FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Trabajo Práctico 1 — Hospital Pokemon

[7541/9515] Algoritmos y Programación II

Segundo cuatrimestre de 2021

Alumna: CONVERSO, Lara Daniela

Número de padrón: 107632

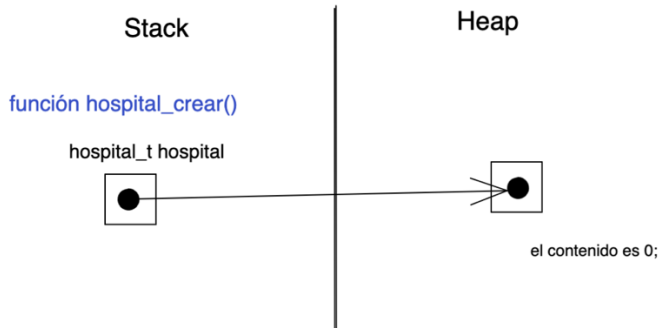
Email: lconverso@fi.uba.ar

Detalles de implementación

1. Detalles de la función hospital_crear:

   En esta función básicamente, se busca crear un hospital, siendo inicializado con malloc para reservar memoria en el heap, y que el contenido dentro sea 0, también se verifica que este hospital creado no sea NULL, de lo contrario no podríamos "operar" dentro de el.

   Por otro lado, inicializo la cantidad de entrenadores, la cantidad de pokemones y el vector correspondiente a los pokemones como nulo, ya que este se encuentra vacío.



   Interpretación grafica de la creación del hospital.

2. Detalle de la función hospital_leer_archivo

   En esta función, se realiza la lectura del archivo que es pasado por parámetro; cuando trabajamos con archivos hay 3 cosas importantes para verificar, que el archivo no sea NULL, que el archivo sea abierto correctamente y que no hay que olvidar de cerrarlo en caso de error.

   Como el contenido del archivo que recibo tiene un formato especial, para obtenerlo del archivo creé una función llamada leer_linea, en la que leo una línea del archivo con la función fgets, y la almaceno en una variable auxiliar, luego debo pasar la línea obtenida por un Split, función que se encarga de sacarle el formato al contenido del archivo. El Split retorna un vector en el que cada elemento contiene un string.

   Este procedimiento se repetirá hasta que la línea sea nula, o se encuentre con algún error, a parte en cada iteración que relizo por línea suma un entrenador a la cantidad de entrenadores en el hospital. (mi interpretación: cada línea es un nuevo entrenador).

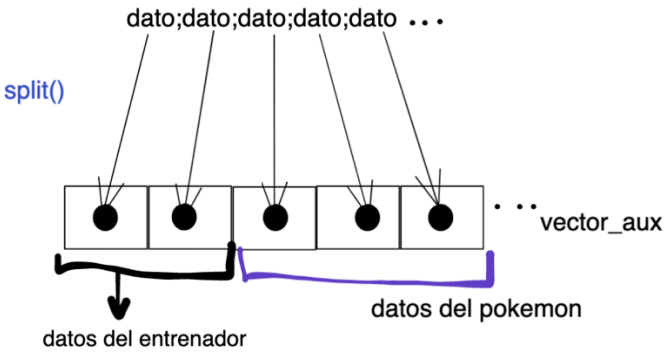
   Como trabajamos con memoria dinámica en caso de error se debe liberar la memoria que designamos, por eso al finalizar la función, libero la memoria utilizada, al igual que si llega a ocurrir un error.

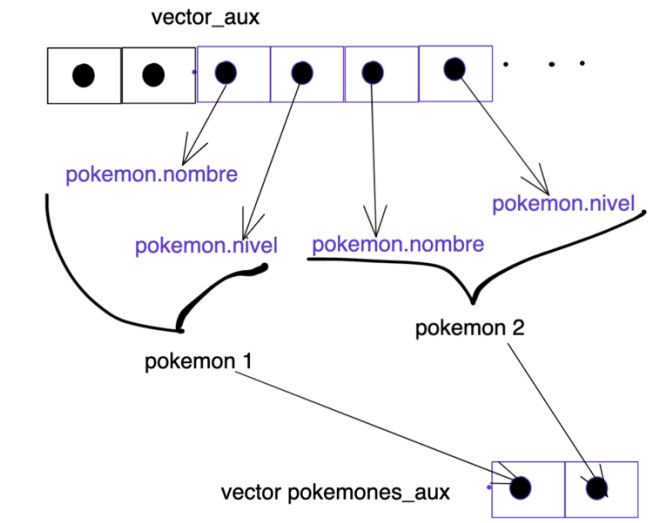
Para ubicar los datos obtenidos dentro del hospital en cada pokemon correspondiente, decidí crear una función que calcule el largo del vector en el que “copie” el contenido del archivo, como los primeros dos datos (elementos 0 y 1 del vector) representaban el nombre del entrenador y el id del entrenador, que no son necesarios en este caso, liberé la memoria que le había asignado antes.

Para el resto de los datos, como operamos por línea y cada posición par del vector incluía el nombre y cada posición impar del vector contenía el nivel, decidí recorrer el vector inicializando el iterador en 2, e ir asignando cada nombre y nivel a medida que son leídos a un pokemon.

Luego ese pokemon lo asigno a otro vector auxiliar del tipo pokemon_t en el cual voy a ir almacenando cada pokemon con su nombre y nivel que obtengo dentro de esa estructura iterativa, esto se repite mientras la línea sea valida.

Una vez que termino de trabajar con los vectores, debo liberar la memoria, para eso arme una función “liberar vectores” que libera las dos primeras posiciones del vector pasado por parámetro, el resto del vector se libera en la función que destruye el hospital.





Interpretación grafica de la función.

Antes de retornar true, indicador de que finalizó el proceso de lectura de un archivo, al vector de pokemones le aplico la función de ordenar_pokemones que ordena alfabéticamente cada pokemon, utilizando del método de ordenamiento de selección, que básicamente busca el elemento de menor valor (alfabético) en el vector y lo intercambia por el primero, luego busca el segundo menor (el siguiente al que ya pusimos en el primer espacio), y lo ubica en el segundo lugar, así sucesivamente hasta que quede ordenado por completo, las comparaciones las realiza con strcmp, debido a que comparo los nombre de cada pokemon.

3. Detalle de la función hospital_a_cada_pokemon

   En esta función verifico que ninguno de los dato pasados por parámetro con los que voy a trabajar sean NULL, o en el cado de la cantidad 0.

   Luego inicializo la variable contador_pokemones_funcion en la que voy a ir sumando la cantidad de pokemones a los que se les aplica la función, luego aplique un for para que itere mientras se cumplan con las condiciones, y así ir sumando la cantidad de veces que se aplica la función en la variable del contador.

   Para lograr que recorran alfabéticamente, decidí crear una función aparte para que el vector de pokemones se encuentre ordenado. (aplicada una vez que se termina de leer el archivo).

4. Detalle de la función hospital_destruir

   En esta función se destruye el hospital libreando la memoria que se le fue asignando mientras trabajábamos dentro de el. Esto incluye todos los elementos dentro de el (el pokemon, su nombre y su nivel, además del vector de pokemones).

5. Detalles sobre las funciones hospital_cantidad_pokemon, hospital_cantidad_entrenadores, pokemon_nivel y pokemon_nombre

   La implementación de estas funciones es muy similar para todas, chequeo que el dato que recibo por parámetro no sea null y retorno un puntero a lo que la consigna pide que debo retornar.