Wordle9511, ejemplo de juego sobrediseñado con TDAs

Sebastián Santisi

2022-10-01

El presente ejemplo intenta replicar un ejemplo de diseño esperable para un TP de 95.11 Algoritmos y Programación I, para un juego relativamente sencillo.

En este ejemplo se explora la encapsulación en TDAs, la delegación de responsabilidades, el chequeo de errores, la utilización de tablas de búsqueda, separación en capas de entrada-procesamiento-salida, pasaje de mensajes, etc.

El ejemplo fue elegido por ser un ejemplo concreto de interacción pero poco representativo de un trabajo práctico real. Es decir, se exponen los temas que se pretende ser aplicados pero no se condiciona la estructura de diseño del trabajo práctico real que corresponda a determinado cuatrimestre, quedando ese análisis y extrapolación desde el presente ejemplo a cargo del alumno.

En base a esta premisa y tanto por motivos pedagógicos o para no exponer una implementación real hay TDAs que se prefirieron no implementar (como por ejemplo un vector dinámico genérico) o estructuras sencillas que se encapsularon como TDAs cuando esto es una sobreingeniería al problema (como por ejemplo el TDA letra).

Uso

Para compilar [1]:

```
$ gcc *.c -o wordle9511 -Wall -std=c99 -pedantic
```

Para ejecutar:

```
$ ./wordle9511
```

si se quiere repetir una partida puede ejecutarse como:

```
$ ./wordle9511 <semilla>
```

Wordle

El Wordle es un juego que consiste en adivinar una palabra de 5 letras del idioma castellano a partir de intentar 6 veces con palabras también del idioma castellano. En cada intento se indican qué letras de la palabra intentada coinciden con las palabras de la palabra secreta y se indica si la posición del intento es correcta o no. Cada una de las pistas son independientes entre sí, es decir, si por ejemplo hubiera una A en la posición correcta, esto no quita que tal vez la palabra secreta tenga otra letra A en otra posición.

Estructura y diseño

Se tiene un `main.c` que implementa la lógica de alto nivel. El Main instancia todos los objetos que participan del juego y se encarga de disparar toda la lógica de entrada-salida, si bien la delega. No hay nada que se realice en el programa sin que el Main no lo ejecute de forma explícita.

[1] Al respecto de Makefile ver sección "A mejorar".

Hay un TDA `diccionario_t` el cual contiene todas las palabras válidas. El Diccionario es instanciado por el Main y de él el Main elige la palabra a adivinar. Cuando se realiza la entrada del usuario la misma se valida contra el Diccionario para sólo permitir el ingreso de palabras correctas.

La entrada y salida del programa está encapsulada en dos módulos; Un módulo, `ansi`, contiene las implementación para controlar los colores de la terminal mediante secuencias de escape ANSI, este módulo define la lista de colores válidos. El otro módulo, `interaccion`, posee las rutinas de lectura y de impresión. Ningún otro módulo del proyecto lee o escribe, todos delegan en Interacción la escritura. El módulo ANSI sólo es consumido por Interacción, quedando invisible para el resto de la aplicación.

El TDA `letra_t` (el cual, como ya se dijo, está innecesariamente encapsulado) representa el estado de una letra dentro de la lógica del juego. Tanto para el alfabeto de letras adivinables como para un intento particular una letra puede haber sido ya arriesgada, puede ser inválida, puede estar en la posición correcta o no. El TDA letra le sirve exclusivamente a la lógica del Wordle, nunca es usado por la lógica de alto nivel del Main, al Main le importa el estado del juego, no el detalle de cómo se implementa. Como el estado de las letras debe ser impreso, la capa de Interacción conoce cómo imprimir a las Letras. Incluso tiene un par de tablas de búsqueda para traducir los estados de las Letras en un código de colores.

El TDA `wordle_t` es el que contiene el estado del juego. Cabe destacar que en un programa más complejo, con más interacción entre TDAs diferentes, listados de objetos, complejidad, etc. la lógica de negocios de alto nivel bien puede estar implementada en el Main sin necesidad de ser encapsulada. La realidad es que no hay reutilización posible, el Wordle sólo sabe jugar al Wordle y si cambia el juego ya no sirve más. En este diseño propuesto el Wordle se instancia a partir de la palabra secreta y es este TDA el que se encarga de validar el estado del juego. El Main va a ser el encargado de pasarle las palabras que el usuario intente adivinar al Wordle y el estado de este TDA será el que indique el progreso del juego. Cabe destacar que en el diseño propuesto es el Main el que verifica que tanto la palabra secreta como los intentos pertenezcan al Diccionario; el Wordle sólo se limita a verificar el intento contra la palabra secreta. El Wordle es el que conoce el estado de todas las Letras tanto del alfabeto como de cada uno de los intentos, ahora bien, sólo dispara la impresión de las mismas si el Main se lo pide, pero lo delega siempre en Interacción que es quien sabe imprimir.

Finalmente el proyecto se completa con un par de módulos auxiliares, un archivo de configuración global `config.h` que tiene etiquetas que le interesan a todos los módulos o particularmente al Main y con un módulo Utils el cual tiene alguna función auxiliar de uso común que no tiene nada que ver con la lógica del juego en sí.

A mejorar

- Notar que el módulo del Wordle asume que las palabras están compuestas por caracteres de la `'A'` a la `'Z'`, sin embargo eso no se valida en ningún lado y Diccionario no sólo no implementa validaciones si no que tampoco conoce esta restricción.
- En la documentación habría que indicar como precondición dónde se está asumiendo que las cadenas recibidas son palabra válidas, es decir, de `LONGITUD_PALABRA` y pertenecientes al Diccionario.
- Como se implementó Letra como un TDA y no se estableció un setter para cambiar el caracter asociado, el Wordle no puede instanciar las letras de un intento si no hasta el momento en el cual el usuario arriesga una palabra. Esto hace que la primitiva `wordle_procesar_intento()` tenga que pedir la memoria para las Letras ingresadas. Sería preferible que toda la memoria se pidiera en el constructor del TDA y que no hubiera posibilidades de falla de memoria al procesar el intento. Esto se solucionaría solo si la Letra no fuera un TDA y fuera sólo una estructura global o si tuviera un setter para el caracter.
- El manejo de memoria en los arreglos quedaría muchísimo más encapsulado y sencillo si se implementara un TDA Vector. Simplificaría notablemente la lógica del Diccionario y del Wordle.
- Hace falta un Makefile.