FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Trabajo Práctico n1 - Escape Pokemón

[7541/9515] Algoritmos y Programación II

Primer cuatrimestre de 2022

Alumno: Martín Abramovich
Padrón: 108762
Email: mabramovich@fi.uba.ar

1. Introducción

El TP1 “Escape Pokemón” consiste en crear una sala de escape a partir de dos archivos de texto, uno de objetos y otro de interacciones. El objetivo es representar su información en memoria a través de distintas funciones que se implementan. Se solicita liberar toda la memoria correctamente.

2. Detalles de Implementación

Mi implementación del programa comenzó por el desarrollo de *objeto.c* en la que a partir de un string cuya estructura debe ser NOMBRE_OBJETO;DESCRIPCION;FLAG devuelve un puntero a un struct objeto con la información del string leído.
Luego, seguí con el desarrollo de *interaccion.c* que hace lo mismo pero la estructura del string que lee es NOMBRE_OBJETO;VERBO;NOMBRE_OBJETO2;ACCION siendo "accion" un string cuya forma es TIPO_ACCION:NOMBRE_OBJETO:MENSAJE. Devuelve un puntero a un struct interacción con la información proveniente del string.

La razón por la que arranqué por *objeto.c* e *interaccion.c* fue porque entendí que iba a necesitar llamar a las funciones desarrolladas en los mismos cuando desarrollara las funciones de *sala.c*

Después sí continué por las funciones solicitadas en el *sala.h* en el orden en que aparecían allí. Corroboré que necesitaba las funciones *objeto_crear_desde_string* e

interaccion_crear_desde_string.

Una vez desarrolladas las funciones solicitadas, desarrollé el escape_pokemon.c donde se encuentra el main, para el que utilicé las funciones previamente desarrolladas. Allí se reciben por parámetro dos nombres de archivo, el primero de objetos y el segundo de interacciones, y se verifica que la cantidad de parámetros recibidos (argc) sea la correcta. Luego se crea con los archivos una sala de escape y se muestran por pantalla los objetos leídos del archivo y la validez de 4 interacciones diferentes entre objetos.

- ***Lectura de Archivos***

La misma la realicé en la función *sala_crear_desde_archivos.* Para leer los archivos, comencé por abrirlos con *fopen.* Luego, continuo por una verificación que retorne error en caso de no poder abrirlo. Utilicé *fgets* para ir leyendo cada linea de los correspondientes archivos y esta linea, que guardo en mi variable *char linea[MAX_LINEA]* es el string que paso por parámetro a las funciones desarrolladas en *objeto.c* e *interaccion.c* que va iterando hasta que el archivo termina. Al terminar de leer los archivos, cierro ambos con *fclose.*

- ***Manejo de Memoria Dinámica***

Usé mallocs y reallocs para reservar memoria dinámica del tipo de dato que fuera necesario según la ocasión. Me aseguré de verificarlos luego de usarlos y de liberar la memoria con *free* en caso de error. Además de liberar la memoria en *destruir_sala*.

Para compilar el programa utilicé la siguiente linea con los siguientes flags:

```
gcc -g -O1 -std=c99 -Wall -Wconversion -Wtype-limits  -Werror src/*.c escape_pokemon.c -o prueba
```

Y luego para ejecutarlo:

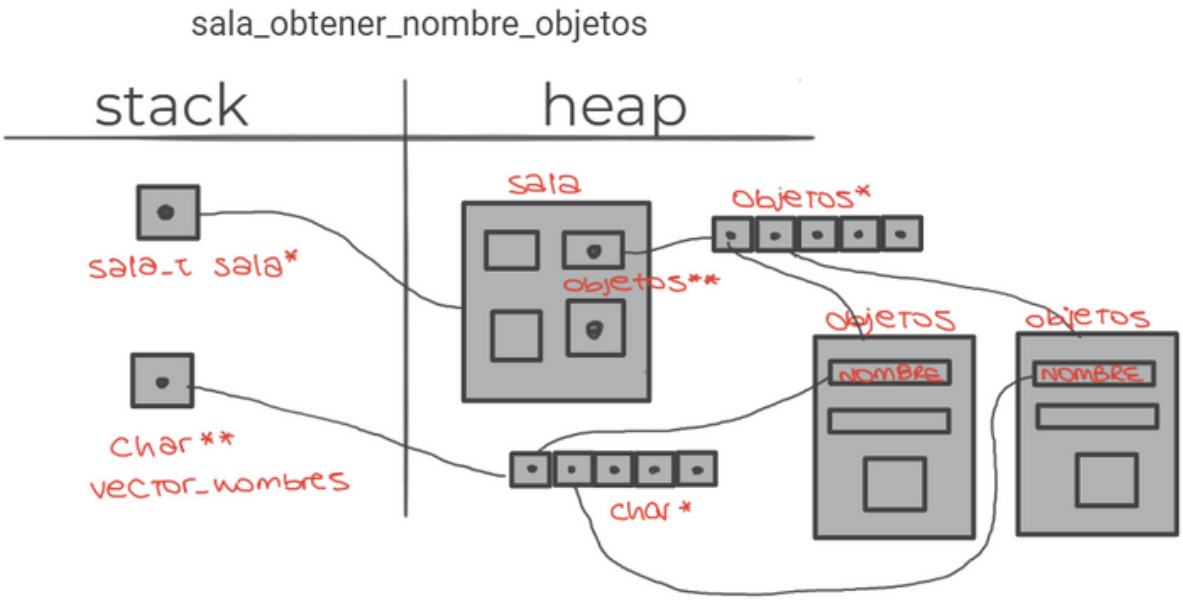
```
./pruebas ejemplo/objetos.txt ejemplo/interacciones.txt
```

3. Diagramas

3.1. sala_obtener_nombre_objetos.

Una de las funciones a implementar para el trabajo es *sala_obtener_nombre_objetos.* La misma recibe por parámetro un puntero a una sala_t y un puntero a un entero. Devuelve un vector dinámico reservado con malloc que contiene los nombres de todos los objetos existentes en la sala de escape.
Para implementarla, lo primero que hice fue verificar que la sala no sea NULL, y en caso de serlo, la función retorna NULL.

El diagrama adjunto grafica lo que hace la función: declaré *vector_nombres,* que es un puntero doble a char. Ahí usé *malloc* para pedir memoria suficiente para generar un vector de punteros a char del tamaño de la cantidad de objetos que hay en la sala. De ser reservado sin errores, se carga el vector con punteros a los nombres de los objetos. La función retorna el puntero doble *vector_nombres*, que tiene las direcciones de los nombres de los objetos.



3.2. main