Trabajo Práctico 1 — Hospital Pokemon

[7541/9515] Algoritmos y Programación II

Segundo cuatrimestre de 2021

| Alumno: | Hurtado, Daniel | | ---------- | ---------- | | Número de padrón: | 107404 | | Email: | dhurtado@fi.uba.ar |

1. Introducción

Este TP1 tiene como objetivo la implementación y puesta en práctica los conocimientos adquiridos en el transcurso del curso en torno al uso de memoria dinámica (heap) con C (creación, asignación, lectura y destrucción), en conjunto al uso de punteros (aritmética de punteros, punteros a función). Así como también el uso en conjunto con otros conocimientos previos como tipos de datos estructurados y manejo de archivos, aplicándose a este enfoque más avanzado.

En cuanto a la temática del TP1, gira en torno a los hospitales pokemon, para los cuales se solicita la creación de diversas funcionalidades relacionadas al guardado y administración de la información de sus pacientes (pokemones) y sus respectivos entrenadores.

Debido al objetivo del mismo, se solicita realizar una implementación libre de limitaciones en cuanto a la información, por lo que presuponer o establecer valores límites para el guardado de información, no es una opción. Por ende toda necesidad de manejo de información desconocida, se llevará a cabo mediante memoria dinámica.

2. Detalles de implementación

Detalles especificos de la implementacion, como compilar, como ejecutar

1. Estrategia: Archivos / Datos

   La estructura de los archivos y datos contenidos en ellos, definida en el enunciado del TP1 consiste en archivos de texto, de acceso secuencial y con la información contenida en ella separada por ';' (CSV). Se trabajará con único tipo de datos a leer que contará con la siguiente estructura definida por id y nombre del entrenador, y posteriormente y de manera ilimitada, el listado y detalle de sus pokemones, de la siguiente manera:

   - 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

     Debido la cantidad de contenido no específico, la lectura de los mismos se realiza de manera dinámica y no específica (como un fscanf), obteniendo cada línea del archivo como string que posteriormente procesaremos para obtener y guardar la información correspondiente. Recalcando la necesidad de poder leer sin establecer límites, se hace uso de memoria dinámica para el guardado del línea leída sin establecer un límite o

buffer, para ello se optó por hacer uso de fgetc, iterando carácter a carácter (e ir guardando en memoria dinámica) hasta llegar al final de la línea (\n), procesar la información, y repetir el proceso para siguientes líneas.

2. Librería: Split Se cuenta con una librería reutilizable cuyo contrato se encuentra en split.h, con una función llamada split, cuya firma es: "char* *split(const char* string, char separador);", que permite separar un string recibido en base al separador en múltiples substrings (con memoria dinámica) y obtenerlos de regreso en forma de array de strings.

   Dicha función es utilizada en el procesamiento de la información obtenida en la lectura de cada línea del archivo para poder obtener individualmente cada uno de los datos contenidos en ella.

   Ver Diagrama de Split.

3. Requerimiento: Hospital El contrato del TP, nos solicita la creación de una serie de funciones con el objetivo de trabajar con hospitales pokemon, permitiendo crear un flujo de creación, actualización/escritura y lectura de su contenido. Principalmente se trabaja con las siguientes funciones:

   Creación: Se encarga de crear el tp, enfocada en memoria dinámica, por lo que toda la información se guardará en el heap, devolviendo el puntero a dicha ubicación de memoria.

   Actualización (Lectura de archivo): Siguiendo la estrategia de lectura de archivos antes mencionada en los puntos anteriores, se actualiza el hospital con los datos recibidos, los cuales también son manejados mediante vectores de memoria dinámica, por lo que se realiza asignación de la memoria requerida por cada dato recibido, permitiendo recibir datos sin límites establecidos (obviando límites externos de memoria, almacenamiento, etc).

   Destrucción: Ya que todos los datos se manejan en memoria dinámica, previo a terminar de utilizar el programa, se utiliza esta función para la destrucción del contenido dentro de los vectores dinámicos y hospital en sí.

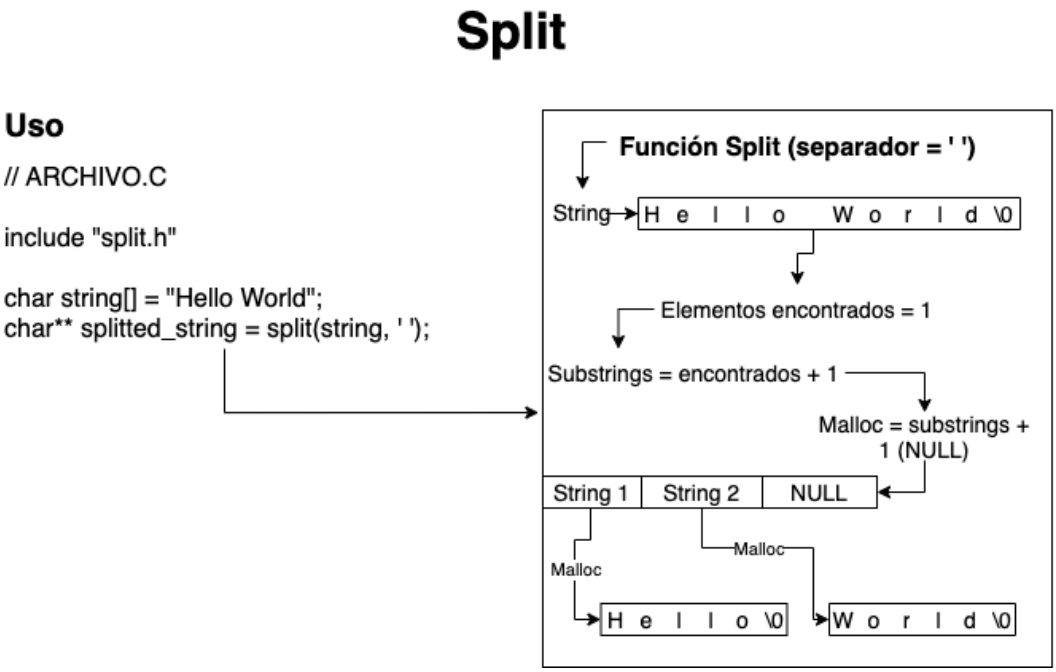
   Ver diagrama de datos y funciones principales.

4. Utilización: Compilación y Pruebas

   El desarrollo completo se pone a prueba mediante el framework de testing y pruebas provistas por la cátedra en el contenido de este TP1. Tienen como objetivo comprobar el correcto funcionamiento de cada una de las funciones solicitadas, así como también detectar errores y problemas en cuanto al uso de memoria dinámica, tales como lecturas o asignaciones inválidas y memoria perdida por no liberarse, haciendo uso de Valgrind.

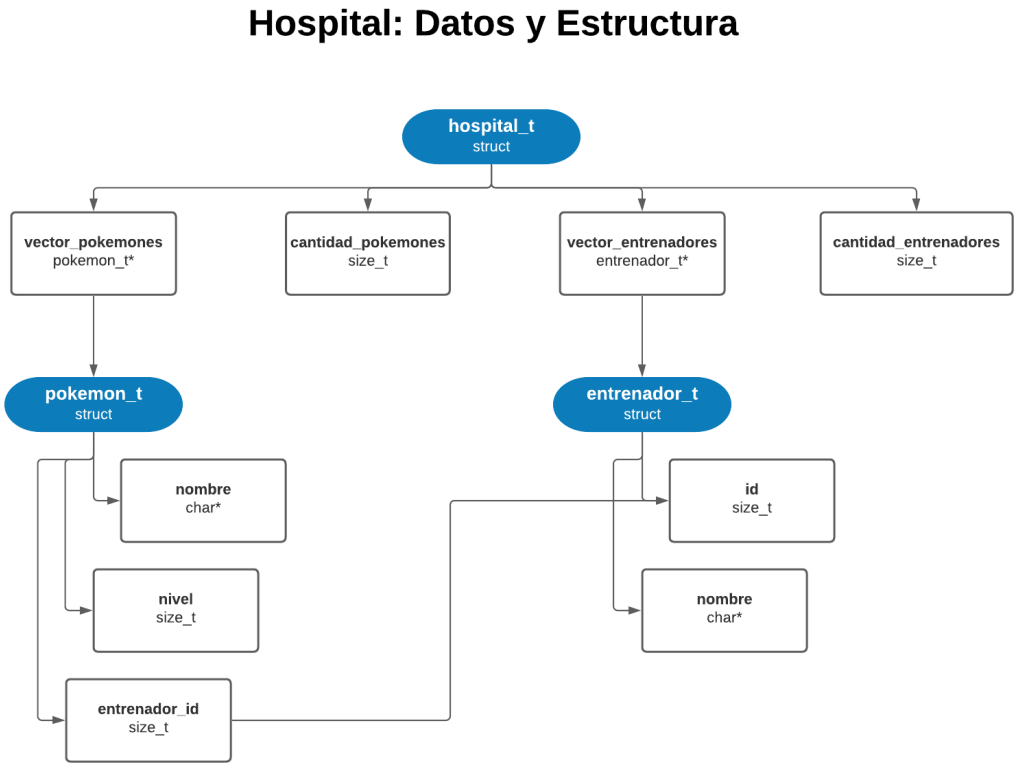
3. Diagramas

1. Función: Split



La implementación de split consiste en obtener un string y un separador, recorrer el string recibido con el objetivo de contar la cantidad de separadores en él, para conocer la cantidad de substrings dentro de él y crear con ellos un array de punteros a char, en donde en cada uno de ellos asignaremos con memoria dinámica el substring correspondiente según el contenido entre separadores de la posición buscada.

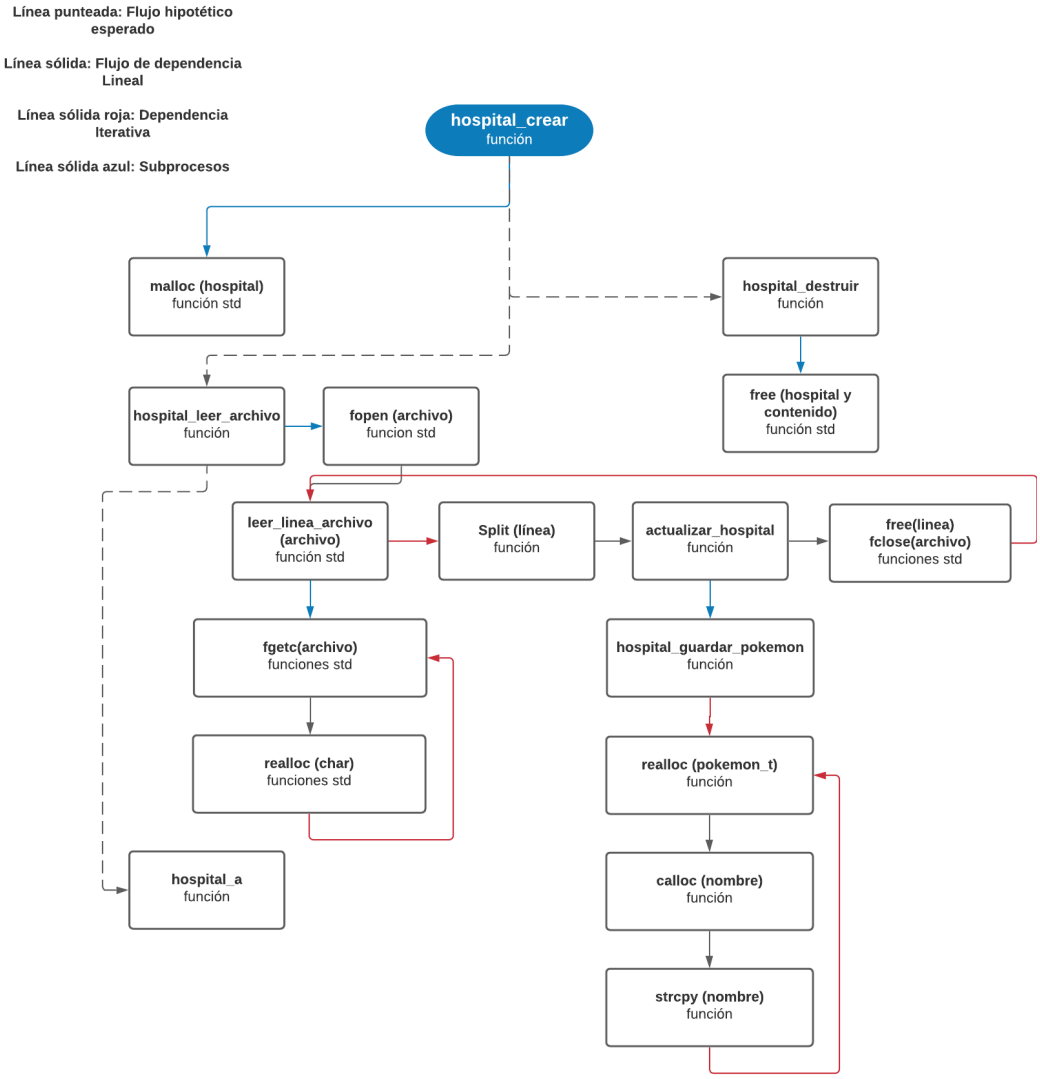
2. Datos y Estructura: Hospital



En el diagrama se puede ver la estructura de datos que conforman Hospital, los cuales son utilizados a lo largo del TP y sus funcionalidades.

3. Funciones Principales: Hospital

Hospital: Funciones Principales



En el diagrama se puede observar el flujo y/o procesamiento de datos de las principales funciones del TP, entendiendo en que etapas de ellas se hace uso de memoria dinámica y lectura de archivos, indicando también las situaciones de iteración.