Marcos Bianchi - 108921

Compilación y Ejecución

El compilador lee el programa escrito por el programador y crea un archivo compilado que se ve así:

```
1   7f45 4c46 0201 0100 0000 0000 0000 0000
2   0300 3e00 0100 0000 e011 0000 0000 0000
3   4000 0000 0000 0000 f865 0000 0000 0000
4   0000 0000 4000 3800 0d00 4000 2400 2300
5   0600 0000 0400 0000 4000 0000 0000 0000
6   4000 0000 0000 0000 4000 0000 0000 0000
7   d802 0000 0000 0000 d802 0000 0000 0000
8   0800 0000 0000 0000 0300 0000 0400 0000
9   1803 0000 0000 0000 1803 0000 0000 0000
10  1803 0000 0000 0000 1c00 0000 0000 0000
```

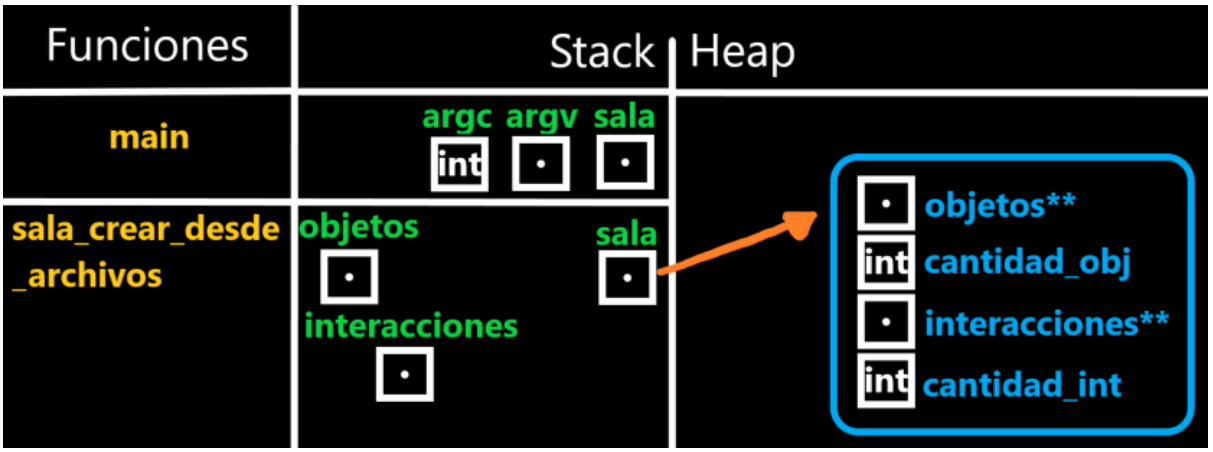
Este texto tiene las acciones que debe cumplir el procesador. De una forma representan las instrucciones que nosotros desde el código que escribimos queremos que siga el programa. Por ejemplo esos primeros 3 segmentos podrían ser los parámetros que le llegan al main y luego el 0100 podría representar una instrucción o declaración de variable.

Explicacion del Codigo

1. Se corre el programa y por parámetro se mandan los nombres de los archivos de (1)objetos e (2)interacciones en este orden.

2. Se crea un puntero de tipo struct sala que apuntará hacia el resultado de la función sala_crear_desde_archivos.

3. Se verifican los parámetros enviados a sala_crear_desde_archivos y si todo está en orden se almacena en la memoria espacio para almacenar una estructura de tipo struct sala y se envían los archivos a la función archivos.



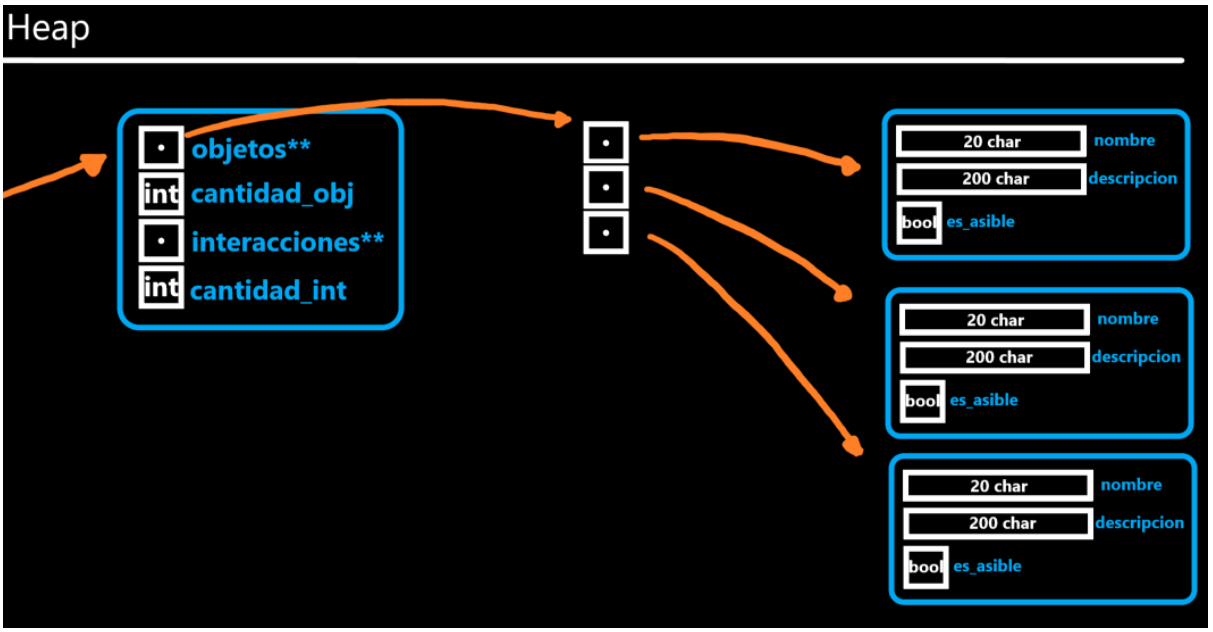
al inicializar la sala con calloc, la función llena los espacios con 0s, esto carga cantidad_objetos y cantidad_interacciones en 0 y a su vez inicializa los vectores de objetos e interacciones apuntandolos a NULL.

4. Ya en la función archivos que se encarga de abrir y cerrar los archivos, en el medio llama a 2 funciones más para que extraigan los contenidos de los archivos para cargarlos en la sala. Recibe por parámetro un puntero a sala y ambos nombres de los archivos.

5. Las funciones objetos arch e interacciones arch funcionan de la misma manera. Ambas reciben su respectivo archivo y un puntero a la sala. Se crea una variable de tipo vector de char llamada “linea” para guardar en ella la línea leída del archivo. Utilice la función fscanf() para leer un string hasta encontrarme con un “\n” y así leer una linea del archivo. En otra variable llamada “leido” me guardo la cantidad de variables leídas, en este caso solo quiero leer la una línea de texto y guardarla como string en “linea” siendo solo 1 variable. Dentro de un while se llama a la función “agregar_(elemento)” y luego se lee otra línea de texto utilizando otro fscanf, guardando la línea de texto en “linea” y la cantidad de variables leídas en “leido”. Una vez que se termina el archivo, la variable “leído” se va a cargar con un 0 y el while se corta determinando que no se deben crear más elementos.

6. Las funciones agregar_objeto y agregar_interaccion tambien funcionan de la misma manera pero agregan su respectivo elemento a su respectivo vector en la sala. La funcion recibe un puntero a la sala, la cantidad de objetos o interacciones y la linea de texto con el elemento a agregar. Primero se manda la linea de texto a la funcion “(elemento)_crear_desde_string” y se guarda en un puntero. Se almacena memoria reservando tamaño para un objeto o interacción más en una variable auxiliar que luego se utilizara para almacenar su contenido en la sala.

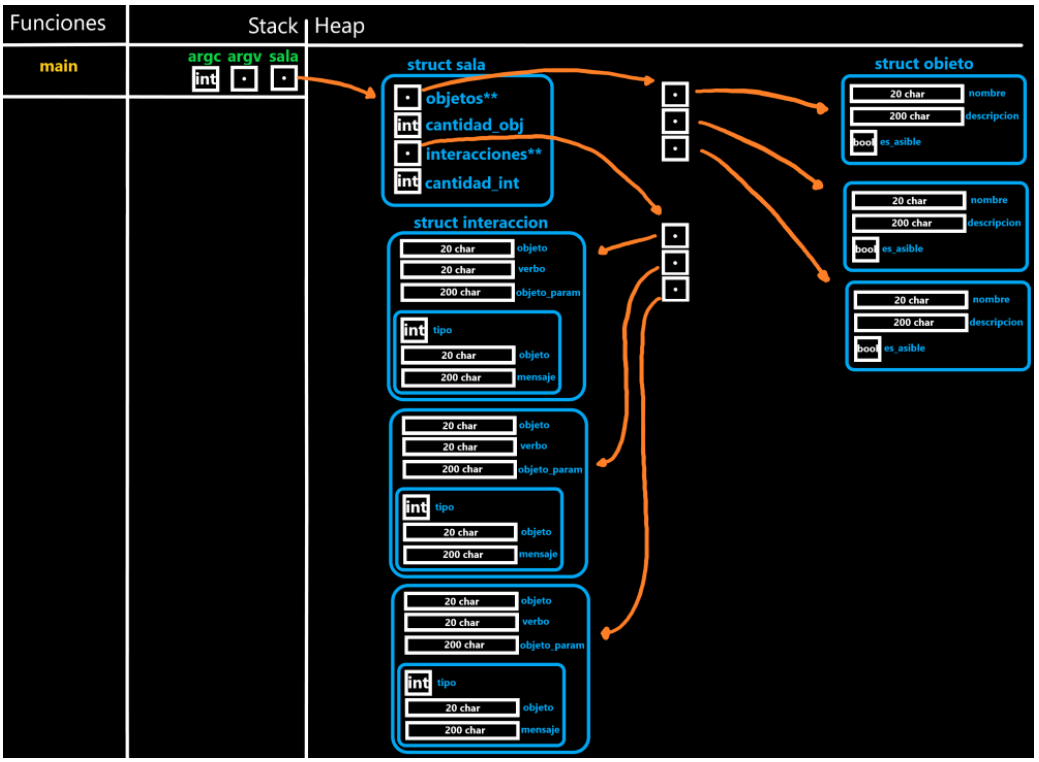
7. Las funciones objeto_crear_desde_string y interaccion_crear_desde_string funcionan también de la misma manera. Crean espacio para su respectivo elemento en la memoria y lo llenan con la línea leída del archivo. Utilice el sscanf para leer la línea y la almaceno en el elemento que quiero agregar.



Esta foto muestra el vector de objetos cargado con 3 objetos.

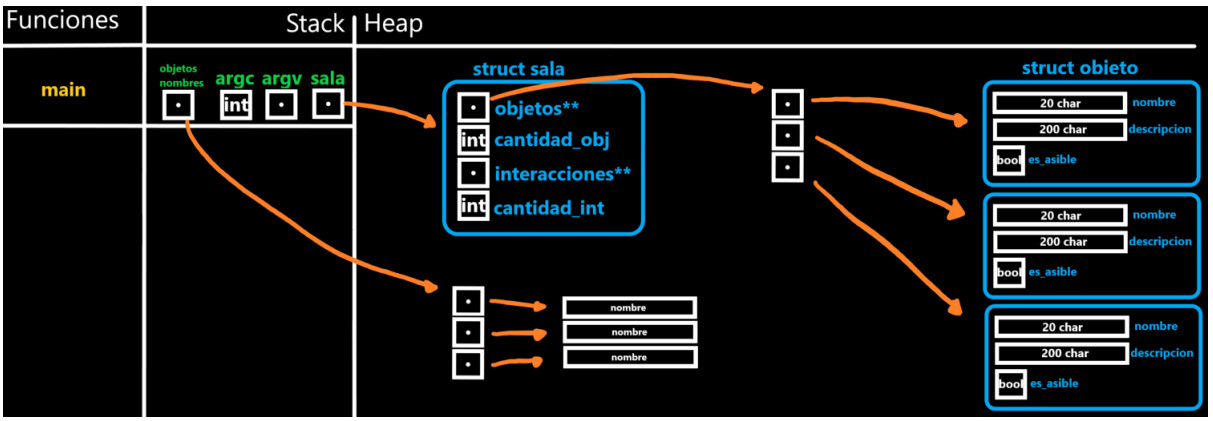
Este proceso se repite al crear las interacciones. Se agranda el vector de interacciones y de la misma manera se van agregando interacciones hasta que se terminan las líneas del archivo de interacciones.

8. Una vez terminado el proceso de creación de la sala, el puntero creado al principio en el main ahora apunta hacia la sala.



En esta foto dibuje una sala de ejemplo con 3 objetos y 3 interacciones.

9.Para el vector de nombres de objetos llamo a la función sala_obtener_nombre_objetos que guarda un espacio en la memoria igual a char* multiplicado a la cantidad de objetos presentes en la sala, llena los strings de nombres con los respectivos nombres de los objetos. Aparte también cambia la cantidad pasada por parametro a la cantidad de objetos en la sala.



En esta foto ilustré el vector de nombres de objetos con 3 objetos.

10. En cuanto a la función de sala_es_interaccion_valida, esta recibe un verbo y 2 objetos. La función compara los parámetros de entrada a las interacciones guardadas en la sala, en caso de encontrar la exacta interacción entre estos elementos devuelve true y en caso contrario false.

11. Los free están ordenados de manera de poder liberar toda la memoria utilizada por el programa. Primero se libera el vector de nombres ya que no puedo llamarlo con la función

destruir_sala, segundo libero todos los objetos por separado y justo después el vector de objetos, lo mismo hago con las interacciones liberando memoria de cada una por separado y luego la del vector de interacciones. Por último libero la memoria utilizada por la sala y así el programa finaliza de manera correcta.