

Algoritmos y Estructuras de Datos II

Laboratorio 2025 - 1er Parcial

Tema A: Comisiones 1 y 2 - 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: Armandando la bandera de Córdoba

Dado un arreglo cuyos elementos son tres colores posibles RED, WHITE y BLUE, **ordenarlos** de manera que queden todos los RED primero, después todos los WHITE y al final todos los BLUE, formando la bandera de Córdoba:



Ejercicio 1.1: Implementar en el archivo `flag_sort.c` un algoritmo que resuelva el problema. El algoritmo **debe tener complejidad lineal (o sea $O(n)$)** y **debe modificar el arreglo sólo a través de la operación swap**.

Compilar y testear con los comandos:

```
$ gcc -Wall -Wextra -pedantic -std=c99 flag_sort.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:

```

proc flag_sort(in/out a: array[1..n] of Color)
  var i, j, k: nat
  i := 1
  j := 1
  k := n

  // invariante:
  // - 0 <= i <= j <= k < n
  // - posiciones [0,i) es todo RED
  // - posiciones [i,j) es todo WHITE
  // - posiciones [j,k] está desordenado
  // - posiciones [k+1,length) es todo BLUE
  // función de cota: k - j
  do j <= k →
    if a[j] = RED →
      // poner el RED al último de los RED
      swap(a, i, j)
      i = i + 1
      j = j + 1
    [] a[j] = BLUE →
      // poner el BLUE al principio de los BLUE
      swap(a, j, k)
      k = k - 1
    [] a[j] = WHITE →
      // dejar el WHITE al final de los WHITE
      j = j + 1
    fi
  od
end proc

```

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_rainfall()` que, dados los datos climáticos y un año, calcula **el total de precipitaciones para 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
```