FACULTAD DE INGENIERIA

Universidad de Buenos Aires

Trabajo Práctico N1 — Hospital Pokemon

[7541/9515] Algoritmos y Programación II

Segundo cuatrimestre de 2021

Alumno:	Vallejos, Matias
Número de padrón:	107924
Email:	matiavallejo@gmail.com
Email Institucional:	mvallejos@fi.uba.ar

1. Introducción

La idea de este TP es plantear una solucion simple para lograr listado y busqueda de pokemones dentro de un struct el cual representa un hospital, se planteo utilizando memoria dinamica y un vector dinamico en donde se alojarian los pokemones con sus respectivos nombres y niveles se utilizaron los siguientes archivos:

- **hospital.h** esta es la biblioteca principal del archivo en ella estan casi todas las funciones utilizadas y sus explicaciones de como utilizarlas
- **split.h** esta biblioteca es utilizada para separar un string dado un delimitador
- **split.c** en este archivo esta la implementacion de las funciones del split.h
- **hospital.c** este es el archivo principal, es aqui donde se usan todas las bibliotecas y se realiza el trabajo final del TP
- **pa2mm.h** este es el framework de pruebas utilizado para probar el correcto funcionamiento del TP
- **pruebas.c** este es el archivo principal de pruebas del TP
- **makefile** este es un archivo ejecutable que nos indicara si nuestro tp es apto para lograr el objetivo mediante la ejecuccion de pruebas

2. Detalles de implementación

1. **bubble_sort**

   Recibe un hospital el cual ya debe estar creado,con un tope inicializado y ordena el vector de pokemones por su nombre de manera alfebetica con el metodo de ordenamiento bubble sort,devuelve un puntero del vector ordenado.

2. **strdup**

   Recibe un puntero a string que no es null, la funcion devuelve un puntero a un nuevo string que es un duplicado del string recibido por parametro, la memoria del nuevo string es obtenida con malloc y debe ser liberada por el usuario. En caso de error devuelve NULL.

3. **leer_linea**

   Recibe un archivo el cual debe ser valido, esta funcion lee la primer linea de un archivo y devuelve un puntero hacia el string de esa linea, para realizar esto lo que hace es utilizar la funcion fgets() la cual lee los caracteres de una linea y los almacena en un buffer hasta llegar al final de la linea osea donde se encuentra el '\0', esta funcion implementa memoria dinamica para poder leer correctamente toda la linea, mediante el uso de esta va aumentando el buffer hasta que realmente llegue al final de la linea. Finalmente devuelve un puntero a ese buffer. El usuario debe encargarse de liberar la memoria. En caso de error devuelve NULL.

4. **split**

   Recibe un string el cual debe ser obtenido mediante la utilizacion de la funcion leer_linea y un separador y devuelve un vector dinamico que contiene a los substrings delimitados por el separador. Al final del vector debe haber un string NULL o '\0' para indicar que es el final del mismo. En caso de falla devuelve NULL. El usuario debe encargarse de liberar la memoria.

5. **split_len**

   Recibe un puntero doble de caracteres que va a ser un string, dicho string debe ser obtenido mediante la utilizacion de la funcion split y devuelve la cantidad de elementos que contiene el split esto lo logra recorriendo el string y buscando donde es NULL ya que la funcion split en el final del vector devuelve null.

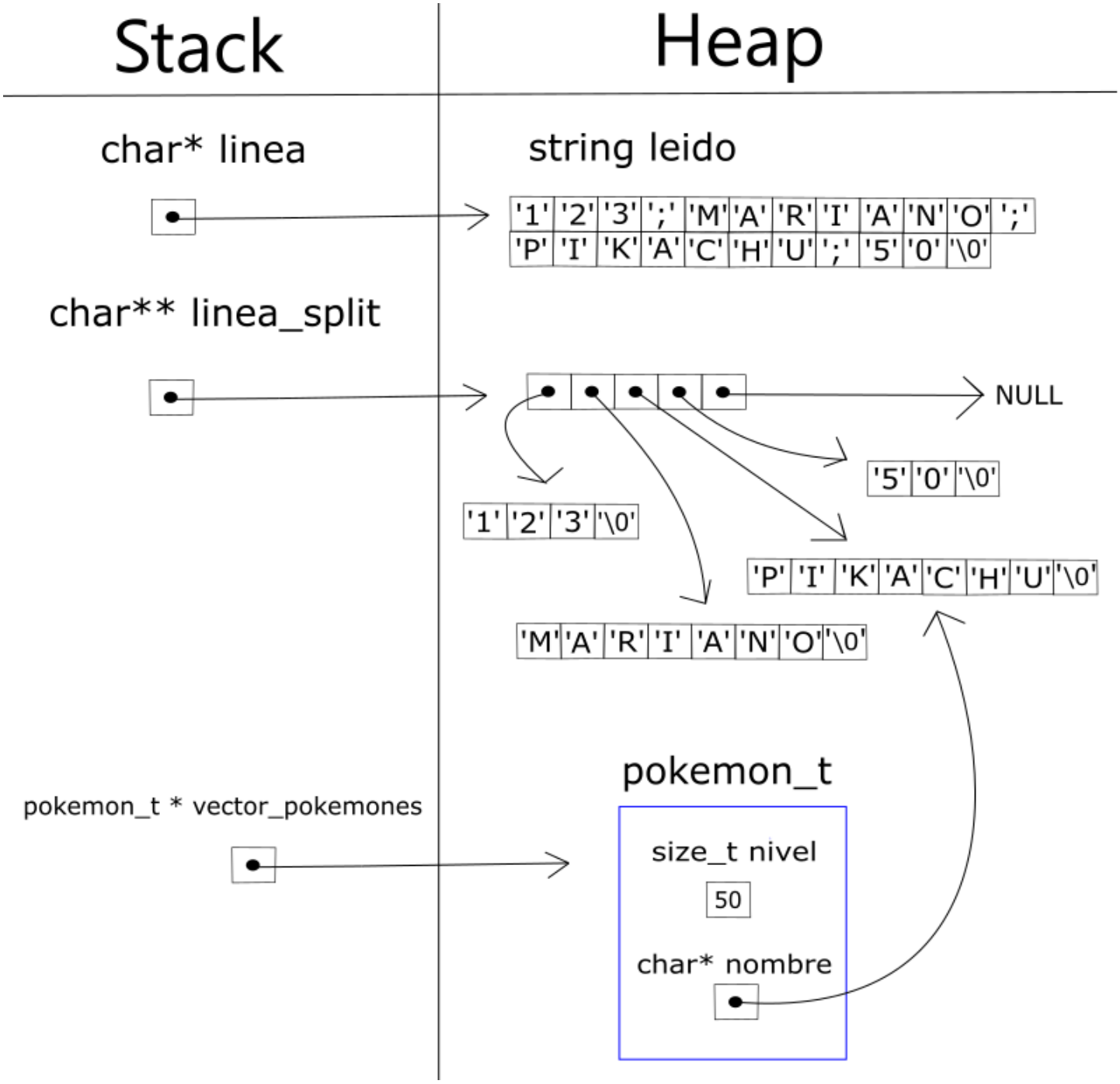
6. **agregar_pokemon**

   Recibe un hospital creado, un char** del vector linea que se consigue con la funcion split, un puntero de tipo pokemon_t el cual se utilizara como vector y una posicion que debe estar inicializada. Agrega el nombre y nivel del pokemon al vector utilizando la funcion strdup y aumenta la cantidad de pokemones. Finalmente agrega este vector al vector_pokemones dentro del hospital. El usuario es el encargado de liberar el nombre del pokemon.

7. **buscar_y_agregar_pokemon**

   Recibe un hospital ya creado, un puntero hacia una linea obtenida con leer_linea, un punto de pokemon_t que es un vector ya creado y un tamaño que debe estar incializado. Esta funcion splitea la linea recibida mediante la funcion split y va agregando los pokemones y sus niveles leidos en esa linea split a un vector utilizando memoria dinamica y el tamaño recibido por parametro, si el tamaño del vector no es suficiente lo duplica, agrega los pokemones utilizando la funcion agregar_pokemon. Finalmente aumenta la cantidad de entrenadores por cada linea leida y devuelve true si todo salio bien o false si hubo algun fallo. (Ver diagrama)

3. Diagrama



En este diagrama podemos observar como se crea el char* linea el cual va a apuntar al string leido con la funcion fgets, luego se crea el char** linea_split el cual va a ser un vector dinamico que va a separar el string de la linea por su separador ';', notese que al final de cada elemento del vector dinamico hay un \0 que indica el final de ese substring. Finalmente se crea el pokemon_t * vector pokemones el cual va a tener un puntero hacia una copia del nombre que esta en el split y el valor del nivel el cual va a ser un size_t y no un puntero.