FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Trabajo Práctico 1 — Hospital Pokemon

[7541/9515] Algoritmos y Programación II
Segundo cuatrimestre de 2021

Alumno:	PELLEGRINO, Dafne
Número de padrón:	104368
Email:	dpellegrino@fi.uba.ar

Índice

1. Introducción

El/los archivo/s a leer contiene/n una o mas líneas. Cada línea contiene diferentes campos separados por punto y coma. El primer elemento de la línea es el ID del entrenador, luego le sigue su nombre, y a continuación le siguen los pokemon y nivel de cada uno de los entrenadores. A continuación se muestra un ejemplo de un archivo con 3 entrenadores:

ID_ENTRENADOR1;NOMBRE_ENTRENADOR1; POKEMON1;NIVEL; POKEMON2;NIVEL; POKEMON3;NIVEL

ID_ENTRENADOR2;NOMBRE_ENTRENADOR2; POKEMON1;NIVEL; POKEMON2;NIVEL

ID_ENTRENADOR3;NOMBRE_ENTRENADOR3; POKEMON1;NIVEL; POKEMON2;NIVEL; POKEMON3;NIVEL; POKEMON4;NIVEL

En esta oportunidad nos vamos a asegurar de que los archivos vengan sin errores. Pueden venir archivos vacíos, pero si el archivo no está vacío, por lo menos va a tener una línea con un entrenador y por lo menos un pokemon. Nunca va a haber entrenador sin pokemon o pokemon sin la información del entrenador.

Para comenzar planteo como va a estar estructurado el hospital y el pokemon y qué datos necesito que tenga, basado en las funciones que necesito implementar.

Necesito que en el hospital esté el dato de la cantidad de pokemones que se atienden y la cantidad de entrenadores que hacen que sus pokemones se atiendan ahí. También necesito un listado de pokemones con el dato de su nombre y nivel.

2. Detalles de implementación

Para compilar el programa se puede utilizar el atajo de makefile "make pruebasz luego para ejecutar el programa corriendolo en valgrind para chequeo de memoria puede sencillamente ejecutar el comando definido en el makefile con "make valgrind".

Para la implementación defino un tipo de dato con una estructura para el hospital y otro para el pokemon. El hospital almacena dos size_t con la cantidad de pokemones y de entrenadores y un arreglo dinámico de tipo pokemon_t (ver diagrama 1). Cada pokemon almacena un size_t con su nivel y un arreglo dinamico de chars con su nombre (ver diagrama 2). Referencia a diagramas de estos tipos de datos y su estructura en memoria en la sección "Diagramas".

2.1. Detalle hospital_crear

Esta funcion reserva memoria para un hospital (tipo de dato hospital_t) y lo inicializa con 0 cantidad de entrenadores y pokemones. También reserva memoria para un pokemon_t* asignándola al vector de pokemones. En cada caso de reserva de memoria chequea si falla y en caso de ser necesario libera alguna memoria anteriormente reservada.

La función no recibe nada y retorna el hospital creado.

2.2. Detalle hospital_leer_archivo, leer_línea y agregar_linea_hospital

La función hospital_leer_archivo recibe un hospital y el nombre de archivo a leer. En caso de que alguno de los dos sea NULL o que el archivo no se pudo abrir, se retorna la información de que no se pudo leer el archivo.

Para la implementación utilicé un bucle que se repite mientras el valor leer_archivo sea verdadero. Ese valor cambia a falso en el caso de que una línea no se pueda leer mediante la función auxiliar de leer_línea o que la función split para separar una línea en distintos string falle. La funcion leer_linea funciona recibiendo el archivo abierto y mientras va aumentando el tamaño de la memoria reservada para un string auxiliar con un incremento de 80 lee el archivo mediante fgets. Si fgets falla la función retorna NULL y si llega al final de la linea con la presencia de "retorna la línea leída.

Luego si dichas funciones se ejecutan sin fallas se llama a la función agregar_linea_hospital, que agrega la línea leída al hospital reservando la memoria necesaria para que los pokemones quepan

y así también sus nombres. Dicha función auxiliar recibe el hospital para poder modificarlo y un doble puntero con los pokemones a agregar al hospital y sus niveles. Para poder guardar el dato de nivel o nombre de pokemon donde corresponde utilizo el dato de que en la posición 0 se encuentra el número de entrenador, en la posición 1 el nombre de dicho entrenador y, a partir de la posición indexada como 2, las posiciones pares contienen nombres de pokemones y las posiciones impares niveles. En cada caso se agrega el dato a la posición correspondiente en el vector de los pokemones.

Siguiendo la función de hospital_leer_archivo cuenta la cantidad de líneas leídas mediante la cantidad de bucles y llamados a la función que lee líneas y se lo suma a la cantidad de entrenadores teniendo en cuenta que cada línea tiene un entrenador seguido de todos los datos de sus pokemones.

Para finalizar cierra el archivo y retorna éxito si todo funcionó correctamente.

2.3. Detalle de hospital_cantidad_pokemon, hospital_cantidad_entrenadores, pokemon_nivel y pokemon_nombre

Las funciones hospital_cantidad_pokemon y hospital_cantidad_entrenadores reciben el hospital y retornan el valor de la cantidad de entrenadores o de pokemones si el hospital no es NULL.

Las funciones pokemon_nivel y pokemon_nombre reciben un pokemon y retornan el nivel o el nombre del pokemon respectivamente si el pokemon no es NULL.

2.4. Detalle de hospital_a_cada_pokemon, ordenar_pokemon_alfabético y swap

La función swap es auxiliar a ordenar_pokemon_alfabético y recibe dos punteros a pokemones e intercambia sus posiciones en memoria utilizando un puntero a pokemon auxiliar y la función memcpy.

La función ordenar_pokemon_alfabético ordena el vector de pokemones recibido mediante bubblesort de la a a la z comparando los nombres de cada pokemon con el posterior. Mientras la variable ordenado sea falsa sigue comparando e intercambiando en el caso necesario.

La función hospital_a_cada_pokemon recibe un hospital y una función a aplicar a cada pokemon del hospital. En caso de que los datos recibidos sean NULL la función retorna 0, sino retorna la cantidad de pokemones recorridos.

Previamente a aplicar la función recibida es preciso ordenar el pokemon alfabéticamente, por lo que se llama a la función descripta anteriormente de orden de pokemones alfabéticamente. Luego se procede a aplicar la función recibida mientras el contador sea menor a la cantidad total de pokemones y mientras la funcion devuelva verdadero. En caso de que la función se aplique y retorne falso se devuelve el contador como cantidad de pokmones.

2.5. Detalle de hospital_destruir

La función de destrucción recibe un hospital y libera la memoria reservada para los nombres de los pokemones, así como para el vector de pokemones en sí y por ultimo el hospital.

3. Diagramas

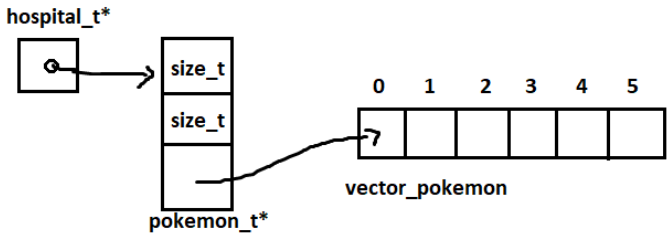

Figura 1: Diagrama 1: Estructura hospital

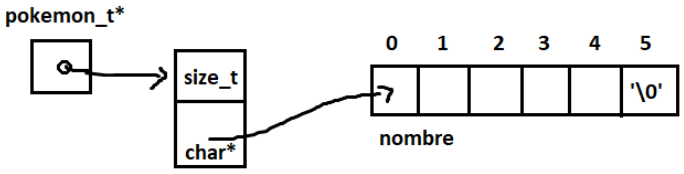

Figura 2: Diagrama 2: Estructura pokemon