 0 0
The sum of the distances of the 4 points to the origin is 12.000
The point further away from the origin is: (+3.000,+4.000)
```

2. Write a program that is able to perform complex number arithmetic. In particular:
    a. Define a data structure, named **`Complex`**, to represent a complex number as a pair of two real numbers, **`re`** and **`im`**, the first holding the real part of the complex number, and the second holding the imaginary part.
    b. Write a function that asks the user for a complex number.
    c. Write a function that prints a complex number according to the following format: **`+#.###+#.###i`**. Note that **`+`** represents here a sign (**`+`** or **`-`**) and **`#`** represents here a digit (**`0`** to **`9`**).
    d. Write a function for each of the following arithmetic operations:
        i. Addition: `(a+bi)+(c+di) = (a+c)+(b+d)i`
        ii. Subtraction: `(a+bi)-(c+di) = (a-c)+(b-d)i`
        iii. Multiplication: `(a+bi)*(c+di)=(ac-bd)+(ad+bc)i`
        iv. Division:

```
(a+bi)/(c+di)=((ac+bd)/(c^2+d^2))+((bc-ad)/(c^2+d^2))i
```

The program should ask the user for one arithmetic operation (**`+`**, **`-`**, **`*`**, or **`/`**) and two complex operands and it should print the result of the operation. It should repeat this until the user enter an invalid arithmetic operation. The interaction model should be the following:

```
Operation: +
Complex number #1 ? 1 1
Complex number #2 ? 2 2
(+1.000+1.000i) + (+2.000+2.000i) = 3.000+3.000i

Operation: -
Complex number #1 ? 50 50
Complex number #2 ? 100 -100
(50.000+50.000i) - (100.000-100.000i) = -50.000+150.000i

Operação: =
Illegal operation. Good bye!
```

3. Write a program that stores some information about when a student is inside the DETI building. To that end, start by implementing:
    a. A data structure, named **`Time`**, that has three integer fields, one for the hour, `[0, 23]`, another for the minutes, `[0, 59]`, and another for the seconds, `[0, 59]`.
    b. A function that prints a time in the format **`HH:MM:SS`**.
    c. A function that asks the user for a valid time; it should keep asking for a time until a valid one is entered.
    d. A data structure, named **`Student`**, to store information about a student. It should contain an integer field for the student number and a string field for the student name.
    e. A function that asks the user for the information about a student.

    The main program should ask information about a student, then ask the time the student entered the DETI building, and then it should ask the time the student left the DETI building. It should then print how much time the student was inside the DETI building, or the text **`Impossible`** if the second time is before the first one.