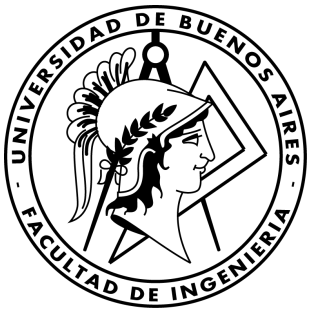

Trabajo Práctico 02
Sala de escape Pokemon

[7541/9515] Algoritmos y Programación II

Primer cuatrimestre de 2022

Autor:	**Email**:	**Padrón**:	**Fecha**:
Luca Lazcano	llazcano@fi.uba.ar	107044	03/07/2022

Índice

1. Introducción

En esta segunda parte del Trabajo Práctico Escape Pokemon, se termina lo comenzado en el TP01 para realizar un juego terminado, jugable desde la consola.

En la implementación se usan los TDA vistos a lo largo del curso.

2. TDA utilizado

Para la manipulación de la información del programa se utilizó el TDA lista realizado previamente. Se utilizaron los archivos `lista.h` y `lista.c` sin modificar, incorporados al directorio `src`.

Se crearon algunas funciones extra que idealmente deberían ser parte del TDA lista (como buscar y devolver un elemento dado un término de búsqueda y un comparador). Estas funciones fueron agregadas a la implementación y extienden la funcionalidad del TDA lista.

Considero que el TDA lista es más que suficiente para esta aplicación, ya que permite realizar las operaciones necesarias: agregar y quitar elementos; eficaz y eficientemente.

No es necesaria la optimización del TDA hash. Discriminar los objetos en claves y elementos, para utilizar el TDA se consideró que no vale la pena. Es cierta que hash tiene algunas funciones útiles que no tiene lista, como buscar un elemento (por ejemplo para buscar 'pokebola', utilizando esto como clave, el hash 'hashea' esto y de forma inmediata devuelve el elemento; mientras que la lista debe iterar y comparar hasta encontrar el elemento).

Tampoco existe una propiedad de 'orden' claramente establecida que justifique utilizar el TDA árbol.

3. Pruebas que fallan

Al correr el archivo de pruebas original, no es posible ejecutar las pruebas de caja blanca, ya que al cambiar las estructuras de los datos utilizados, las pruebas de caja blanca intentan acceder a campos que ya no existen.

Para evitar estos problemas, conviene atenerse estrictamente a las funciones provistas al usuario por el archivo .h.

4. Compilación y ejecución

Se provee un `makefile` con las siguientes opciones:

- `make`: compila todos los archivos necesarios para jugar y ejecuta el juego con valgrind. Al ejecutar `make` comenzará el juego.
- `make pruebas` y `make pruebas_chanu`: compila los archivos de pruebas
- `make valgrind-pruebas` y `make valgrind-pruebas_chanu`: ejecuta las pruebas con valgrind.
- `make juego_con_ncurses`: compila la demo de la interfaz del juego, pensada para originalmente la interfaz del juego. Se elabora en la sección correspondiente.

5. Interfaz original del juego

Si bien el trabajo fue entregado con una interfaz básica, basada en texto, que permite interactuar con el programa y jugar, originalmente se había comenzado a implementar una interfaz gráfica, en la consola, más compleja, con la librería `ncurses` de Linux.

Se puede compilar y correr esta demo con:

```
make juego_con_ncurses && ./juego_con_ncurses
```

Es necesario tener instalada la librería. En la mayoría de las distribuciones de Linux se puede instalar con:

```
sudo apt-get install libncurses5-dev libncursesw5-dev
```

Ejecutando el programa obtenemos la siguiente interfaz en la propia consola:

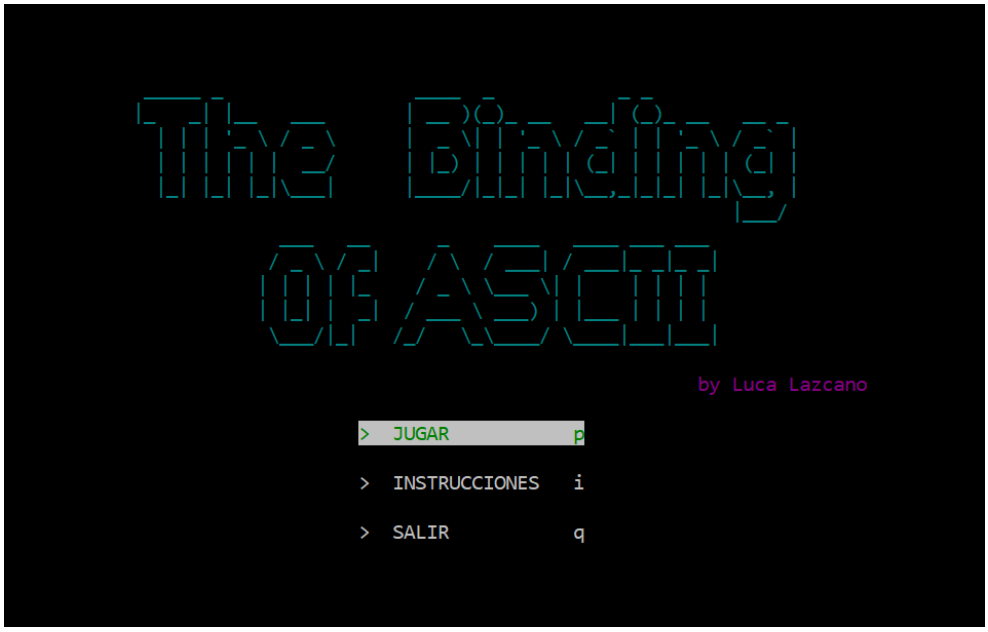

Figura 1: Menú del juego.

En este menú se puede navegar con las flechas arriba y abajo, o seleccionar la opción con el caracter indicado.

Entrando al juego se obtiene la siguiente pantalla:

Figura 2: Interfaz del juego.

Esta sección todavía se encuentra en construcción, pero se pueden ver las principales secciones de la interfaz, y posee cierta funcionalidad.

Presionando las flechas arriba, abajo, izquierda o derecha, podremos mover al personaje, que cambiará de apariencia según la dirección en la que se mueva.

El 'mapa' de la sala fue harcodeado en el propio archivo .c para esta demo, pero la idea era que se puedan leer de archivos y así seleccionar distintos escenarios y niveles.

La sección inferior representa el inventario del usuario, y el objetivo era que refleje los distintos elementos que posee el jugador, ya se por texto o 'ASCII art'.

La sección derecha es la 'consola' o interfaz mediante la cuál el usuario puede comandar a su personaje para que realice acciones, recibir mensajes del juego u otros comandos, como por ejemplo salir u obtener ayuda.

Se accede a esta sección pulsando '.', con lo cual aparecerá una barra para escribir los comandos. Al darle 'enter', la interfaz reflejará lo escrito por el usuario. Con el juego terminado, esta consola debería tomar los comandos para interactuar con `texttt.h` y realizar las interacciones, y demás comandos que permite el trabajo realizado.

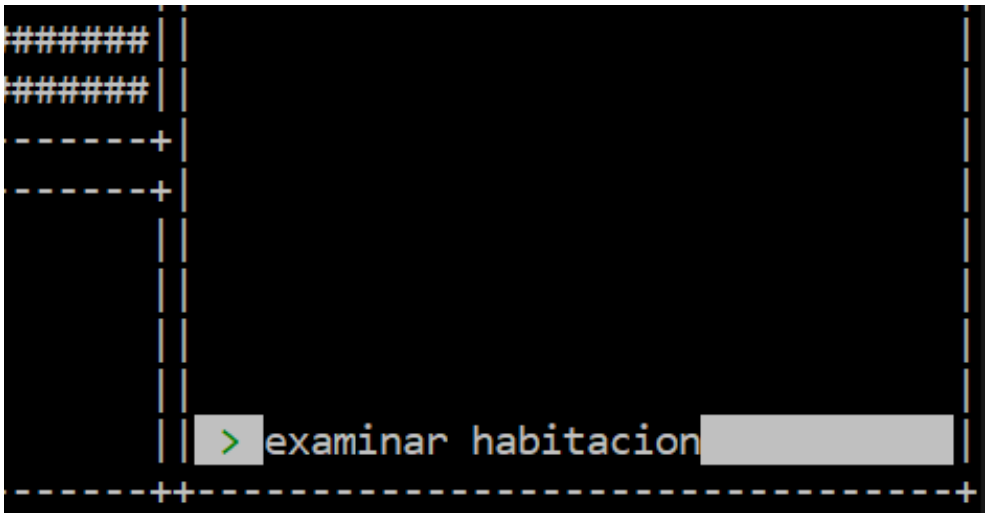

Figura 3: Consola.

Lamentablemente, esta conexión entre `sala.c` y el juego no se llegó a realizar, por lo que enviar un comando... no hace nada.

No se llegó a concretar ni siquiera un 'producto mínimo viable' que funcione como interfaz del programa, por lo que esto no es más que una demo de lo que se quería, y no se pudo realizar.

Otras cosas que faltan son detección de colisiones para restringir el movimiento del personaje, o detectar su proximidad a objetos y recibir mensajes del juego o poder interactuar con objetos. También faltan agregar los objetos visualmente a la sala, o los objetos en el inventario. Otras funcionalidades deseadas incluyen: un nivel de 'ejemplo' para aprender a jugar; la posibilidad de cargar otros escenarios o que existan varias salas encadenadas para que hayan varios niveles.

Figura 4: :peepo sad: