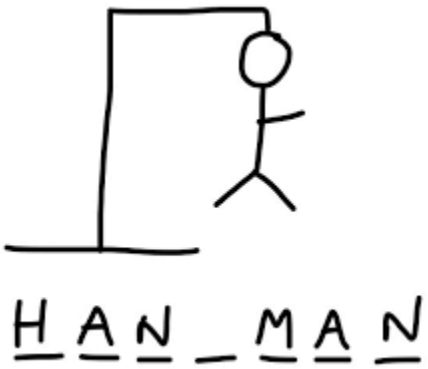
Universidad de Costa Rica

Escuela de Ingeniería Eléctrica

Curso de Plataformas Abiertas IE-1116

Proyecto Final del curso

Juego programado en C: “el ahorcado”



Estudiantes:

Alex Ramírez Binns B76157

María Isabel Varela Moreno A76713

II Semestre, 2021

Índice

Introducción

Mediante el presente documento se explica la creación y el desarrollo del popular juego del ahorcado, donde el jugador debe atinar la palabra propuesta por el juego. El juego del ahorcado es un juego de adivinanzas de dos o más jugadores, en el cual, un jugador piensa una palabra, frase, oración y el otro trata de adivinarla basado en el orden de las letras o basado en un cierto número de oportunidades.

El desafío lanzado para adivinar las palabras está basado en el conocimiento del vocabulario español del jugador y su capacidad de asociación y memorización. Específicamente en este proyecto decidimos trabajar con adivinar frutas, pero incluso se podría hacer un poco más complejo ampliándolo a más categorías como países, frutas, animales o dejarlo abierto al participante sin explicarle a qué tipo de categoría pertenece la palabra.

Desarrollo

Estructura del Código

El juego implementado está diseñado en tres porciones:

1. Menú principal– en el cual el jugador puede empezar a jugar, consultar las máximas puntuaciones almacenadas o declinar su intención de juego, dependiendo de cuál número escriba en el teclado cada opción será habilitada.
2. Juego de ahorcado– en esta sección el jugador empezará a jugar y recibirá una palabra para adivinar. Una función selecciona una palabra de una lista de palabras de la categoría de frutas, y de acuerdo con la palabra seleccionada el puntero dibujará la cantidad de guiones para que el jugador sepa la longitud de la palabra. El jugador tiene 6 intentos para tratar de adivinar la palabra propuesta. Con cada intento fallido al ingresar una letra que no se encuentre en la palabra se irán restando los intentos y además se irá dibujando las partes las extremidades de un hombre que colgará de un poste, hasta llegar a perder en el juego, donde el dibujo

del hombre colgando se dibujará por completo. Caso contrario, si la persona adivina la palabra obtiene la máxima calificación de puntos.

3. Máximas puntuaciones o ranking– esta sección recibe las 10 máximas puntuaciones guardadas de la sección anterior, con el fin de mostrar un arreglo de las top 10 calificaciones. Esta sección se implementó usando una lista dinámica (tipo pila, donde acomoda el último en entrar es el primero en salir LIFO), o varía la cantidad almacenada de datos conforme a ésta varíe.

El código de este proyecto está desarrollado en el lenguaje de programación C, cuyo lenguaje es uno de los más básicos y trascendentales comparado con otros lenguajes existentes. Con él se puede programar comandos, aplicaciones y herramientas que nos ayudan en la solución de ciertos problemas computacionales o cotidianos, sin darnos cuenta de que interactuamos con un gran número de dispositivos embebidos programados en C. Su versatilidad radica en que se puede operar y ejecutar en sistemas operativos como Windows, Mac, Linux y Unix. Como principales ventajas para programar en C están la eficacia y capacidad de manejar instrucciones de CPU, usa un lenguaje de máquina, por tanto, las instrucciones están optimizadas, no depende del hardware y el código se puede transferir a otros sistemas operativos.

Funciones, comandos y bibliotecas empleadas en el proyecto

Adicional a la estructura del código se emplean bibliotecas, funciones o comandos en el lenguaje de C como las siguientes:

#include (bibliotecas): todo programa de este lenguaje debe comenzar con esta función #include que permite inicializar el entorno de trabajo, ficheros con extensión.h. Por ejemplo, se agregan las bibliotecas stdio.h o stdlib.h que contienen órdenes implícitas como el *printf* para imprimir en consola. Entre las bibliotecas más comunes se encuentran la *stdio.h* para entrada y salida, *math.h* para operaciones matemáticas y *time.h* para le fecha y hora actual. Se deben especificar todos los ficheros de cabecera () correspondientes a las librerías de funciones utilizadas.

Son librerías implementadas y listas para que nosotros las usemos con sólo llamar a la función que tenga implementada dentro dicha librería. Por ejemplo:

```
#include <stdio.h>
#include <conio.h>
```

Declaración de argumentos o parámetros: es información que se pasa de otras secciones del programa. Cada final de la sentencia debe terminar con un punto y coma (;). Se realiza una declaración de los prototipos de las funciones implementados en este módulo que le da información al compilador de una función que va a ser utilizada pero que todavía no ha sido implementada. En particular le dice al compilador qué tipo de datos requiere y cuál devuelve la función. Puede que la declaración se dé para números enteros, números flotantes, caracteres, un arreglo, una cadena (*string*). Por ejemplo:

```
int resta (int a, int b);
float puntaje;
char nombre [30];
```

Int main-Función principal: el código cuenta con una función principal denominada main, que puede constar de una o más funciones. Cada función debe contener un header, un argumento y lo que se denominan sentencias. Int main permite que el programa vuelva al mismo punto de retorno luego de ejecutarse y necesita llevar corchetes { } al inicio y al final. Se define como el punto de partida donde todos los programas inician su ejecución sin importar donde se ubiquen dentro del código fuente; lo que se contiene dentro de las llaves que delimitan la función es lo que hace cuando se ejecuta. El return 0 hace que le código principal termine. Por ejemplo:

```
main() {
……
……
```

```
return 0;

}
```

Void: Funciona de la misma forma que la función main, sin embargo, no debe retornar ningún valor de vuelta. En contraste con el main el cual sí debe retornar un valor. Una variable de este tipo almacena un puntero a cualquier dato. Por ejemplo:

```
void lista_palabras();
```

Typedef: Me sirve para crearme un tipo distinto de los preestablecidos de partida, y usarlo de manera particular. **Por ejemplo:**

```
typedef struct ranking {

int a;

int b;

 } puntaje;
```

Punteros en el código: Por ejemplo, *printf* para imprimir en consola, *scanf* para recibir y almacenar el texto o número que se escriba en consola, *exit* para salir de la consola o finalizar la ejecución de un proceso o terminar el programa de manera inmediata. Los punteros son variables que contienen la dirección de memoria de un dato o de otra variable que contiene al dato en un arreglo y permiten manipular la memoria del ordenador de forma eficiente. El operador "&" seguido del nombre de una variable devuelve su dirección de memoria. El tipo de datos del resultado es "puntero a" seguido del tipo de la variable utilizada.

Por ejemplo:

Puntero a int a

Puntero a la dirección de a int *a

Puntero que devuelve el contenido de a int &a

Estructuras-struct: son tipos de datos construidos utilizando objetos de otros tipos. Todo lo que esté dentro de las llaves pertenece a una estructura. Pueden contener variables de muchos tipos diferentes de datos. Las estructuras pueden ser inicializadas mediante listas de inicialización como con los arreglos. Para inicializar una estructura escriba en la declaración de la variable a continuación del nombre de la variable un signo igual con los inicializadores entre llaves y separados por coma (ver ejemplo 1). Las estructuras pueden ser pasadas a funciones pasando miembros de estructura individuales o pasando toda la estructura. Cuando se pasan estructuras o miembros individuales de estructura a una función se pasan por llamada por valor. Para pasar una estructura en llamada por referencia tenemos que colocar "*" o "&" (ver ejemplo 2).

Ejemplo 1:

```
struct ejemplo {
 char c;
int i;
};
```

Ejemplo 2:

```
punto creo punto(int a, int b)
{
 punto temp;
temp.x=a;
temp.y=b;
return temp;
}
```

Switch: en el caso de que se deban tomar varias decisiones después de realizar una elección múltiple o varias alternativas se puede utilizar este comando. Se utiliza default en caso de que alguna de las opciones del switch no coincida con el resultado al evaluar la expresión. La palabra break hace que el flujo del programa se detenga por un momento para evaluar la opción o ejecutar el programa. La sintaxis para su uso es la siguiente:

```
switch(var int o char)
{case opcion 1:
instrucciones; break;
case opcion 2:
instrucciones;
break;
default: instrucciones;
};
```

Condicionales- if, else: La sentencia *if* elige entre varias alternativas en base al valor de una o más expresiones booleanas. Por ejemplo:

```
if( a>0, a--) <bloque a ejecutar cuando la expresión es verdadera>
        else <bloque a ejecutar cuando la expresión anterior es falsa>
```

La sentencia *else* es opcional, puede utilizarse o no. En el caso de no utilizarlo, cuando la expresión evaluada sea falsa la ejecución seguirá con la sentencia inmediatamente posterior al *if*.

For: la sentencia *for* se utiliza para realizar ciclos, donde usualmente una variable se debe iterar varias veces. Por ejemplo, la sintaxis correcta para usarse sería:

```
for (inicialización; condición; incremento)
{instrucción(es);
};
```

Comentarios: pueden aparecer en cualquier parte del programa y sirve para entender o explicar en qué consisten ciertas secciones de un programa, o el propósito de lo que se desea alcanzar. Usualmente se pueden usar dos barras inclinadas //, y permite escribir el comentario sin que se ejecute nada. Se puede usar al final de un comando o como un encabezado o título que explique parte del código siguiente. Por ejemplo:

```
contador = contador + 1; // esta parte del código le suma una unidad al contador
```

Estructuras dinámicas de datos: se refiere a un tipo de estructuras de datos que agrupan los datos con los que trabajará nuestro programa y además uno o más punteros "autoreferenciales", es decir, punteros apuntan a objetos del mismo tipo. Existen diferentes tipos: listas dinámicas, pilas dinámicas, colas, árboles y grafos. En una lista dinámica lineal cada elemento apunta al siguiente, es decir, cada elemento tiene información de dónde está el siguiente. Las pilas utilizan el razonamiento de LIFO, el último en ingresar es el primero en salir. Existen las colas, usando la dinámico FIFO, el primero en entrar es el primero en salir,por tanto los elementos se almacenan en fila, pero sólo pueden añadirse por un extremo y leerse por el otro. Por medio de punteros también se puede liberar memoria de manera dinámica.

Función Malloc (memory allocation) y free: Esta función nos permite asignar memoria sirve para solicitar un bloque de memoria del tamaño suministrado como parámetro. Devuelve un puntero a la zona de memoria solicitada en la función. Cuando la reserva de memoria malloc ya no se ocupa y requiere ser liberada se usa el comando free. Por ejemplo:

```
puntero= (int*)malloc ( 10*sizeof(int) ); // reservando un espacio de 10 enteros
…
free(puntero);
```

Retos de la programación

Para mencionar algunos de los principales retos para este juego, encontramos la siguiente información:

- Poco conocimiento del entorno de Github para utilizarlo como una herramienta de control de versiones: En este caso nos dimos a la tarea de investigar cómo utilizar la herramienta de push, pull, commit en Github para unificar el código que trabajamos por secciones e ir realizando cambios en conjunto. Es muy útil su aplicación en entornos colaborativos donde se comparte un proyecto con varias personas a la vez y todos debe aportar su sección del código. Como no lo habíamos utilizado previamente nos dimos a la tarea de investigar qué comandos se requerían para crear el repositorio y compartirlo con nuestro equipo.
- Aún nos es un poco difícil comprender cómo utilizar las estructuras en el código C, pues nunca había sido empleado en otros casos similares. Por tanto, la mayor parte del código emplea funciones para ir completando el juego, pero no dejamos de lado el uso de las estructuras para realizar ciertas tareas específicas y simplificar el código.
- Desconocíamos de qué forma hacer el dibujo del ahorcado por lo que debimos hacer una búsqueda bibliográfica para aprender a imprimir en pantalla las partes del cuerpo del dibujo de la persona según cada caso de intentos fallidos.
- Uno de los principales retos que tuvimos durante el semestre y que aplicaba al curso fue llevarlo de manera virtual y de manera asincrónica fuera del horario de la universidad. Quizá existan otros cursos en la Universidad que posean el mismo inconveniente para aquellos estudiantes que debemos trabajar y estudiar al mismo tiempo.
- Además de los retos mencionados anteriormente, pudimos encontrar el reto de aprender más sobre la programación en lenguaje C, donde nos dimos a la tarea de investigar sobre el tema y aprender a usar condicionales, librerías, estructuras, funciones y desarrollar el pensamiento lógico para programar, pues debimos reconocer el problema a desarrollar e implementar manera de cómo solucionarlo. El conocimiento y aprendizaje básico de la programación del lenguaje C es una

tarea diaria y continua que requiere tiempo, autoaprendizaje y búsqueda bibliográfica. No es un lenguaje tan sencillo, sin embargo, con la constancia y orden adecuado se puede ir dominando paulatinamente.

Conclusiones

En todo momento disfrutamos el trabajo de manera colaborativa, pues además del uso de herramientas, el curso nos enseñó a crear redes de apoyo para resolución de problemas de programación. Pudimos buscar información, conocer un poco más al respecto y también escuchar el consejo u opinión de otros colegas ingenieros para el desarrollo del proyecto.

El proyecto nos enseñó a conocer mucho más acerca de programación, a conocer nuevos entornos como el de C, el cual desconocíamos por completo, y nos ayudó a poner a prueba nuestra capacidad de lógica para desarrollar el pensamiento creativo. El hecho de aprender mediante el juego es catalogado como una experiencia atractiva al ser humano donde la parte lúdica permite desarrollar el pensamiento creativo.

El aprendizaje de los lenguajes de programación en código abierto es muy importante para nuestra carrera, sin embargo, son pocos los cursos que se ofrecen en la escuela con un método de aprendizaje que sea muy interactivo, o se desarrolle la parte creativa con mayor énfasis que en otras carreras. Agradecemos este proyecto propuesto pues nos enfrentó al mundo de la programación de plataformas abiertas que se emplean en el diario vivir en las herramientas que usamos en la vida cotidiana y que pasan desapercibidas.

Referencias bibliográficas

1. Estructuras de registros en C: http://cryptomex.org/Tutorial-LengC/estructura.html
2. Estructuras de registros en C: http://programandoenc.over-blog.es/article-struct-o-estructuras-de-registros-en-c-123607830.html
3. Registros-struct: https://www.aprendeaprogramar.com/cursos/verApartado.php?id=16007

4. Configuración de Github: https://docs.github.com/es/account-and-profile/setting-up-and-managing-your-github-user-account/managing-access-to-your-personal-repositories/inviting-collaborators-to-apersonal-repository
5. Juego en Lenguaje en C: https://m.youtube.com/watch?v=kmQvQ2Y0X88&feature=youtu.be
6. Manual de programación de Lenguaje en C: chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/viewer.html?pdfurl=https%3A%2F%2Fpaginas.matem.unam.mx%2Fpderbf%2Fimages%2Fmprogintc%2B%2B.pdf&clen=150007&chunk=true
7. Estructuras dinámicas en C: chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/viewer.html?pdfurl=https%3A%2F%2Fkesquivel.files.wordpress.com%2F2014%2F08%2Festructuras-dinamicas.pdf&clen=301935&chunk=true