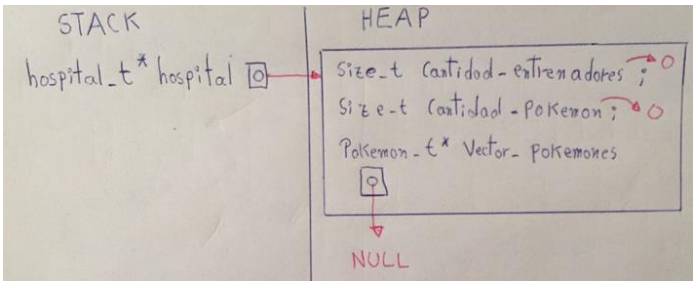
INFORME DEL TRABAJO PRACTICO #1 DE ALGORITMOS Y PROGRAMACIÓN II CURSO 04-MENDEZ

Alumno: Víctor Zacarías, Padrón: 107080, Mail: vzacarias@fi.ubar.ar

-Descripción: En este informe se describen los algoritmos implementados en el desarrollo del trabajo práctico, también se incluyen algunos esquemas que facilitan el entendimiento de las implementaciones de los algoritmos.

1-Función hospital_crear: Inicializa un puntero de tipo hospital_t* con la dirección de memoria de un bloque de tamaño hospital_t, reservando la memoria necesaria para poder crear dicho registro en el heap, en donde se encuentra la información del hospital (vale recordar que esta se encuentra inicializada después de usar la función, es decir con ceros y NULLs donde corresponda).

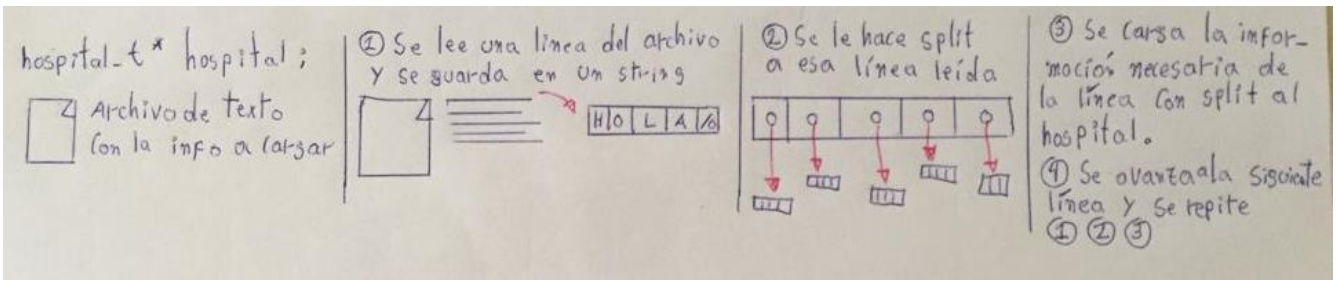


Hace también los chequeos necesarios durante la reserva de memoria, que, en caso de haber un error, aborta la ejecución de la función devolviendo NULL.

2-Función hospital_leer_archivo: Dado un archivo de texto con un formato determinado, y un puntero a un hospital (ya inicializado), la función se encarga de procesar el archivo de texto línea por línea, y cargar esa información al hospital.

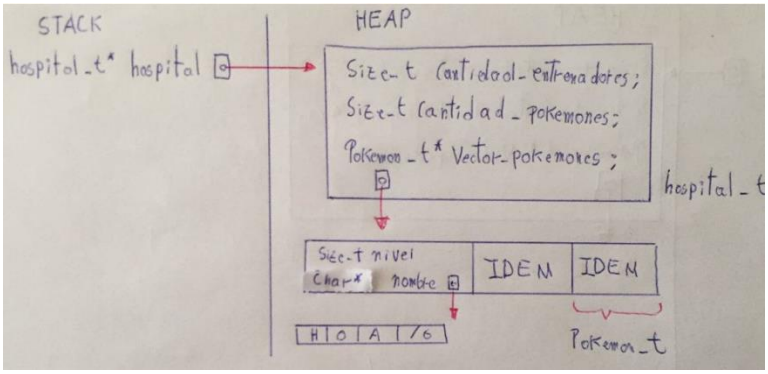
Después de hacer los chequeos sobre los parámetros de la función, verificando que todo esté en orden, se crean dos variables: linea_de_archivo y split_linea_de_archivo.

El algoritmo implementado se basa en leer una línea del archivo y guardarla en linea_de_archivo, hacerle split a esa línea y guardarla en split_linea_de_archivo, y luego haciendo uso de una función auxiliar, se carga la información importante del split al hospital. Este algoritmo se repite hasta leer



Bucle de lectura de archivo y carga de información al hospital, el bucle termina al llegar el fin de archivo o si ocurre un problema

todas las líneas del archivo y cargar su información al hospital. La memoria necesaria para linea_de_archivo, split_linea_de_archivo, y para completar el hospital, es reservada en las funciones auxiliares utilizadas por esta función. Si hay algún error durante la ejecución, se libera la memoria que no se va a seguir usando (linea_de_archivo y split_linea_de_archivo) y se aborta la ejecución de la función, dejando al hospital actualizado con la información cargada antes del error. Si todo sale bien, el hospital queda cargado de la siguiente forma:



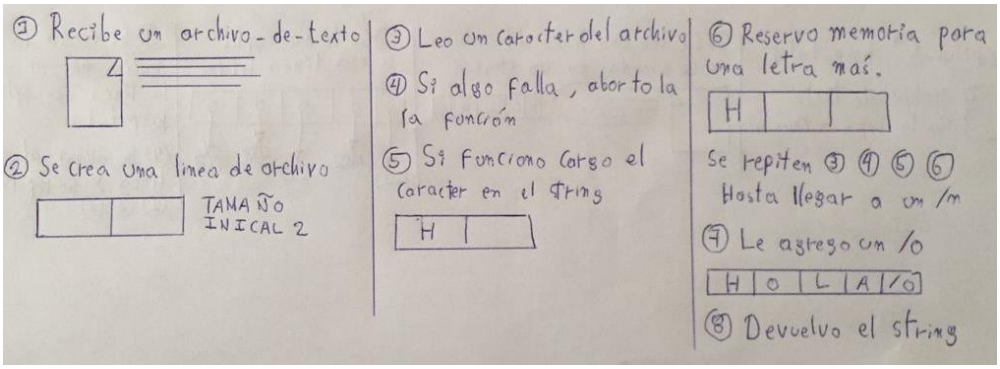
A continuación, se detallan las funciones auxiliares usadas en hospital_leer_archivo:

2.1-Función proxima_linea_de_archivo: La función recibe el archivo a leer, y se encarga de leer el archivo hasta encontrar un '\n', cargando la información leída en un string dinámico (linea_de_archivo), dicho string es posteriormente retornado por la función.

La función hace uso de una estructura iterativa para leer carácter a carácter una línea de un archivo (hasta llegar a un '\n') e ir cargando esos caracteres en el string dinámico, la idea es que al llegar al '\n', se tiene cargado en el string dinámico la línea actual del archivo.

Para esto último, se leen los caracteres del archivo con la función fgetc, y se va aumentando el tamaño del string dinámico a medida que se lean nuevas letras.

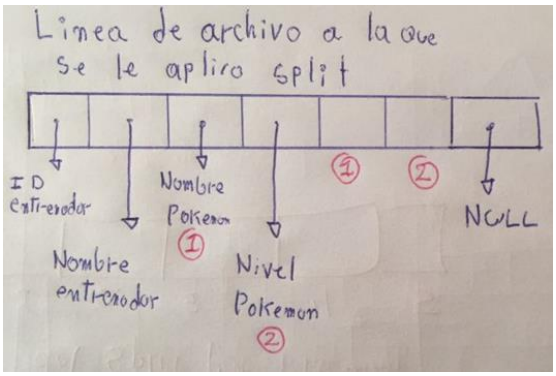
La función a su vez hace todos los chequeos necesarios cada vez que se lee una letra o se agranda el tamaño del string dinámico.




```
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
```

Estructura del archivo leído.

2.2-Función split: Dado un string y un separador, la función separa al string de los separadores en un vector de strings.



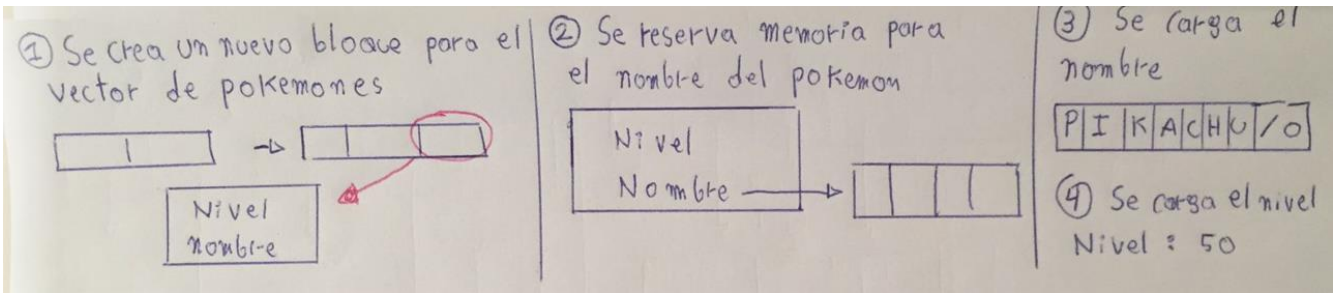
Línea de archivo a la que se le aplico split, el separador en este caso es ';'

2.3-Función actualizacion_de_hospital: La función se encarga de actualizar el hospital con la información de los pacientes (línea del archivo a la que se le hizo split). Para esto, la función después de verificar los parámetros mandados y de crear variables auxiliares, se vale de una estructura iterativa que permite cargar la información de los pacientes en el vector de pokemones del hospital.

La información de los pacientes se encuentra en un vector de strings, que se lee desde la tercera posición y a dos posiciones por iteración (esto último porque las dos primeras posiciones del string no contienen información que requiere el hospital, y a partir de ahí, la información de cada paciente se guarda en dos posiciones del string. *Mirar foto de función split*).

Por cada iteración se reserva un bloque de memoria adicional para el vector de pokemones, luego en el campo nombre (del vector de pokemones) se reserva la memoria necesaria para guardar el nombre del pokemon, una vez hecho esto se guarda el nombre del pokemon en el bloque creado. Finalmente, se guarda la información del nivel del pokemon en el campo nivel (del vector de pokemones). La información que se carga al hospital se lee del vector de strings (info pacientes) y cada vez que se agrega algo se avanza a la siguiente posición del mismo, una vez terminada una iteración se repite el proceso ya explicado (si se cumplen las condiciones para hacerlo).

La función hace todos los chequeos necesarios sobre parámetros y asignación de memoria, y en caso de haber un error en la ejecución de la función, la misma deja de ejecutarse, y el hospital queda actualizado con la info cargada antes del error.



Algoritmo de carga de información al hospital, el bucle termina hasta leer todo el split o al haber un error.

2.4-Función ordenar_pokemones_alfabeticamente: No es más que un algoritmo de burbujeo para ordenar el vector de pokemones (campo del registro hospital) alfabéticamente (por nombre de pokemon).

NOTA: El algoritmo de la función hospital_leer_archivo se basa en usar una estructura iterativa con las funciones 2.1, 2.2, 2.3 hasta cargar toda la información del archivo al hospital. Esto es posible ya que en 2.1, cada vez que se lee una línea, el archivo queda "parado" en la siguiente línea, lo que permite leer el archivo hasta al final y procesar la información del mismo.

Leer->Hacer split->Cargar información al hospital (bucle).

3-Función hospital_a_cada_pokemon: A partir de una estructura iterativa se le aplica una función a todos los pokemones del vector de pokemones (hasta que no queden más pokemones) o hasta que la función devuelva false. La función devuelve a cuantos pokemones se le aplico la función "funcion".

4-Función hospital_destruir: Libera la memoria ocupada por hospital y por todas sus estructuras internas: vector de pokemones y nombre de cada pokemon.

NOTA: No se agregaron al informe la descripción de funciones y algoritmos cuyo funcionamiento consta de pocas acciones o de una sola acción.