Córdoba 30 de Mayo del 2024

Segundo Parcial de Laboratorio

Algoritmos y Estructura de Datos II

TEMA A: La iniciativa

Ejercicio

El mismo grupo de gente que estuvo jugando rol en el primer parcial está pensando en implementar un sistema de batalla por turnos para su personaje y necesita priorizar en qué orden cada uno de ellos realizará una acción, lo que en jerga de juegos de rol se conoce como “Cálculo de iniciativa”.

Para ello cuenta con la misma estructura Character del primer parcial que representa a cada personaje con sus atributos, de los cuales son relevantes en nuestro problema: el tipo de personaje **`ctype`** (***`agile`***, ***`physical`***, ***`magic`*** o ***`tank`***), su agilidad (**`agility`**) y si está vivo o muerto (**`alive`**). Con estos datos la iniciativa de cada personaje en el sistema de turnos se calcula como:

```
Iniciative = baseInitiative * modificador * isAlive
```

- **`baseInitiative -`** es la agilidad del personaje.
- **`isAlive`** - es ***`0`*** si el personaje está muerto y ***`1`*** si está vivo.
- **`Modificador -`** es ***`+50%`*** si el personaje es de tipo agile (multiplica por **`1.5`**) y ***`-20%`*** si es tank (multiplica por **`0.8`**)

Por ejemplo si un jugador:

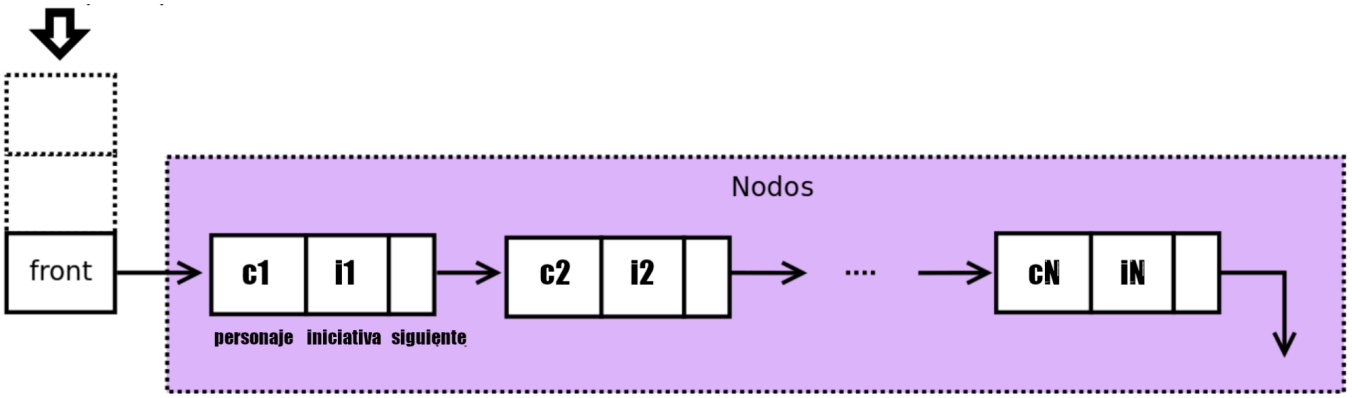
- `[Está` **`vivo`**`, tiene agilidad de` **`10`** `y es un “`**`agile`**`”]` **`= 10 * 1.5 * 1 = 15.0`**
- `[Está` **`muerto`**`, tiene agilidad de` **`8`** `y es un “`**`tank`**`”]` **`= 8 * 0.8 * 0 = 0.0`**

Así **`iniciative`** tiene valores entre ***`[0, 15]`*** y a más iniciativa/prioridad (juega antes) el personaje.

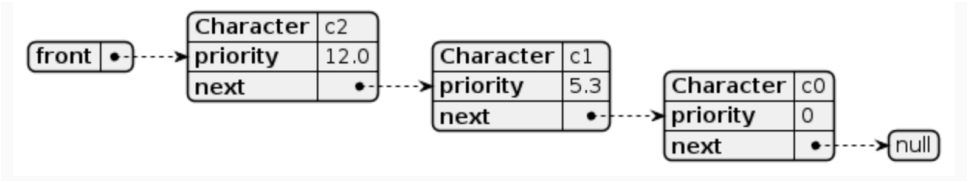
Para acceder a los atributos de los personajes se disponen de las funciones ***`character_agility()`***, ***`character_ctype()`*** y ***`character_is_alive().`***

El enum ***`charttype_t`*** contiene los tipos de los personajes.

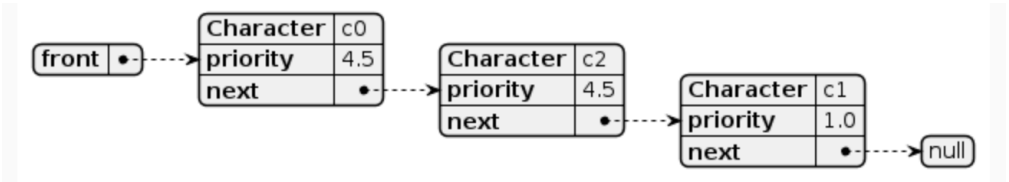
Con esta información se pide Implementar el TAD ***pqueue*** que representa una cola de prioridades. Una cola de prioridades (**pqueue**) es un tipo especial de cola en la que cada elemento está asociado con una prioridad asignada. En una cola de prioridades un elemento con mayor prioridad será desencolado antes que un elemento de menor prioridad. Sin embargo, si dos elementos tienen la misma prioridad, se desencolarán siguiendo el orden de la cola. En este caso vamos a implementar **pqueue** usando lista enlazadas y una estructura principal:



En la representación, **`c1`** es el primer *personaje* (**`Character`**) de la cola y **`i1`** (primera iniciativa) tiene la mayor prioridad. En realidad la propiedad fundamental es que **`i1`** ≥ **`i2`** ≥ … ≥ **`iN`** (notar que aquí la prioridad representada con i1 es la mayor prioridad, y números más bajos establecen prioridades más bajas). Además, si siempre se usa la misma prioridad, el TAD ***pqueue*** debe comportarse exactamente igual a una cola común (FIFO: first input, first output). Algunos ejemplos:



En la ***pqueue*** de arriba, el próximo elemento a desencolar es el **c2** ya que tiene prioridad **12.0** (la más prioritaria).



En este ejemplo el próximo elemento a desencolar es **c0** ya que aunque tiene la misma prioridad que **c2**, fue agregado primero el **c0** a la cola. **Lo importante es mantener la estructura de nodos bien ordenada para desencolar el elemento correcto**.

El TAD ***pqueue*** tiene la siguiente interfaz

Función	Descripción
`pqueue pqueue_empty(void)`	Crea una cola de prioridades/iniciativa vacía
`pqueue pqueue_enqueue(pqueue q, Character c);`	Inserta un personaje a la cola calculando su prioridad.
`bool pqueue_is_empty(pqueue q);`	Indica si la cola de prioridades está vacía
`unsigned int pqueue_size(pqueue q)`	Obtiene el tamaño de la cola de prioridades
`Character pqueue_peek(pqueue q)`	Obtiene el Character con mayor prioridad
`float pqueue_peek_priority(pqueue q)`	Obtiene el valor de la prioridad del elemento con mayor prioridad.
`pqueue pqueue_copy(pqueue q)`	Crea una copia de la cola de prioridades q.
`pqueue pqueue_dequeue(pqueue q)`	Quita un elemento con mayor prioridad más antiguo de la cola
`pqueue pqueue_destroy(pqueue q)`	Destruye una instancia del TAD Cola de prioridades

En **`pqueue.c`** se da una implementación incompleta del TAD ***pqueue*** que deben completar **siguiendo la representación explicada anteriormente.** Además se debe asegurar que las funciónes **`pqueue_size()`, `pqueue_peek_priority()`** y **`pqueue_dequeue()`** sean de orden constante (***O(1)***). Por las dudas se aclara que no es necesario que la función **`pqueue_enqueue()`** sea de orden constante ya que puede ser muy complicado lograrlo para la representación que se utiliza y no recomendamos intentarlo.

Para verificar que la implementación del TAD funciona correctamente, se provee el programa (**main.c**) que toma como argumento de entrada el nombre del archivo cuyo contenido sigue el siguiente formato:

Lo que quiere decir que cada fila tiene el siguiente formato:

```
[<a | p | t | m> <g | e>] <Name> life: <[0-100]>, strength: <[0-10]>, agility: <[0-10]>
```

Ejemplo:


```
[p e] Nadia life: 39, strength: 5, agility: 3
```

Donde:

- **El primer carácter representa el tipo de personaje:** Se representa con un carácter que es uno entre `'a'`, `'p'`, `'t'`, `'m'` que denotan el personaje de tipo ágil, personaje de fuerza física, personaje resistente (tanque) y personaje mágico respectivamente.
- **El segundo la alineación:** Es un carácter que representa el alineamiento del personaje, ya sea "bueno" o "malo". Debe ser la letra `'g'` (bueno) o `'e'` (malo).
- Luego *5* caracteres del nombre.
- Luego la vida (`life`) [0, 100]. Recordar que un personaje con vida 0, no está vivo.
- Finalmente la fuerza (`strenght`) y la agilidad (`agility`) [1-10].

Consideraciones: El archivo de entrada representa una lista de personajes con sus características. Entonces el programa lee el archivo, carga los datos en el TAD ***pqueue*** y finalmente muestra por pantalla la cola de prioridades/iniciativa de los personajes con los atributos relevantes.

El programa resultante no debe dejar *memory leaks ni lecturas/escrituras inválidas en ninguno de los archivos de ejemplo*.

Una vez compilado el programa puede probarse ejecutando:

```
$ ./show_initiative inputs/character9.in
```

Obteniendo como resultado:

```
=== CHARACTERS INITIATIVE (9) ===
[12.0000] -> Character name: Lenny, Alive? Yes, ctype: Agile, agility: 8
[9.0000] -> Character name: Eddie, Alive? Yes, ctype: Magic, agility: 9
[7.5000] -> Character name: Alice, Alive? Yes, ctype: Agile, agility: 5
[6.0000] -> Character name: Mario, Alive? Yes, ctype: Physical, agility: 6
[4.8000] -> Character name: Bella, Alive? Yes, ctype: Tank, agility: 6
[3.2000] -> Character name: Oscar, Alive? Yes, ctype: Tank, agility: 4
[3.0000] -> Character name: Nadia, Alive? Yes, ctype: Physical, agility: 3
[2.0000] -> Character name: Eliza, Alive? Yes, ctype: Magic, agility: 2
[0.0000] -> Character name: Conan, Alive? No, ctype: Tank, agility: 6
```

Otro ejemplo de ejecución:

```
$ ./show_initiative inputs/character3.in
=== CHARACTERS INITIATIVE (3) ===
[3.2000] -> Character name: Nadia, Alive? Yes, ctype: Tank, agility: 4
```


```
[0.0000] -> Character name: Oscar, Alive? No, ctype: Agile, agility: 4
[0.0000] -> Character name: Conan, Alive? No, ctype: Agile, agility: 10
```

Consideraciones:

- Se provee el archivo **`Makefile`** para facilitar la compilación.
- Los personajes (**Character**) también hay que copiarlos y eliminarlos, para eso se ofrece las funciones **`character_copy`** y **`character_destroy`**.
- Revisar todas las funciones **`character_*`** para acceder a los atributos de los personajes.
- Se recomienda usar las herramientas **`valgrind`** y **`gdb`**. ☣️
- Se ofrece una carpeta **`outputs`** que tiene las salidas esperadas para ***make test*** y los ***valgrind*** 🛟
- Si el programa no compila, no se aprueba el parcial.
- Los *memory leaks* bajan puntos.
- Entregar código muy improlijo puede restar puntos
- Si **`pqueue_size()`** no es de orden constante baja muchísimos puntos ☣️
- Para promocionar **se debe** hacer una invariante que chequee la propiedad fundamental de la representación de la cola de prioridades.
- **No modificar los .h ni el main.c** ☣️