Algoritmos y Estructuras de Datos II

Laboratorio 2025 - 1er Parcial

Tema C: Comisiones 3 y 4 - 1er turno - DNI impar

Requerimientos:

1. **Debe compilar**. Si no compila, no se aprueba el ejercicio.
2. **Debe pasar los tests**. Si no pasa los tests, no se aprueba el ejercicio.
3. **No debe usar break, ni continue, ni goto.** Baja nota.
4. **No debe usar return a la mitad de una función.** Baja nota.
5. **El código debe ser prolijo y comprensible, indentado y comentado.** Si no, baja nota.

Ejercicio 1: Búsqueda binaria básica

Dados un arreglo de enteros ordenados menor a mayor y un número entero x, devolver la posición de x dentro del arreglo. Si no se encuentra, devolver -1.

Ejercicio 1.1: Implementar en el archivo **`binary_search.c`** un algoritmo que resuelva el problema. El algoritmo **debe usar la estrategia divide y vencerás para tener complejidad O(log n)**.

Compilar y testear con los comandos:

```
$ gcc -Wall -Wextra -pedantic -std=c99 binary_search.c tests.c -o tests
$ ./tests
```

Ejercicio 1.2: Inventar y agregar en **tests.c** cinco (5) nuevos casos de test.

Ayudas:

- Mirar los tests para entender los casos particulares.
- Este es el algoritmo en el lenguaje del teórico/práctico:

```
{ PRE: "el arreglo 'a' está ordenado" }
fun binary_search(in a: array[1..n] of int, x: int) ret i : int
    i := binary_search_rec(a, x, 1, n)
end fun

fun binary_search_rec(in a: array[1..n] of int, x, lft, rgt: int) ret i : int
    var mid: int
```

```
    if lft > rgt →
        i := -1
    [] lft <= rgt →
        mid := (lft + rgt) / 2
        if x < a[mid] →
            // buscar hacia la izquierda
            i := binary_search_rec(a, x, lft, mid-1)
        [] x = a[mid] →
            // lo encontramos!
            i := mid
        [] x > a[mid] →
            // buscar hacia la derecha
            i := binary_search_rec(a, x, mid+1, rgt)
        fi
    fi
end fun
```

Ejercicio 2: Más cálculos sobre datos climáticos

Como en el laboratorio 3, en el archivo **`input/weather_cordoba.in`** se proveen datos climáticos históricos de Córdoba para los años comprendidos entre 1980 y 2016. Cada línea del archivo contiene números enteros con los datos de un día, con el siguiente formato:

`<año> <mes> <día> <t_media> <t_max> <t_min> <pres> <h> <prec>`

Por ejemplo la línea:

`1982 2 8 200 256 168 10148 77 0`

indica que el 8 de febrero de 1982 se dieron las siguientes mediciones:

- temperatura media t_media: 20.0 grados
- temperatura máxima t_max: 25.6 grados
- temperatura mínima t_min: 16.8 grados
- presión atmosférica pres: 1014.8 hectopascales
- humedad h: 77%
- precipitaciones: 0 milímetros

Por simplicidad, se incluyen datos solamente de los días 1 a 28 de cada mes.

Ejercicio 2.1: Implementar en **`queries.c`** la función **`year_min_moisture()`** que, dados los datos climáticos y un año, calcula **la humedad mínima que hubo en un día de ese año**.

Compilar y testear con los comandos:

```
$ gcc -Wall -Wextra -pedantic -std=c99 weather.c weather_table.c queries.c tests.c -o tests
$ ./tests
```