

Sesión 08: *Structs* y cadenas enlazadas (incompleto)

Hoja de problemas


Programación para Sistemas


Ángel Herranz

aherranz@fi.upm.es

Universidad Politécnica de Madrid

2019-2020

 **Ejercicio 1.** Repasa las transparencias de clase. El trabajo con *structs* es muy natural, gran parte de su sintaxis es idéntica a la de otros lenguajes de programación y otra parte es muy parecida. Sin embargo, la memoria dinámica, especialmente cuando se viene de lenguajes con recolección automática de basura, se hace muy complicado. De nuevo, entender cada transparencia se hace fundamental.

 **Ejercicio 2.** Utilizando la definición de **struct** rectángulo, tienes que escribir un programa que lea rectángulos de la entrada estándar, calcule su área y los imprima en orden, de mayor área a menor área.

La entrada estándar tendrá el siguiente formato:

- Un entero n que indica el número de rectángulos que habrá que leer. El dato n estará entre 1 y 1000.
- n filas con cuatro enteros A_x , A_y , B_x , B_y cada una que indican los puntos (A_x, A_y) y (B_x, B_y) que definen el rectángulo.

La salida estándar tiene que sacar n filas, cada una con un rectángulo, estando las filas ordenadas de menor a mayor área.

Veamos un ejemplo de entrada:


```
2
0 0 1 1
-1 -2 3 1
```

Y su correspondiente salida:

```
-1 -2 3 1
0 0 1 1
```

►► **Ejercicio 3.** Siguiendo las indicaciones del ejercicio sobre pilas de la sesión anterior, tu

trabajo es elaborar una implementación basada en cadenas enlazadas.

 **Ejercicio 4.** Lo cierto es que las pilas hacen un uso muy restringido de las cadenas enlazadas. Por ello, en este ejercicio, te proponemos elaborar un tipo abstracto de datos *listas*. Las funciones podrían ser algo como:

- crear_vacía
- insertar_por_el_principio
- insertar_por_el_final
- insertar_por_posición
- longitud
- primero
- último
- iésimo
- borrar_primerio
- borrar_último
- borrar_por_posición