FACULTAD DE INGENIERIA
Universidad de Buenos Aires

# Trabajo Práctico N°1 – Hospital Pokemon

[75.41/95.15] Algoritmos y Programación II

Segundo cuatrimestre 2021


| | |
|---|---|
| Alumno: | DE NOBILI, Lautaro |
| Padrón: | 107394 |
| Email: | ldenobili@fi.uba.ar |

# Índice

1) **Introducción.**

El trabajo consiste en implementar las funciones de una librería que labura hospitales de pokemones.

Mi aproximación al TP fue hacer las funciones cortas y las largas y complicadas las fui haciendo al largo de la duración del periodo de entrega con mapas/diagramas mentales y escritos de memoria.

2) **Teoría.**

Cada espejo que comprás, ya viene usado.

3) **Implementación.**

I. Modificaciones al .c.
   i. Agregué size_t cantidad_entrenadores para facilitar el hacer la función hospital_cantidad_entrenadores. Esta variable aumentaría cada vez que se lee una línea del archivo (equivalente cada línea a un entrenador).

II. Funciones cortitas.
   i. Son aquellas que no entran explícitamente en las categorías de funciones largas o funciones agregadas. Son cortitas por su naturaleza de ser pocas líneas y sencillas de entender. Cada una de estas se basan simplemente en acceder a las estructuras hospital o pokemon y devolver los valores pedidos, los cuales ya existen dentro de esas estructuras. La única función que cabe destacar es la de destruir_hospital la cual ha de ser más larga que el resto, al tener que liberar memoria de cada variable de las estructuras.

III. Funciones agregadas.

i. obtener_siguiente_linea: lee del stream, ya abierto, pasado por parámetro una línea y la asigna al string que devuelve la función. El string que lee de la línea es una cantidad de bytes elegida arbitrariamente, pero en caso de no haber leído la línea completamente, se le asigna más memoria al string. Altamente basado en la implementación de Lucas en su video de cortitos.

ii. cantidad_pokemones: devuelve la cantidad de pokemones que hay en el split pasado por parámetro. Esta cantidad se obtiene recorriendo el split con un contador, que aumenta por cada substring. Esta cantidad total se le restan 2 del ID el entrenador y el nombre, y se lo divide por 2 para no contar el nivel de los pokemones, devolviendo así la cantidad de pokemones.

iii. agregar_pokemon_ordenadamente: utiliza una inserción ordenada para que los punteros a pokemon apunten en orden alfabético a cada pokemon. La inserción ordena consiste en recorrer el vector que contendrá al pokemon, en la recorrida si se encuentra un valor mayor en el vector al pokemon que queremos agregar (en este caso en base al nombre de ambos pokemones), usamos un auxiliar para almacenar la dirección de este pokemon mayor en el vector, luego en la posición de ese mayor ponemos a nuestro pokemon a agregar y ahora nuestro pokemon a agregar va a ser el que previamente fue mayor. Así hasta llegar al tope de pokemones. Finalizado el recorrido, asigno el pokemon que me quedo auxiliar, que sería el más “grande” en la posición del tope y agrando el tope. Algo a destacar, al final de la función se contempla la posibilidad de que el pokemon a agregar sea el primero. Otra cosa para destacar, como precondición, ya tiene que haber memoria reservada para este pokemon.

iv. splitted_a_hospital: pasa los substrings a sus respectivos lugares en el hospital. Recorro el split tantas n veces como la función cantidad_pokemones me diga. Tengo memoria reservada para un pokemon a agregar actualmente, el cual le asigno en la iteración el nombre según 2+2*n y el nivel según 3+2*n, por ser estas las posiciones de esos valores en el split. Paso el hospital y ese pokemon actual a agregar alfabéticamente. Finalizado eso, aumento en uno la cantidad de entrenadores.

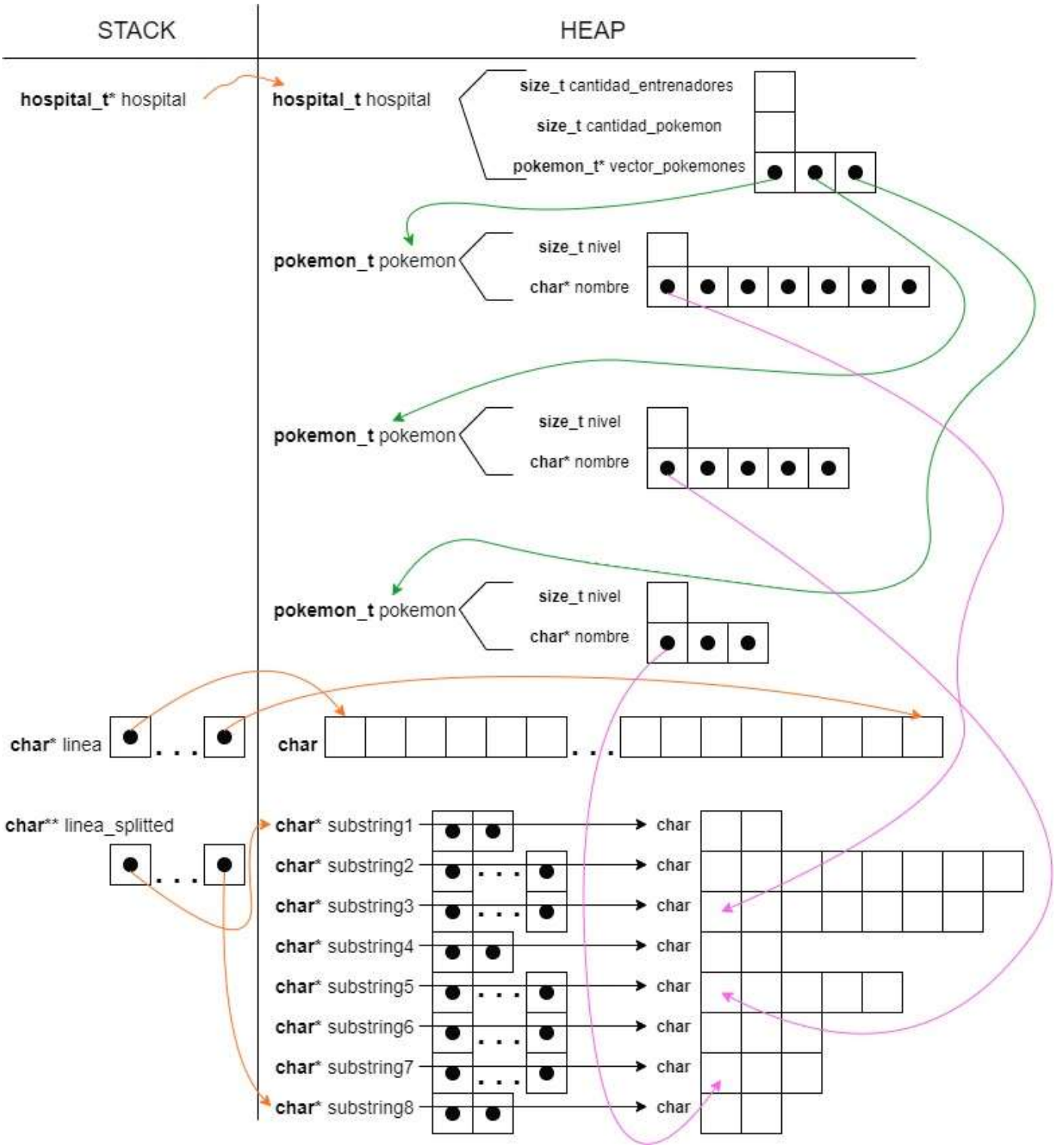
IV. Funciones largas.

i. hospital_leer_archivo: abro el archivo (y compruebo que se haya abierto bien), y obtengo la primera línea del archivo con la función obtener_siguiente_linea. Itero con un while mientras el obtener línea no me devuelva nulo, lo cual es sinónimo de que se llego al EOF. Dentro del while hago un split a la línea, reservo memoria para poder agrandar la cantidad de pokemones y asigno los contenidos del split al hospital. El vector de pokemones se deja ordenado alfabéticamente.

ii. hospital_a_cada_pokemon: itero con un while mientras no me haya pasado/llegado de/a la cantidad de pokemones en el hospital Y no haya obtenido un false en el puntero a función. Dentro del while aumento un contador de la cantidad de veces que apliqué la función a cada pokemon y aplico la función, asignando su valor de retorno a la condición de seguimiento para el while. Devuelvo el contador. Cabe destacar que antes ordeno mis punteros para que apunten alfabéticamente.

## 4) **Diagramas**

### I. Memoria durante lectura del archivo



## II. Memoria al final de la lectura del archivo