Taller 1: Wordle

TD3: Algoritmos y Estructuras de Datos

En este taller individual vamos a programar una versión simplificada del juego Wordle en C++.

1 Fecha y formato de entrega

- La fecha límite de entre es el **domingo 30 de marzo**.
- Se entrega via campus.
- Deben adjuntar únicamente los archivos `wordle.cpp` y `lib.cpp` con la implementación completa.

2 Reglas del juego

- El objetivo del juego es adivinar una palabra secreta de 5 letras en español. Para eso se tienen 6 oportunidades para intentar ingresar la palabra correcta.
- Luego de cada intento el juego indica, por cada letra de la palabra ingresada, si la palabra secreta contiene esa letra en esa posición, si la palabra secreta contiene esa letra pero en otra posición, o si la palabra secreta no contiene esa letra.
- Con esta información se puede realizar un nuevo intento hasta o bien adivinar la palabra secreta o bien agotar los 6 intentos.

3 Proyecto C++ adjunto

El archivo zip contiene un esqueleto del proyecto C++ sobre el que van a trabajar y la configuración del container para que lo puedan abrir el VSCode con el método usual descrito en otro apunte.

A continuación ofrecemos una breve descripción de los archivos que componen el proyecto:

- `lib.h`: encabezado que declara un conjunto de funciones y tipos de datos necesarios para el juego.
- `lib.cpp`: archivo con implementación de las funciones declaradas en `lib.h`
- `wordle.cpp`: este archivo contiene la función `main` que debe utilizar las funciones declaradas en `lib.h` para implementar el juego interactivo.
- `listado.txt`: listado de palabras en español de cinco letras que se debe utilizar como diccionario del juego. **Por simplicidad sólo vamos a utilizar palabras sin letras repetidas, sin tildes y sin mayúsculas.**
- `tests/`: directorio que contiene casos de test escritos por la cátedra para que verifiquen sus implementaciones. Utiliza el framework Google Test (gtest).
- `CMakeLists.txt`: configuración de CMake. Define las opciones que VSCode ofrece para compilar y ejecutar el programa, incluyendo sus tests.

4 Consigna

Para empezar, deben implementar en `lib.cpp` todas las funciones declaradas en `lib.h`. La declaración de cada función tiene un comentario asociado describiendo su comportamiento esperado. De ser necesario, en la implementación de cada función deben escribir comentarios que ayuden a entender la forma en que está resuelta.

Luego deben implementar la función `main` de `wordle.cpp`, que utiliza las funciones implementadas anteriormente para definir el comportamiento interactivo del juego:

- El juego debe pedir al usuario que ingrese la ruta del archivo de palabras a utilizar (por ejemplo, el del archivo `listado.txt` provisto).
- Cargar el listado de palabras del archivo y, en cada ejecución, elegir una de estas palabras al azar para utilizar como palabra secreta. Para obtener un numero pseudo-aleatorio pueden utilizar la función `rand()` de la biblioteca de C: https://en.cppreference.com/w/cpp/numeric/random/rand.
- Luego se le debe pedir al usuario que haga un intento para adivinar la palabra secreta.

Ante cada intento, el programa reacciona de la siguiente manera:

- Si el intento es válido se debe evaluar el intento y darle la respuesta al usuario por pantalla. Esto le resta un intento al usuario para adivinar la palabra.
- Si el intento no es válido (es decir, no sea legal) se debe indicar por pantalla que el intento no es valido e invitar al usuario escribir un nuevo intento. En este caso no se le resta al usuario una oportunidad.
- Si el usuario adivina la palabra el juego termina y se debe indicar por pantalla que el usuario adivinó.
- Al agotar los cinco intentos el juego termina y se debe indicar por pantalla que el número de intentos se agotó.

A continuación vemos un ejemplo de ejecución exitosa:

```
$ /workspaces/wordle/build/wordle
Ruta al listado de palabras: ../listado.txt
La palabra secreta tiene 5 letras.
Te quedan 5 intentos
dejar
X--XX
Te quedan 4 intentos
subir
XXX-X
Te quedan 3 intentos
viejo
¡Correcto!
```

Notar que el path al archivo ingresado por el usuario es relativo al directorio desde el cual se **ejecuta** el programa. En este caso, VSCode generó y ejecutó el programa `wordle` dentro de un subdirectorio `build`, por lo que usamos `../` para indicar que `listado.txt` está en el directorio padre de `build`.

5 Tests

El proyecto CMake está configurado para que sus funciones se testeen automáticamente según una suite de pruebas definidas por la cátedra.

Su implementación debe pasar todos los tests satisfactoriamente para poder aprobar, incluso aunque no haya errores obvios en la interfaz gráfica del programa interactivo `wordle`.

Los tests pueden ejecutarse desde la interfaz gráfica siguiendo las instrucciones estándar de la cátedra para nuestros entornos de desarrollo, y corriendo el ejecutable `tests` generado por CMake. **Se recomienda**

fuertemente ir corriendo los tests de cada función a medida que van trabajando para agilizar el proceso de trabajo y evitar "arrastrar" errores.