Introducción a la algorítmica y programación

<Departamento de Computación>

<Facultad de Ciencias Exactas, Físico-Químicas y Naturales>

<Universidad Nacional de Río Cuarto>

Año 2013

PROYECTO:

<Juego PasaPalabra>

AUTORES:

Elena, Pablo - DNI: 38.339.088

Gremiger, Santiago - DNI: 40.504.384

Martinez, Christian - DNI:35.544.318

Enunciado:

El juego “Pasapalabra” consiste de 26 letras ordenadas (de la A a la Z, sin la Ñ), donde cada letra se corresponde a una palabra que contiene dicha letra. Al jugador (usuario) se le muestra cada palabra (una por vez). Hay 2 niveles, el fácil y el díficil. En el nivel fácil se tapan dos letras de cada palabra, y en el difícil 3 o más (si la palabra tiene menos de 4 letras, se tapa solamente una letra). Pero nunca se tapa la letra que corresponde a la palabra, es decir, si estoy en la letra A y la palabra es marcos, se muestra ma_c_s). El juego comienza por la letra A y se recorre dos veces el abecedario. En el primer recorrido el usuario tiene 2 opciones:

a) intentar acertar la palabra

b) o bien decir pasapalabra. En esta segunda opción el usuario continúa jugando con la siguiente palabra que le corresponde. La opción de pasapalabra puede ser usada cuantas veces como quiera en el juego, pero sólo 3 seguidas. En el segundo recorrido del abecedario (en orden de A a Z nuevamente), el usuario solo puede jugar con las palabras que salteó eligiendo pasapalabra, es decir, si intentó adivinar pero falló, no juega de nuevo con esa palabra.

ANALISIS

Problema:

El problema principal es el juego en su totalidad, que a su vez se divide en 3 sub

problemas

- Generar rosca (elegir palabras)
- Llevar registro de los usuarios y los puntajes
- Ocultar las letras dentro de una palabra
- Creacion del menú principal
- Iniciar partida

***Generar Rosca:**

Se debe crear un arreglo de 26 elementos que contendrán las palabras.

Cada campo del arreglo es un registro con 3 campos:

Letra: Contiene la letra de la palabra.

Palabra: Contiene una lista simplemente encadenada de caracteres que forma la palabra.

Jugar: decide si la palabra juega o no en la segunda vuelta.

Para cargar el arreglo con las 26 palabras, se utiliza el archivo “palabras.dat” donde se utiliza un indice para recorrer el archivo cada 5 registros (porque cada letra contiene 5 palabras) dentro de este índice, hay un sub índice que utiliza un random para devolver una posición aleatoria dentro de esas 5 palabras, y carga la en el arreglo.

El modulo tiene como pre condición que el archivo “palabras.dat” debe contener 130 registros (palabras)

*Registro de usuarios y puntajes:

Se generan 2 archivos para registrar los usuarios, y sus puntajes:

Usuarios.dat: archivo que contiene el registro de todos los usuarios

Puntajes.dat: Archivo que almacena un registro de 3 campos

Usuario: Nombre del usuario.

Puntaje: puntaje obtenido

Dificultad: dificultad en la que jugo

*Ocultar letras de una palabra:

Se utiliza la función random para generar un numero aleatorio entre 1 y la cantidad de letras de la palabra para usarlo como posición dentro de la lista que contiene los caracteres de la palabra, se verifica que la letra en esa posición no este oculta y no sea la letra correspondiente a la palabra, si la verificación es correcta, so procede a ocultar la letra. Si la dificultad es fácil, oculta solo dos letras (si la palabra tiene menos de 4 letras, oculta 1 letra), si la dificultad es difícil, tapa tantas letras como la mitad de la longitud de la palabra. Una vez cumplida esta condición según la dificultad, pasa a la próxima palabra.

*Creación del menú principal:

En el menú principal tenemos varias opciones (dejando de lado el iniciar partida, explicado más adelante):

a. Calcular promedio:

Para calcular el promedio se utilizo una función recursiva en cola.

b. Cambiar de usuario:

Es un modulo que lee una string, e invoca una función que busca esa string (usuario) dentro del archivo “usuario.dat” si lo encuentra retorna “true” si el usuario existe en el archivo, por ende cambiara a ese usuario, en caso de no existir le preguntara al usuario si desea crear uno nuevo o ingresar otro nombre.

c. Crear usuario:

Es un modulo que lee una string y la inserta en el archivo “usuarios.dat”. No se podrá ingresar un usuario en blanco (Si se presiono enter sin ingresar un nombre).

d. Mostrar los 10 mejores puntajes:

Para mostrar los 10 mejores puntajes se cargan todos los registros del archivo “puntajes.dat”, donde se compara el registro a insertar con el primer registro, si es mayor lo inserta antes, si es menor lo inserta después y vuelve a comprar con el siguiente, y asi hasta cumplir la condición.

e. Salir del juego:

Detiene la ejecución del programa.

*Iniciar partida:

A la hora de jugar el usuario tiene 3 opciones:

a. Jugar palabra:

Le solicita al usuario que ingrese una palabra (string) luego compara ésta con la palabra del registro que se encuentra en el arreglo Rosca. En primer lugar verifica las longitudes, si no coinciden ya sabemos que es incorrecto, en caso de coincidir pasa a verificicar que en la misma posición de la lista y de la string los caracteres coincidan, si esto ocurre da la palabra como correcta, caso contrario como incorrecta. Cada vez que se juega una palabra al campo “.jugar” del registro actual del arreglo rosca se le asigna “false” por ende esta palabra no jugara en la 2da vuelta independiemente si haya acertado o no.

b. Pasapalabra:

El usuario puede saltear la palabra actual para jugarla en la 2da vuelta, solo puede hacerlo 3 veces seguidas, luego esta obligado a jugar si o si, cada vez que juega el contador de pasapalabras utilizables vuelve a 3, cada vez que se usa pasalapabra al campo ".jugar" del registro actual del arreglo rosca se le asigna "true"

c. Volver al menú principal:

Se finalizara la partida actual y no se guardaran los datos obtenidos hasta el momento.

Problemas durante el proceso

El primer problema con el que nos encontramos fue la selección de la palabra correspondiente a la letra a jugar. Queríamos que fuese completamente al azar, que no siga un patrón, entonces decidimos hacer uso de un random. Utilizabamos un índice que supuestamente era la letra (1=a, 2=b, 3=c) y a eso le sumábamos un random(5)+1, el problema con esto es que estaba mal planteado, puesto que o se ponían dos palabras de una misma letra o se salteaba alguna que otra.

Solución: dentro de un ciclo, a una variable 'pos' le asignamos random(5) más la letra (variable 'sigLetra' que va de cinco en cinco, por lo tanto las palabras de la primer letra están de 0 a 4, las palabras de la segunda de 5 a 9 y así). Esto lo hace 26 veces, una por cada letra de la rosca.

Otro de los problemas fue durante la creación de la función recursiva "promedio", pues no se puede usar un puntero en un case.

Solución: usar un if (si)

Dentro de la misma función se presentó el contratiempo de que analizaba todos los puntajes menos el último.

Solución: cambiar de *if ((lista^).next=nil) then* a *if (lista=nil) then begin*

Luego queríamos representar la "rosca" como tal, hacer un círculo con las letras del abecedario. Tras varios intentos, como no pudimos hacer un círculo debido a la cantidad de caracteres, intentamos mostrarlo en un cuadrado. Pero esto también trajo problemas pues sobraba una letra (la Z estaba en medio del cuadrado, no podía formar parte del borde).

Solución: Representar la rosca como una linea recta, de la siguiente manera:

[A] [B] [C] [D] [E] [F] [G] [H] [I] [..] [Z]