

Algoritmos y Estructuras de Datos II

Laboratorio 2025 - 1er Parcial

Tema B: Comisiones 1 y 2 - 2do turno - DNI par

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: Encontrando el primer cero

Dado un arreglo de la forma $[1, 1, \dots, 1, 0, 0, \dots, 0]$, devolver la posición del primer elemento que es un cero (0). Si no hay ningún cero, devolver -1.

Ejercicio 1.1: Implementar en el archivo `first_zero.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 first_zero.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' es de la forma [1, 1, ... , 1, 0, 0, ..., 0]" }
fun first_zero(in a: array[1..n] of int) ret result : int
  result := first_zero_rec(a, 1, n)
end fun

fun first_zero_rec(in a: array[1..n] of int, lft, rgt: int) ret result : int
  var mid: int

  if lft = rgt →
```

```

        // casos base
        if a[lft] = 0 →
            result := lft
        [] a[lft] = 1 →
            result := -1
        fi
    [] lft < rgt →
        mid := (lft + rgt) / 2
        if a[mid] = 0 →
            // buscar hacia la izquierda incluyendo mid
            result := first_zero_rec(a, lft, mid)
        [] a[mid] = 1 →
            // buscar hacia la derecha (sin mid)
            result := first_zero_rec(a, 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_max_temp()` que, dados los datos climáticos y un año, calcula **la temperatura máxima 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

```

