Proyecto: Juego Ahorcado

Curso: IE-0117 Programación bajo plataformas abiertas

Grupo: Pokerface

Integrantes:

Erick Vargas Monge C08215

Denisse Ugalde Rivera C07893

Índice

Introducción

El objetivo de este proyecto es programar un juego en un entorno colaborativo en C. El lenguaje de programación C es un lenguaje de programación creado en 1972 por Dennis M. Ritchie. Uno de los objetivos de diseño del lenguaje C es que sólo se necesitan pocas instrucciones en lenguaje máquina para traducir cada elemento del lenguaje, sin que sea necesario utilizar un soporte intenso en tiempo de ejecución.

El juego escogido por el grupo Pokerface corresponde al clásico Ahorcado, en donde el jugador tiene como reto adivinar la palabra que el sistema le pide, y esto se hace según lo que el jugador sugiere por letras o dentro de un cierto número de oportunidades. Se trabajó en replit y en github para la elaboración del mismo; también se utilizó la máquina virtual para asegurarse el funcionamiento correcto del programa. El miércoles de cada semana el equipo se reunía para avanzar juntos en cada parte, además de los avances que se realizaban en el transcurso de la semana.

Con este proyecto se busca aprender a trabajar en equipo, comunicarse mejor, mejorar nuestras habilidades blandas y estudiar cómo funciona el lenguaje C. El lenguaje en C es un lenguaje de programación general de medio nivel que nos permite, entre otras cosas, acceder a la memoria del equipo, gestión de archivos, uso de bibliotecas, uso de arreglos, funciones, estructuras, etc. Conocer este lenguaje nos permitirá en un futuro adentrarnos en temas más avanzados y resolver problemas en relación a la carrera.

El informe técnico presente presenta cómo se pensó el código del programa(cada sección del juego), qué retos y problemas (principales retos) hubo y qué podemos aprender sobre la realización del juego (conclusiones). Para esto se añadirán imágenes del código y de la consola utilizada. En general el objetivo del informe es explicar nuestro programa para que las personas externas a la creación del juego, puedan comprender de cómo funciona y cómo se realizó.

Diseño General

Primero incluímos las librerías que necesitamos en nuestro código, las cuales serían:

```
//Librerías
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
```

La primera corresponde a la librería estándar de C. La segunda sirve para el manejo de la memoria dinámica y procesos; contiene la función (“system”) y (“rand”). La tercera es para la manipulación de arrays. La cuarta y última librería la usaremos para conocer la fecha y la hora del equipo y la usaremos para generar números pseudoaleatorios.
True y false nos servirá para los menús y sus opciones, ya que estos se mantienen en pantalla gracias a ciclos. Utilizamos módulos Void ya que estos se utilizan cuando son llamados y no devuelve un valor, por lo que solo se usarán cuando el usuario lo requiera. También hicimos uso del ciclo “do while” el cual es un bloque de instrucciones que se ejecutarán hasta que la condición cambie, es decir, cada vez se realiza el código y luego se evalúa si lo vuelve hacer o si termina el ciclo.

A continuación se definen las funciones que utilizamos y su utilidad;

```
10  //Se define para ciclos de los menu
11  #define TRUE 1
12  #define FALSE 0
13
14  // Funciones principales del menu
15  void menuPrincipal();  //Menu principal del juego
16  void menuNuevojuego(); //Entramos a jugar
17  void menuRankings();   //Ver la tabla de records
18  void menuReglas();     //Ver las reglas

20  // Funciones del submenú del nuevo juego
21  void juego(char palabras[][15],char nombre[]); //Codigo y ejecucion del juego
22  void categorias(); //Categoria de la plabra
23  void categoriaingresada(int opcion); //Lista de palabras y categoría seleccionada
24  void fallos(int intentos); //Dibujo e intentos del jugador

26  // Funciones del submenú de rankigs
27  void records(); //Buscar los records registrados

29  //Funciones del submenú de reglas
30  void reglas(); //Imprime las reglas en pantalla
```

Lo primero del código será imprimir el menú principal, lo cual lo haremos en el main y esperando que el jugador ingrese una opción. El menú principal está compuesto por las opciones de “nuevo juego”, “reglas”, “rankings” y “salir”.

```
32  //Funcion main y llama al menu principal
33  int main()
34  {
35      menuPrincipal();
36      return 0;
```

```
41  void menuPrincipal() //imprime el menu principal y recibe una opcion del jugador
42  {
43      int opcionPrin;
44      char repetir = TRUE;
45
46      do
47      {
48          system("clear");
49          printf("\n\n\t\t\tAhorcado\n");
50          printf("\t\t\t********\n");
51          printf("\n\t1. Nueva partida\n");
52          printf("\t2. Reglas\n");
53          printf("\t3. Rankings\n");
54          printf("\t0. SALIR\n");
55
56          printf("\n\tIngrese una opcion: ");
57          scanf("%d", &opcionPrin);
```

Luego con la función scanf, switch y cases; leemos la opción del cliente. Scanf permite leer varios tipos de datos pero en este caso solo esperamos que se ingrese un entero. La estructura de Switch nos permite, en un solo bloque, ejecutar diferentes acciones con respecto a la variable. Si la variable es igual a uno se hará el “case 1”, si es igual a dos se hará el “case 2” y así sucesivamente. Funcionan muy parecido a los ciclos while/if pero decidimos que para trabajar se nos hacía más fácil visualizarlo con switch y cases. Dependiendo del caso, se llama a otra función y se utiliza “break” para terminar este ciclo.

```
59          switch (opcionPrin)
60          {
61          case 1:
62              menuNuevojuego();
63              break;
64
65          case 2:
66              menuReglas();
67              break;
68
69          case 3:
70              menuRankings();
71              break;
72
73          case 0:
74              repetir = FALSE;
75              break;
76          }
77      } while (repetir);
78  }
```

Si la opción ingresada fue rankings, se mostrará la puntuación y nombre de los jugadores. Para esto utilizamos la función “fopen,fread y fclose”. Fopen nos permite abrirlo como un flujo y poder manipularlo, en este caso lo abrimos con constante “r” ya que solo lo vamos a leer (“read”). Si el archivo de texto por alguna razón no estuviera en el equipo, saltará un error. (“perror”). Para leer el archivo de texto utilizamos “feof”, lo que hace es leer todo el archivo hasta el final, cada vez que hay un carácter lo guarda en una variable con “fgetc”. “Fgetc” lo que hace es guardar ese carácter donde se encuentre el cursor(flujo) en ese momento; es decir irá en un ciclo guardando e imprimiendo hasta llegar al final del archivo. Por último, cerramos el archivo para evitar problemas con “fclose”.

```
116     int main ();
117     int opcion;
118     char repetir = TRUE;
119     do{
120      printf("\n\n\t\t\tRankings\n");
121      printf("\t\t\t--------\n");
122      FILE * flujo=fopen("record.txt","r"); //abrimos el archivo de txt en modo read
123         if (flujo==NULL)
124           perror("Error al abrir el archivo");
125         char caracter;
126         while(feof(flujo)==0){
127           caracter = fgetc(flujo);
128           printf("%c",caracter);
129         }
130         fclose(flujo);
131      printf("\n\nIngrese un número para regresar al menú: ");
132      scanf("%d", &opcion);
```

```
114   void menuRankings() //funcion de los records
115   {
116     int main ();
117     int opcion;
118     char repetir = TRUE;
119     do{
120      printf("\n\n\t\t\tRankings\n");
121      printf("\t\t\t--------\n");
122      FILE * flujo=fopen("record.txt","r"); //abrimos el archivo de txt en modo read
123         if (flujo==NULL)
124           perror("Error al abrir el archivo");
125         char caracter;
126         while(feof(flujo)==0){
127           caracter = fgetc(flujo);
128           printf("%c",caracter);
129         }
130         fclose(flujo);
131      printf("\n\nIngrese un número para regresar al menú: ");
132      scanf("%d", &opcion);
```

Si la opción ingresada fue reglas, se imprimirán en pantalla las reglas del juego; las cuales son:

Reglas

- La opción de “Nueva Partida” permite al jugador ingresar un nombre, elegir una categoría de palabras e iniciar el juego de ahorcado.
- La opción de “Reglas” desglosa las instrucciones de juego.
- La opción de “Ranking” permite observar las posiciones actuales de las mejores puntuaciones y su usuario correspondiente.
- Después de cada juego, la cantidad de puntos que obtuvo el jugador al final se guarda automáticamente en el sistema.
- El juego se pierde si el jugador no logra encontrar la palabra en menos de 7 intentos. En caso de perder, saldrá la frase “Ahorcado” y juntos con dos opciones: “Volver al Menú” (pantalla inicial) o “Salir” (el cual permite finalizar el juego).
- Si el jugador gana, saldrá la frase “FELICIDADES, LOGRÓ COMPLETAR LA PALABRA”, y se guardará la puntuación.

Esto se hace con un simple printf y espera que el usuario ingrese una tecla para salir. Cabe aclarar en este punto que la función (“system” clear), borra lo que haya en la terminal para evitar que se acumule cosas en la pantalla y sea más cómodo de leer.

```
89  void menuReglas() //reglas sobre el funcionamiento del juego
90  {
91    int opcion;
92    char repetir = TRUE;
93    do
94    {
95     system("clear");
96     printf("\n\n\t\t\tReglas\n");
97     printf("\t\t\t------\n");
98     printf("\tLas reglas son las siguientes: \n1. La opción de “Nueva Partida” permite al jugador ingresar un nombre, elegir una categoría de palabras e iniciar el jue
99
100    printf("\n\nIngrese un número para regresar al menú: ");
101    scanf("%d", &opcion);
```

La última opción será la de “nuevo juego”, la cual es el alma del programa ya que aquí se encuentra el juego en sí. Primero se imprimen las categorías y el jugador selecciona la que desea iniciar. La palabra que tendrá que descubrir será de la categoría seleccionada y habrá 5 posibles opciones para cada una.
Las posibles categorías y palabras son:

- Países (Usa, Francia , Japón, Italia, México)
- Frutas (Manzana, Melocotón, Uva, Naranja, Pera)

- Animales (Tigre, Conejo, Caballo, Rinoceronte, Cerdo)
- Deportes (Fútbol, Natación, Baloncesto, Boxeo, Tenis)

```
147  void categorias() //menu de categorias para que el usuario elija y escriba su nombr
148  {
149
150    printf("\n\n\t\t\tNUEVO JUEGO\n");
151    printf("\t\t\t----------\n");
152
153    int op;
154    system("clear");
155    printf("Categorías \n\n");
156    printf(" 1. Países\n");
157    printf(" 2. Frutas\n");
158    printf(" 3. Animales\n");
159    printf(" 4. Deportes\n");
160    printf(" 5. Salir\n");
161    printf("Selecione una categoría: \n");
162    scanf("%i",&op);
163
164
165    if (op==1) categoriaingresada(op);
166    if (op==2) categoriaingresada(op);
167    if (op==3) categoriaingresada(op);
168    if (op==4) categoriaingresada(op); // selecionar categoría
169    if (op==5) menuPrincipal();
170    else
171       categorias();
```

La opción ingresada es mandada al la función “categoríaingresada(op)” para luego guardar esa cadena de caracteres y mandarla a la función “juego”.

```
176  void categoriaingresada(int op){ //Lista de palabras de cada categoría
177    char nombredecategoria[4][15]={"Paises","Frutas","Animales","Deportes"};
178    char Paises [5][15]={"usa","francia","japon","italia","mexico"};
179    char Frutas [5][15]={"manzana","melocoton","uva","naranja","pera"};
180    char Animales [5][15]={"tigre","conejo","caballo","rinoceronte","cerdo"};
181    char Deportes [5][15]={"futbol","natacion","baloncesto","boxeo","tenis"};
182
183    switch(op){
184      case 1:
185        juego(Paises,nombredecategoria[op-1]);
186        break;
187      case 2:
188        juego(Frutas,nombredecategoria[op-1]);
189        break;
190      case 3:
191        juego(Animales,nombredecategoria[op-1]);
192        break;
193      case 4:
194        juego(Deportes,nombredecategoria[op-1]);
195        break;
```

En la función “juego”. Recibirá las palabras de la categoría seleccionada y le preguntará al usuario su nombre que desea usar. Luego se definen las variables que se utilizaran más adelantes (intentos, puntos, longitud, espacios, aciertos).

```
199  void juego(char palabras[][15],char nombre[]) { //recibe la opcion y el nombre de la categoria para esocger de manera aleat
200    char nombrejugador[100];
201    printf("\n\n Digite su nombre: \n\n");
202    scanf("%s",nombrejugador);
203    int opcion,i,j,k,longitud,espacios,puntos=600;//variables que se usaran
204    int otravez;
205    char letra;
206    int aciertos = 0;
207    int intentos = 0;
208    int ganar = 0;
```

Luego se selecciona una palabra de manera pseudoaleatoria y se almacena la longitud de la palabra, esto para que se coloque los guiones bajos en relación a esta por medio de un for.

```
209     srand(time(NULL));
210
211     opcion = rand() % 5; //Numero aleatorio entre 0 y 4 para escojer la palabra
212     longitud = strlen(palabras[opcion]); //se almacena la longitud de la palarba
213     char frase[longitud];
214
215     for(i=0; i<longitud;i++){ //se colocan la cantidad de guiones bajos
216       frase[i]= '_';
217     }
```

Luego, se imprimirá en pantalla datos que el jugador deberá tener presente. Los cuales son la categoría, los intentos, la puntuación y se imprimen los guiones bajos.

```
219     do{ //ciclo del juego
220       aciertos = 0;
221       system("clear");
222       printf("\n\t\t\t\tEL AHORCADO\n\n");
223       printf(" CATEGORIA: %s\n\n",nombre);
224       printf(" Intentos: %i\t\t\t\tPuntuacion: %i\n\n",7-intentos,puntos);
225       fallos(intentos);
226
227       printf("\n\n\n"); //imprime los guiones bajos
228       for (i=0;i<longitud;i++){
229         printf("%c ",frase[i]);
230       }
```

El juego en este momento, verificará si el jugador ya ganó o si por el contrario perdió. El jugador pierde si se queda sin intentos (inicialmente tiene 7). El programa detecta que el jugador ganó la partida si ya no quedan guiones bajos; para esto contará los guiones por medio de un ciclo. En ambos casos saldrá en la pantalla la puntuación final y se guardará la puntuación en el archivo de texto por medio de flujos. Para esto se abrirá de manera “append”, en este modo permite guardar datos, sin borrar los anteriores o crear el archivo en caso de que no exista. Se realiza de manera parecida a cuando se lee el archivo solo que aquí usaremos “fputs” la cual guarda una cadena de texto o un espacio. El “fflsuh” sirve para eliminar buffer (memoria) en el archivo y evitar algunos posibles errores ya que se utiliza mucho “scanf” en el código. Se le pedirá al jugador si quiere volver al menú principal o volver a jugar.

```
233     if (intentos==7){ //verifica si todavia tiene intentos
234       printf("\n\n PERDISTE!! :C ");
235       printf("La solucion era; %s \n\n",palabras[opcion]);
236       printf("\n\n Puntuacion final: %d \n\n",puntos);
237        FILE * flujo = fopen("record.txt","a"); //abre el archivo y guarda el puntuaje
238       if (flujo==NULL){
239         perror("Error al abrir el archivo\n\n");
240       } else{
241         fputs(" ",flujo);
242         fputs(" ",flujo);
243         fputs(nombrejugador,flujo);
244         fputs(" ",flujo);
245         fputs(" ",flujo);
246         fputs(",",flujo);
247         fputs(" ",flujo);
248         fputs(" ",flujo);
249         fprintf(flujo,"%d",puntos);
250         fputs("\n",flujo);
251       }
252       fflush(flujo);
253       fclose(flujo);
```

```
278           if (espacios == 0){
279                         printf("\n\n FELICIDADES GANASTE!! :) \n\n");
280        printf("\n\n Puntuacion final: %d \n\n",puntos);
281        FILE * flujo = fopen("record.txt","a"); //abre el arhcivo de txt y guarda el puntuaje
282       if (flujo==NULL){
283         perror("Error al abrir el archivo\n\n");
284       } else{
```

Ahora el programa, si el jugador no ha ni ganado ni perdido, le pide que ingrese una letra para verificar si se encuentra en la palabra, la palabra ingresada se guarda en la variable “letra”. Se hace por medio de ciclos en función a la longitud de la palabra, si se encuentra la letra en la palabra se suma un acierto y vuelve a empezar el ciclo (desde la línea 219, la cual era un “do”) si no, le resta 100 puntos al jugador y suma un intento. La variable intentos se define como (“7-intentos”) por lo que cuando se vuelva el ciclo se tendrá cada vez menos intentos.

```
314      //verificar si la letra ingresada se encuentra en la palabra
315      for (j=0;j<longitud;j++){
316         if (j==longitud && aciertos==0){
317           intentos ++;
318           puntos -= 100;
319         }
320         if (letra == palabras[opcion][j]){
321           frase[j] = letra;
322           aciertos ++;
323         }
324      }
325
326      if (aciertos==0){
327         intentos ++;
328         puntos -= 100;
329      }
330
331     }while(intentos != 8);
332     printf("\n\n");
```

Por último tendríamos el dibujo del ahorcado, la cual irá cambiando en función de los intentos, se utilizan funciones similares(voids, switch y cases) a las que hemos ido utilizando en todo el código. Si “intentos” aumentan, el “case” cambia al siguiente.

```
339   void fallos(int intentos) {
340           switch (intentos) {
341           case 0:
342                   puts("Comienza el juego, exitos! ^_^");
343                   puts("  _ _ ");
344                   puts(" |   | ");
345                   puts(" |    ");
346                   puts(" |   ");
347                   puts(" |    ");
348                   puts("_|_  ");
349                   break;
350           case 1:
351                   puts(" Oh no letra incorrecta! ");
352                   puts("  _ _ ");
353                   puts(" |   | ");
354                   puts(" |    ");
355                   puts(" |   ");
356                   puts(" |    ");
357                   puts("_|_  ");
358                   break;
```

Principales retos

1. El primer reto que nos encontramos fue que no teníamos mucha experiencia práctica en el uso del lenguaje por lo que tuvimos que repasar y buscar mucha información en relación al lenguaje C. Muchas de las cosas no nos acordabamos bien o no sabíamos exactamente cómo era el formato. También hubo algunos problemas de conectar github con la máquina virtual pero se soluciona rápido. Creemos que nuestro programa se podría mejorar y optimizar de una mejor forma.
2. Un problema más específico fue en el uso de la función scanf, utilizada para que el usuario ingrese la letra, recibir el carácter y luego comprobar si la letra se encuentra en la palabra. Era necesario dejar ese espacio entre las comillas, es decir “-espacio-%c” ya que si no se dejaba ese espacio el programa recibía el carácter con el “enter” del usuario. Es decir el programa comprueba que si la letra está en la palabra y luego comprueba si el “enter” está también. Esto era un gran problema ya que hacía que el usuario perdiera dos intentos de una vez o incluso si la letra era correcta igual restaba un intento. Se tuvo que buscar en stack overflow. (https://stackoverflow.com/questions/8464620/program-doesnt-wait-for-user-input18-with-scanfc-yn)

```
scanf(" %c",&letra);
```

3. Hubo problemas con la parte de guardar las puntuaciones ya que no lo logramos del todo. Nuestro programa lo que hace es guardar el nombre y la puntuación de cada jugador al terminar cada partida pero en el archivo de txt y al imprimir las puntuaciones el programa no las acomoda de mejor a peor. Es decir no se logró acomodar las puntuaciones. Una de las causas fue que esta semana teníamos algunas pruebas por lo que el tiempo se redujo. La idea inicial era separar los nombres y las puntuaciones en dos arreglos; luego buscar el mayor de ese

arreglo e imprimirlo junto al nombre (ya que guardan la misma posición del arreglo) y borrarlo del arreglo para que se vuelve hacer en un ciclo hasta imprimir los 10 mejores.

4. El juego tiene algunos errores por ejemplo cuando se le pide al usuario un número para escoger una opción pero se ingresa una letra; el programa a veces entra en un ciclo infinito raro y falla. Creemos que si se ingresa una letra debería seguir saliendo el menú de categorías pero entra en un ciclo que no permite ingresar otra opción ya que se imprime el menú de manera infinita.

5. Una de las ideas iniciales consistía en agregar una opción de ayuda donde el jugador durante la partida tuviera la posibilidad de “pedir ayuda” en caso de no saber cuál palabra ingresar pero tampoco querer perder. Esta ayuda iba a consistir de ingresar cierto comando y así recibir a cambio de una penalización, una frase la cual le diera una idea más clara al jugador sobre qué palabra podría hacer referencia el juego. Esta ayuda no fue implementada al final debido a que ingresar un comando en media partida era difícil de implementar, ya que durante el juego cualquier tecla que el jugador iba a contar como parte del juego.

Conclusiones

El trabajo realizado fue una experiencia gratificante, ya que aunque no logremos el trabajo cien por ciento perfecto; nos sirve para aprender que tenemos que mejorar, visualizar nuestras habilidades, comportamiento para nuestro futuro laboral. El presente trabajo también necesitaba una parte de investigación ya que ocupamos información extra para lograr el funcionamiento del programa. Nuestro grupo, en lo personal, la pasamos muy bien construyendo el código y se logró una buena química, ya que siempre hubo una gran comunicación y cada quien cumplía con su trabajo.

Programar en un lenguaje en el cual no teníamos experiencia previa supuso un gran reto a enfrentar pero consideramos que hubo un gran avance en el aprendizaje en esta área.

Aprender el funcionamiento básico del lenguaje de programación en C es un gran apoyo a la persona para mejorar sus habilidades en el manejo de la computadora y las tecnologías en general. En la actualidad, con el uso del celular y computadoras, las personas usan diariamente diversas aplicaciones como ayuda en las actividades diarias, tales como aplicaciones de mensajería o sistemas de navegación; también como entretenimiento y aprendizaje, como es el caso de videojuegos o servicios de video. Todas estas aplicaciones están hechas con base en diversos lenguajes de programación, y tener una mejor comprensión de estos lenguajes nos expande el conocimiento de cómo funcionan todas aplicaciones que utilizamos día a día.

Referencias

1. Historia del Lenguaje C. (2019, junio 24). *EcuRed,* . Consultado el 17:07, noviembre 30, 2021 en https://www.ecured.cu/index.php?title=Historia_del_Lenguaje_C&oldid=3425421

2. Lenguaje C. *Enrique Vicente Bonet Esteban.* https://informatica.uv.es/estguia/ATD/apuntes/laboratorio/Lenguaje-C.pdf

3. Bautista, C. (2020, 30 junio). *El Ahorcado en Lenguaje C* [Vídeo]. YouTube. https://www.youtube.com/watch?v=yJWTtwhLHj4&t=988s

4. Programador EC. (2020, 3 septiembre). *Juego en lenguaje C | El Ahorcado | Descarga Gratis* [Vídeo]. YouTube. https://www.youtube.com/watch?v=kmQvQ2Y0X88&t=52s