Elías Rojas
Comisión 2 AUS

Documentación TP Final Taller I

Índice

Elías Rojas
Comisión 2 AUS

Objetivo

El objetivo de este trabajo es programar el juego WORDLE, el mismo consiste en adivinar una palabra de 5 letras en 6 intentos. Luego de un intento, se dan pistas sobre la palabra a adivinar basandose en la la palabra que fue ingresada previamente.

- Las letras en color GRIS representan letras que no pertenecen a la palabra.
- Las letras en color AMARILLO representan letras que pertenecen a la palabra, pero no estan en el orden correcto.
- Las letras en color VERDE representan letras que pertenecen a la palabra y están en el orden correcto.

El usuario deberá guiarse con las pistas provistas por el programa para poder poder adivinar la palabra misteriosa en la menor cantidad de intentos. En caso de no lograrlo, el usuario perderá.

El programa en primera instancia, poseerá las siguientes características:

1. No tendrá interfaz gráfica.
2. Poseerá un sistema de puntajes (Página 2).
3. El usuario elegirá participar en una sesión de juego y especificará la cantidad de los partidas (máx. 8)
4. Antes de iniciarse una partida, se indicará el número de la partida y el total. Ej: Partida 3 de 5
5. Las palabras :
    - No se repetirán en una sesión de juego.
    - No repetirán letras, para simplificar el programa.
    - Serán tomadas de un archivo externo.
    - Pertenecerán al idioma español.
    - Contarán con 5 letras.

Elías Rojas
Comisión 2 AUS

Sistema de puntaje:

- El usuario inicia con 5000 puntos
- Si acierta la palabra en el primer intento, gana la partida con 10000 puntos
- A medida que realice intentos sin lograr descubrir la palabra, es decir use otra fila, se le descuenta 500
- Ademas, en cada intento suman 50 puntos las letras **nuevas** acertadas (en lugar incorrecto) y 100 las **nuevas** letras ubicadas correctamente
- Finalmente al ganar recibe 2000 puntos adicionales. Si no logra descubrir la palabra, esa partida tendra una puntuacion de 0

						No ganó aún. Se descuenta 500 por pasar de fila y se le suma **100** por la letra acertada en posición correcta y **50** por la letra acertada en posición incorrecta. Puntaje: **5000** - 500 + 100 + 50 = 4650
1		100		50		
2		0				No ganó aún. Se descuentan 500 por pasar de fila. No acertó letras. Puntaje: 4650 - 500 = 4150
3		0	50		100	No ganó aún. Se descuentan 500. Se le suma **50** por la *nueva* letra acertada en posición incorrecta y **100** por *otra* correcta. Puntaje: 4150 - 500 + 150 = 3500
4	50	0			0	No ganó aún. Se descuentan 500. Se le suma **50** por la *nueva* letra acertada en posición correcta. Puntaje: 3500 - 500 + 50 = 3050
5		0	50		0	No ganó aún. Se descuentan 500. Se le suma **50** por la *nueva* letra acertada en posición incorrecta. Puntaje: 3050 - 500 + 50 = 2600
6						**Ganó! Adivinó la palabra misteriosa**. Se suman **2000** puntos a lo acumulado. Puntaje final: 2600 + 2000 = 4600.

Elías Rojas
Comisión 2 AUS

Practica C

En este ejercicio se utilizaron los siguientes recursos de C:

- Loops
- Funciones
- Macros
- Arrays y manejo de arrays
- Tomar data de archivos de texto externos

Librerias utilizadas:

- <stdio.h> para el input y output del usuario
- <stdlib.h> para la utilizacion de rand y srand en cada ejecucion
- <string.h> para encontrar caracteres en una string con strchr() y poder utilizar getWordInLine
- <unistd.h> para poder utilizar el process id como seed

Elías Rojas
Comisión 2 AUS

Variables y funciones importantes

Variables importantes del main

```
char tracking[WORD_LENGTH-1] = {'-'};//cada numero representara la posicion de cada letra de la palabra de una partida,al acertar/adivinar un caracter se completara con '.' al final de la partida se seteara todo a '-' de nuevo

int scores[MAX_GAMES];//Este array tendra la puntuacion de cada partida donde cada partida esta representada por el numero de la misma, menos uno.

char misterious_word[WORD_LENGTH], guess[WORD_LENGTH], try_feedback[WORD_LENGTH-1];//en misterious_word se almacenara la palabra a adivinar ,en guess el intento y try_feedback tendra el feedback de cada intento
```

Funciones importantes del programa

-Feedback despues de jugada

```
void print_feedback(char feedback[],char word[], char guess[]){
  int i = 0;
  while(i < WORD_LENGTH-1){              //loopeo por toda la string de manera que:
    if( word[i] == guess[i] ){           /*si encuentro caracteres iguales en la
                                       misma posicion de la palabra a adivinar y
                                       el intento, printeo + y completo
                                       feedback[i]*/

      feedback[i] = '+';
      printf("+");
    }else if(strchr(word, guess[i]) != NULL){/*si encuentro que el caracter i del
                                             intento, se encuentra en la palabra
                                             a adivinar, printeo * y completo
                                             feedback[i]*/

      feedback[i] = '*';
      printf("*");
    }else{                                     /* si definitivamente el caracter no
                                             esta en la palabra a adivinar,
                                             printeo '-' y lo pongo en la misma
                                             posicion*/

      feedback[i] = '-';
      printf("-");
    }
    i++;
  }
  printf("\n");
}
```

-Preparar score

```
int set_score(char tracking[],char try_feedback[]){//recibira tracking para ver que letras fueron adivinadas ya, si la letra fue adivinada, no contara en el puntaje porque se marco previamente en tracking. Tambien recibira el intento de la jugada actual.
  int try_score = 0, i;
```

```
  for(i = 0; i < WORD_LENGTH; i++){
    if(try_feedback[i] == '+' && tracking[i] != '.'){//si la pega y es nueva, suma 100 y es marcada en tracking con un punto
      tracking[i] = '.';
      try_score += WELL_PLACED_LETTER;
    }else if(try_feedback[i] == '*' && tracking[i] != '.'){//si casi la pega y es nueva, suma 50 y es marcada en tracking con un punto
      tracking[i] = '.';
      try_score += WRONG_PLACED_LETTER;
    }
  }

  if(user_won(try_feedback)){//si el usuario gana, devuelve 2000
    return WIN_SCORE;
  }else{
    return try_score-PENALTY;// si no devuelve el puntaje de la jugada
  }
}
```

-El usuario gano?

```
int user_won(char try_feedback[]){//si detecta una seguidilla de + en try_feedback, o sea en el intento actual, el usuario gana y devuelve 1, si no devuelve 0
  int i, count = 0;
  for(i = 0; i < WORD_LENGTH-1; i++){
    if(try_feedback[i] == '+'){
      count++;
    }
  }
  if(count >= 5) return TRUE;
  return FALSE;
}
```

Flujo del programa

El flujo del programa, generalizado, es el siguiente:

- Consultar el usuario si quiere jugar y el numero de partidas
- Comienza el juego:
  - El programa elige una palabra de otro archivo de texto y setea el puntaje inicial de la partida
  - Comienza la partida:
    - El usuario ingresa un intento de 5 palabras y se le devolvera el feedback de ese intento.
    - Se actualizara el puntaje de esa partida segun el intento.
    - Si el usuario descubrio la palabra(gana) en el primer intento, esa partida tendra un puntaje de 10000.
    - Si el usuario gana la partida, se le agregan 2000 puntos.
  - Termina la partida
  - Si el usuario quiere dejar de jugar, el juego termina
  - Se imprimen las estadisticas del juego

Compilacion y ejecucion

1. Ir a la carpeta **WordleC**

   ```
   cd WordleC
   ```

2. Compilar el archivo

   ```
   gcc -Wall wordle.c -o wordle
   ```

3. Ejecutar el archivo

   ```
   ./wordle
   ```

El programa por cada intento de obtener la palabra, devolvera lo siguiente:
+ : si la letra esta en la palabra y en la posicion correcta
* : si la letra esta en la palabra y no en la posicion correcta
- : si la letra no esta en la palabra

Se recomienda NO insertar letras en la cantidad de partidas. Caso contrario, cortar el bucle infinito con CTRL+C.

Mejoras futuras

El programa todavia requiere de detalles como:
- Tener un mejor control del input.
- Un mejor control de los bucles, para no estar usando break todo el tiempo, tal vez plantearlo desde otra perspectiva .

Y probablemente varios mas.

Ahora, se le podrian agregar nuevas funcionalidades como:
- Mostrar un mapa/diagrama de todos los intentos de la partida al final de la misma.
- Tiempos limites para adivinar una palabra, para luego restarlos a la puntuacion.
- Colores(o una interfaz grafica con colores en el mejor de los casos).
- Cada printf deberia estar mas prolijo. Se deberia alinear en la consola lo impreso, por ejemplo.