Trabajo Práctico 2 — Simulador

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

Alumno:	Castillo, Carlos E.
Número de padrón:	108535
Email:	ccastillo@fi.uba.ar

Índice

1. Introducción

El trabajo practico presenta un hospital pokemon en el que los diferentes entrenadores pueden atender a sus pokemones. Inicialmente se provee la información de los entrenadores ingresantes y sus pokemones a traves de un archivo de texto. El objetivo es implementar la funcionalidad de atencion de los pokemones en el hospital a traves de un simulador que permita gestionar a los pacientes internados.

Dicho simulador toma control de la informacion del hospital y permite simular una serie de eventos a traves de diferentes comandos, que ayudan a las enfermeras del hospital a atender a cada pokemon adivinando su nivel.

Para la construcción del simulador del el hospital de pokemones se deben utilizar algunas de las diferentes estructuras de datos estudiadas durante el curso de la materia, identificando cual tda resulta mas conveniente para implementar cada funcionalidad del simulador.

2. Teoria

3. Detalles de implementación

La implementación de esta estructura de datos fue escrita en el lenguaje de programación C, siguiendo la especificación del lenguaje detallada en el estándar C99. Los archivos principales se encuentran dentro del directorio **src** ubicada en la raíz del repositorio.

Además se incluye un archivo **pruebas.c** en el que se encuentran diferentes pruebas automatizadas para detectar errores en la ejecución del programa o en la asignación, manejo y liberación de memoria dinámica. Para esto se utiliza **gcc** como compilador y **valgrind** como herramienta de análisis de memoria.

El trabajo practico se encuentra dividio en dos componentes principales: la interfaz y el simulador. La implementacion de la interfaz abarca los archivos **main.c** y **juego.c**. Por otra parte, la implementacion del simulador se divide en los archivos **simulador.c**, en donde se encuentra la funcionalidad base del simulador, y los archivos auxiliares **aux_simulador_atencion_pokemon.c** y **aux_simulador_dificultades.c**, que almacenan funcionalidad comun utilizada en bastantes partes del simulador.

3.1. Simulador

El simulador provee tres funciones principales: `simulador\_crear`, que se encarga de la creacion e inicializacion de un nuevo simulador con su respectivo hospital, `simulador\_simular\_evento`, que evalua los diferentes comandos enviados al simulador y ejecuta el evento correspondiente a dicho comando, y finalmente `simulador\_destruir` que se encarga de la destruccion del simulador y liberacion de la memoria ocupada por el mismo.

La estructura propuesta para el simulador es la siguiente:

```
struct simulador {
    hospital\_t* hospital;
    EstadisticasSimulacion estadisticas;

    heap\_t* recepcion;
    lista\_iterador\_t* sala\_espera\_entrenadores;
    lista\_iterador\_t* sala\_espera\_pokemones;
    PokemonEnRecepcion* pokemon\_en\_tratamiento;

    abb\_t* dificultades;
    DatosDificultadConId dificultad\_en\_uso;
    unsigned intentos\_actuales;

    bool en\_curso;
};
```

El primer campo `hospital` hace referencia al hospital que se utiliza con el simulador. Este ultimo obtiene la informacion de los registros de ingresos de entrenadores y pokemones guardados previamente en el hospital antes de que estos sean cargados por turnos a la simulacion. El campo `estadisticas` le permite al simulador llevar control sobre los el estado actual de la simulacion, almacenando datos como el numero total de entrenadores y pokemones, la cantidad de pokemones atendidos y en espera, entre otras estadisticas de la simulacion en curso.

Los campos `dificultad\_en\_uso`, `intentos\_actuales` y `en\_curso` permiten al simulador mantener una referencia a ciertos datos del estado actual de la simulacion que son de utilidad en algunos de los eventos registrados en el simulador.

3.2. Sala de Espera, Recepcion y Atención de Pokemones

Inicialmente los entrenadores esperan pacientemente junto con sus pokemones en la sala de espera del hospital. Para almacenar esta informacion en el hospital se utilizan tres estructuras de datos diferentes: dos listas enlazadas y un árbol binario de búsqueda. Las listas enlazadas son de utilidad para guardar un registro de los entrenadores que van ingresando al simulador en el orden de llegada de los mismos al hospital, que es el mismo orden en el que los entrenadores son cargados al simulador. Al utilizar una lista enlazada este orden se respeta, ya que los elementos se almacenan en nodos dispuestos uno tras otro de manera consecutiva, en el orden en el que van siendo insertados. Precisamente por esta razón, en este caso la operación de inserción en la lista enlazada solamente se hace al final de la lista para cada nuevo elemento.

3.3. Interfaz

La parte de la interfaz hace las veces de cliente para el simulador.

El campo `casillas` de cada hash es un vector de listas simplemente enlazadas y puramente recursiva, implementadas de la siguiente forma:

```
typedef struct casilla {
    char* clave;
    void* elemento;
    struct casilla* siguiente;
} casilla_t;
```