Documentación proyecto final Wordle.

# **Índice**

[Índice 1](#_n23jvkwt4b97)

[Objetivo: 2](#_aj1rxkehpw1u)

[Sistema de puntajes del juego: 2](#_ruhj6yb0hwmc)

[Casos especiales: 3](#_b8mtqlibnte0)

[Opciones para el jugador: 3](#_rjpfbiamewas)

[Practica de C: 3](#_k4lntgg8h0cq)

[Estructura de datos: 3](#_1yi9yhpyxlmw)

[Algoritmo y funciones importantes: 4](#_6lqt0wyr5hgv)

[Bucle principal: ( combinados ) 5](#_s5d5en9owy7v)

[Función importante: 5](#_84aluoovf2x0)

[El jugador gana en el primer intento. 6](#_2xnbgsdxcho5)

[El jugador adivina luego del primer intento. 7](#_tbejod5wrvx5)

[El jugador no adivina nunca y pierde. (este caso no se encuentra anidado) 7](#_tabizqutxtjp)

[En cada caso se evalúa si corresponde si el jugador quiere seguir jugando o no. 7](#_25nzr7itxal)

[Compilacion y ejecucion: 8](#_q6yih91fqk1y)

[- Compilación: Descomprima el archivo.rar. 8](#_rx0539u9apn)

[- Compilación desde consola: Descomprima el archivo.rar. 8](#_ykjlp6yipr9e)

[Ejemplo de ejecución: 9](#_q52j7vckl3tn)

# 

# **Objetivo:**

El objetivo es programar un mini juego, en este caso el Wordle.

El juego consiste en adivinar una palabra secreta, la misma está compuesta por cinco letras, ejemplo; “GATOS”, el jugador deberá adivinar dicha palabra ingresando diferentes palabras que contengan cinco letras, con un máximo de seis intentos, en la medida que se revelan si las letras son o no pertenecientes a la palabra secreta en cada intento, el jugador podrá tomarlo como pistas para adivinar la palabra.

Así también el juego cuenta con un sistema de puntajes y casos especiales.

El juego tiene un máximo de ocho partidas que este es a elección del jugador cuántas quiere jugar, y de forma predeterminada hasta seis intentos.

## Sistema de puntajes del juego:

* El juego inicia con un puntaje de cinco mil puntos.
* Por cada letra acertada en el lugar exacto a la palabra secreta suma cien puntos.
* Por cada letra acertada en el lugar incorrecto pero perteneciente a la palabra secreta suma cincuenta puntos.
* Por cada letra incorrecta no coincidente con la palabra secreta suma cero puntos.
* Por cada intento que el jugador deba ocupar para adivinar, a su puntaje actual se le restan quinientos puntos.

## Casos especiales:

* Si el jugador tiene un golpe de suerte, y adivina la palabra en el primer intento de los seis en total, gana automáticamente la partida con una suma de diez mil puntos.
* Si el jugador adivina la palabra secreta entre los intentos dos y seis inclusive sumará a su puntaje actual dos mil puntos.

## Opciones para el jugador:

* En cada partida finalizada el jugador tendrá la opción de abandonar la partida.
* Podrá elegir la cantidad de partidas que el jugador quiera jugar, máximo ocho.
* El jugador durante sus partidas podrá visualizar el número de su partida actual sobre el total. (Partida 2 de 4).
* Si se quiere, el jugador podrá optar por ver su promedio de puntajes en logro una victoria.
* Así también podrá visualizar los puntajes más bajos y más altos por partida completada.

# **Practica de C:**

En este programa se utilizan los siguientes elementos de programación.

* Directivas de preprocesador: Macros.
* Arreglos unidimensionales.
* Estructura de for anidados.
* Instrucciones de if - else anidados.
* Combinación de bucles while y for.
* Funciones con valor de retorno y void.

El programa solo consiste de un archivo:

* codwordle.c

Total de líneas del programa: 443

# **Estructura de datos:**

Hay un total de siete arreglos unidimensionales que se encuentran en el programa, de estos los más importantes son.

char \*box\_palabra []= {};

Es un arreglo de puntero a caracteres que almacena la lista de las palabras secretas que son las utilizadas por el programa para que el jugador adivine.

int box\_puntaje\_partidas [] = {};

Es un arreglo que se utiliza para guardar los puntajes de cada partida completa.

int box\_puntaje [];

Es un arreglo que contiene los puntajes dedicados en cada intento. (100, 50, 0 pts)

El resto de arreglos son utilizados para almacenar respuestas del jugador, aquellas interacciones que realiza con el programa para poder visualizar puntajes, continuar o no con su partida e ingresar su palabra para su juego.

char p [];

char r\_nuevapartida [];

char resp [];

char mensaje []

# **Algoritmo y funciones importantes:**

Los pasos del programa son los siguientes:

* Se muestra un cartel de bienvenida. ( for anidado para crear el cartel )
* Se muestra un menú de “reglas”. ( se ejecuta en main con los printf )
* Se muestra un panel de configuración, se le pide al jugador la cantidad de partidas que quiere realizar. Para esto hay una función dedicada que retorna el valor que ingresa el jugador y se almacena en una variable int cantidad\_de\_partidas = configuro\_partida ()
* Se selecciona la palabra secreta llamando a la función. (este dentro de while)
* Luego se iteran los intentos hasta seis (for) y en la medida que el jugador ingrese su palabra se pone en evaluación -> si adivina el primer intento -> si adivina entre intentos dos a seis inclusives -> si llega al último intento sin adivinar la palabra.
* En cada intento se llama la función para realizar el cálculo del puntaje y mostrarlo.

Aclaración importante

los comentarios de cada paso del programa se encuentra en el código

## Bucle principal: ( combinados )

Comienza por el bucle while ( partida\_actual <= cantidad\_de\_partidas )

Este controla que se jueguen las partidas. Usando como condición

( partida\_actual <= cantidad\_de\_partidas )

Se inicializa el puntaje inicial, se imprime el indicador de partida, se llama a la función para seleccionar la palabra secreta y se inicializa una variable para controlar cuando el jugador quiera seguir jugando o no.

while ( partida\_actual <= cantidad\_de\_partidas )

{

puntaje = PUNTAJE\_INICIAL;

printf ("\n-------------------->\n");

printf ("Partida: %d --> %d\n", partida\_actual + 1, cantidad\_de\_partidas );

printf ("-------------------->\n");

palabra\_aleatoria ( &xrandom, box\_palabra, &longitud );

int seguir\_jugando = 1;

Luego ingresa el bucle for ( int i = 1; i < MAX\_INTENTOS; i++) {}

Este controla los intentos durante la partida, recordando que el máximo de intentos es seis, su condición se inicia en 1 hasta que sea < MAX\_INTENTOS y este será incrementado i++.

Y también se evalúan los casos (intrucciones if - else anidado) que se mencionan en los pasos generales del programa.

# **Función importante:**

Dentro de for se utiliza la función: int adivina\_palabra ( char \*xrandom, char \*x )

int adivina\_palabra ( char \*xrandom, char \*x )

{

int puntaje\_inicial = PUNTAJE\_INICIAL;

int puntaje\_ganador = PUNTAJE\_PRIMER\_ACIERTO;

int puntaje\_final = 0;

if ( strcmp ( xrandom, x ) == 0 )

{

puntaje\_final = puntaje\_inicial += puntaje\_ganador;

return 1;

} else {

return 0;

}

}

Es importante ya que se usa un valor de retorno 1 o 0. Para poder evaluar los casos (primer intento, gana entre intentos 2 y 6, ayuda en cada intento comprar y calcular, y evalua el caso de perder todos los intentos ) Este valor se usa con las instrucciones if - else para evaluar cada caso. Y en cada bloque de los casos en el momento se actualiza el puntaje que se suma con el puntaje que viene realizando para almacenar el puntaje final por partidas.

## **El jugador gana en el primer intento.**

if ( adivina\_palabra ( xrandom, p ) == 1 && i == 1 )

{

printf ("--------------------\n");

printf ("\n%cAdivino la palabra!\n", 173);

printf ("La palabra secreta era --> %5s\n", xrandom );

puntaje += PUNTAJE\_PRIMER\_ACIERTO;

printf ("Ha sumado --> %d pts, resultado de su partida:%d\n", PUNTAJE\_PRIMER\_ACIERTO, puntaje );

printf ("\n<----------------------------->\n\n");

printf ("%cDesea seguir jugando? Si o No\n", 168);

scanf ("%s", r\_nuevapartida );

printf ("\n--------------------\n");

final\_puntaje = puntaje;

box\_puntaje\_partidas [ partida\_actual - 1 ] = final\_puntaje;

## **El jugador adivina luego del primer intento.**

else if ( adivina\_palabra ( xrandom, p ) == 1 && i > 1 ) {

printf ("-------------------->\n");

printf("\n%cAdivino la palabra en el intento %d!\n", 173, i);

printf("La palabra secreta era: %5s\n", xrandom);

puntaje += PUNTAJE\_ACIERTO;

printf("Ha sumado --> %d pts, resultado de su partida:%d\n", PUNTAJE\_ACIERTO, puntaje); printf ("\n<----------------------------->\n\n");

printf("%cDesea seguir jugando? Si o No\n", 168);

scanf("%s", r\_nuevapartida); printf("\n--------------------\n");

final\_puntaje = puntaje;

box\_puntaje\_partidas [ partida\_actual - 1 ] = final\_puntaje;

## **El jugador no adivina nunca y pierde. (este caso no se encuentra anidado)**

if ( i == 6 && adivina\_palabra ( xrandom, p ) != 1 )

{

printf ("\n-------------------->\n");

printf ("\n No adivinaste la palabra en el intento %d\n", i );

printf ("La palabra secreta era %s\n", xrandom );

printf ("\n|> Puntaje total de la partida: %d\n\n", puntaje );

final\_puntaje = puntaje;

box\_puntaje\_partidas [ partida\_actual - 1 ] = final\_puntaje;

}

## **En cada caso se evalúa si corresponde si el jugador quiere seguir jugando o no.**

if (strcmp(r\_nuevapartida, "SI") == 0) {

cantidad\_de\_partidas++;

break;

} else {

seguir\_jugando = 0;

break;

Finalmente cerramos el for y ahora nos encontramos en las últimas instrucciones dentro de while. Donde preguntamos al jugador si quiere saber sobre sus puntajes y se evalúa si se sigue con la partida o no.

if ( partida\_actual == cantidad\_de\_partidas )

{

printf ("%cDesea ver los puntajes finales de sus partidas?\n", 168 );

scanf ("%s", resp );

if ( strcmp ( resp, "SI" ) == 0 )

{

printf ("Puntajes final de su partida: \n");

for ( j = 0; j < cantidad\_de\_partidas; j++)

{

printf ("Partida %d, %d\n", j + 1, box\_puntaje\_partidas [j] );

}

}

}

if ( seguir\_jugando == 0 )

{

printf ("%cGracias por jugar!. %cHasta la proxima!\n", 173, 173 );

break;

}

# 

# **Compilacion y ejecucion:**

Importante antes de la compilación, el programa fue desarrollado en un entorno de Windows.

### Compilación: Descomprima el archivo.rar.

Acceda al directorio generado.

Ejecute el .exe.

### Compilación desde consola: Descomprima el archivo.rar.

Acceda al directorio generado.

cd /ruta de la descompresión del .rar.

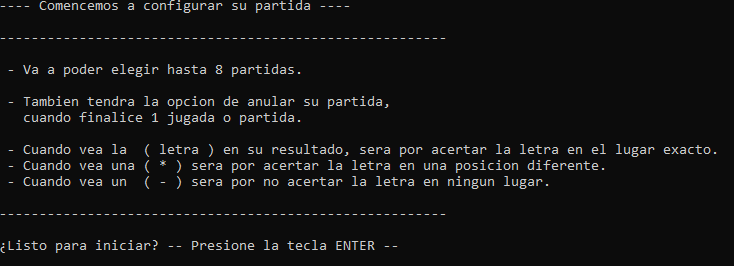
Compile el codigo ( gcc -Wall codwordle.c -o ejecutable )

Ejecute el .exe ( ./ejecutable ).

En caso de ser necesario puede usar este [link](https://www.onlinegdb.com/online_c_compiler) para compilar de manera rápida el código.

## **Ejemplo de ejecución:**





En las siguientes imágenes se puede visualizar una partida realizada.

