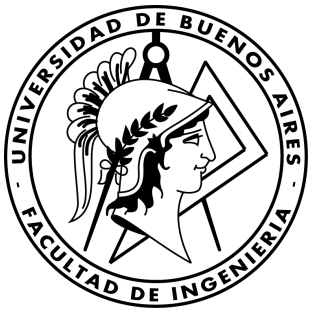

Trabajo Práctico n1 — Escape Pokemón

[7541/9515] Algoritmos y Programación II
Primer cuatrimestre de 2022

Alumno:	Vainstein Aranguren, Tomás
Número de padrón:	109043
Email:	tvainstein@fi.uba.ar

Índice

1. Introducción **2**

2. Teoría **2**

3. Detalles de implementación **2**
3.1. Detalles de 'objeto/interaccion crear desde string' 2
3.2. Detalle de 'agregar objeto/interaccion' 3

4. Diagramas **3**

1. Introducción

La idea general del TP es cargar la información de dos archivos utilizando vectores dinámicos para luego poder imprimir la información por pantalla y, por último, liberar la memoria utilizada.

2. Teoría

1. - Cómo funciona la lectura de archivo

   La lectura de archivos se hace asigandole lo devuelto por la función fopen a un puntero FILE con un nombre determinado, en donde se utiliza el path del archivo que se quiere abrir, y usando 'r' si se quiere leer y 'w' si se quiere escribir. En caso de que se falle la apertura de archivos, fopen devuelve NULL. También se suele indicar que hubo un fallo imprimiendo un mensaje de error en la terminal y/o devolviendo un valor como -1 o NULL.

2. - Cómo manejas la memoria dinámica

   Para asignar memoria dinámica se utilizan las funciones malloc en caso de que sea la primera vez que se asigna y realloc cuando se quiere modificar el tamaño de la memoria ya asignada. Para asignarle memoria a un tipo especifico de dato, hay que especificar la cantidad multiplicada por el tipo con el que se está trabajando. Luego de intentar asignar la memoria dinámica hay que verificar si el proceso fue exitoso o no, y al finalizar la utilización de la misma se tiene que liberar ordenadamente para no perder la dirección de memoria de cada tipo de dato con el que estamos trabajando.

3. Detalles de implementación

Lo primero que vi necesario hacer cuando empecé a realizar el trabajo práctico fue cargar la información de los archivos. Para lograr esto, primero le asigne los parámetros argv[1] y argv[2] al llamado de la función "sala crear desde archivos", lo que haría que se abran los dos archivos recibidos en los argumentos cuando se ejecute el programa, esperando que ambos sean los path a los archivos de objetos e interacciones respectivamente. Utilizando dos procedimientos (para el archivo de objetos y otro para el de interacciones) guardo la linea del archivo obtenida en un string con máximo de 1024 caracteres, que luego utilizo en las funciones 'objetos/interacciones crear desde string' de las bibliotecas objetos.h e interacciones.h. Luego, de asignar la información a los campos de las esctructuras de objetos y de interacciones respectivamente, se utilizan las funciones 'agregar objeto' y 'agregar interaccion' para agregar la información de los structs a una nueva posición de los vectores. Por último, se vuelve a iterar el proceso de obtener una nueva linea hasta que se termine el archivo. Para imprimir la lista de objetos en la terminal, se utiliza un vector de strings que se carga en cada posición con el campo "nombre"del vector de objetos creado previamente. En el caso de las interacciones se utiliza la función 'sala es interaccion valida' para verificar si las interacciones hardcodeadas en el main se imprimen con la etiqueta 'Válida' o 'Inválida'.

Para compilar se utilizó la línea:

gcc src/*.c escape$_p$*okemon.c* − *oescape*$_p$*okemon* − *g* − *O*0 − *std* = *c*99 − *Wall* − *Wconversion* − *Wtype* − *limits* − *Werror*

y para ejecutar el programa:

./escape$_p$*okemonejemplo/objetos.txtejemplo/interacciones.txt*

3.1. Detalles de 'objeto/interaccion crear desde string'

Estas funciones se encargan de separar la información del string recibido y asignarla a los campos de las estructuras de objetos e interacciones auxiliares utilizando sscanf para ser devueltos al final de la función. En la función de objetos se guarda la última parte del string en una variable

llamada 'flag' para verificar si la misma es 'true' o 'false'. En el caso de que no lo sea, la función devuelve NULL y se libera la memoria. En el caso de que cumpla la condición, se determina si es 'true' o 'false' y se modifica el campo bool 'es asible' acordemente. En el caso de la función de interacciones, si el campo 'objeto parametro ' llega a contener un 'guion bajo', se le asigna un string vacío al mismo. Por otro lado, se guarda la informacíon recibida luego del último ; en una variable llamada 'accion' para luego utilizar sscanf y asignar cada valor a su campo del struct interacciones correspondiente. El caracter obtenido en la variable char 'tipo accion' se utiliza para verificar y asignar el tipo de dato enum correspondiente.

3.2. Detalle de 'agregar objeto/interaccion'

Estas funciones reciben un vector incializado en NULL, lo que hace que en el realloc que se encuentra en la primera línea de la función funcione como un malloc en la primera iteración. En caso de que se falle al asignar memoria, se cierra el archivo de objetos e interacciones respectivamente. Luego, se le asigna la información del struct recibido a la posición del vector recibido determinada por el valor de la cantidad de objetos o interacciones que corresponde, la cual se aumenta un valor al final de cada iteración. La memoria asignada en estas funciones de libera en al final del programa en la función 'destruir sala', en la cual se recorren ambos vectores usando la cantidad de objeto y la cantidad de interacciones como tope.

4. Diagramas

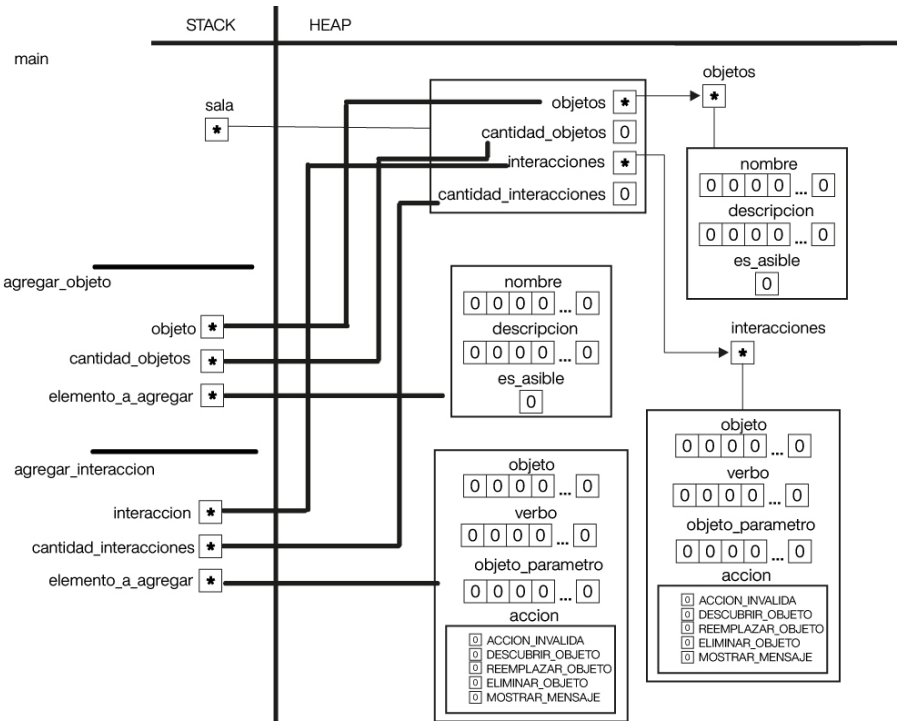

Figura 1: Funciones de agregar objeto e interacción al vector.

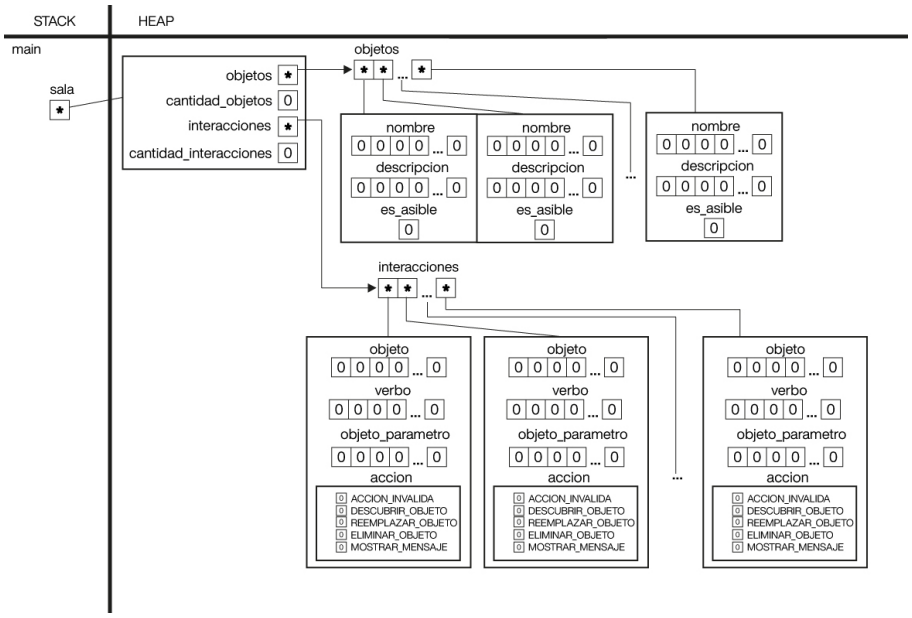

Figura 2: Vectores de objetos e interacciones completos.