Palapasabra

Realizaremos una implementación, totalmente operacional, del juego Pasapalabra o Rosco. Conceptualmente mantendremos información de un conjunto de jugadores con la cantidad de partidas que ha ganado cada uno y cada partida consistirá en una contienda entre dos jugadores de los cuales uno ganará. Se puede ver una idea del juego aquí, obviamente nuestra implementación será mucho más simple.

> Nota: Todos los ítems marcados como (**PROMOCION**) significa que los tienen que resolver **sólo** los alumnos que les interesa promocionar, por lo tanto que sacaron 7 o más en el parcial.

El juego tendrá el siguiente menú de opciones:

1. Agregar un jugador.
2. Ver lista de jugadores.
3. Ver los diez jugadores con mayor cantidad de partidas ganadas. (**PROMOCION**)
4. Jugar
5. Borrar un Jugador (**PROMOCION**)
6. Salir

Al comenzar el programa carga desde el archivo jugadores.dat (**que ustedes deben crear y mantener en su propio nombre**) y crea el árbol jugadores que estará ordenado por nombre y posee la cantidad de jugadas ganadas. Luego muestra el menú de arriba.
Al finalizar no guarda ninguna información adicional.

Veremos ahora una explicación de cada una de las opciones de arriba:

1. **Agregar un jugador**.
    a. Pide el nombre del jugador.
    b. Verifica que el nombre no existe en el árbol jugadores, si existe avisa y no permite su agregado.
    c. Si no existe lo agrega en el árbol jugadores y al final del archivo jugadores.dat con cantidad de partidas ganadas en cero.
2. **Ver lista de jugadores**: A partir del recorrido in-order del árbol jugadores muestra todos los existentes con la cantidad de partidas ganadas por cada uno.
3. **Diez Jugadores más ganadores**: A partir del recorrido in-order del árbol guardará en un arreglo de tamaño diez los punteros a los mayores ganadores en orden descendente y luego mostrará sus nombres y cantidades por pantalla.(sólo **PROMOCION**)
4. **Jugar**:
    a. pide el nombre de dos jugadores, deben estar en el árbol y ser distintos, se cargan en el arreglo partida.
    b. carga las preguntas, todas marcadas como [Pendiente] en dos listas circulares apuntadas por el arreglo partida, estas preguntas se obtienen del archivo palabras.dat.
    c. comienza el juego, con el primer jugador:
        i. Se verifica que aún tiene palabras sin contestar, sinó termina la partida.

ii. Avanza hasta la primer palabra sin contestar. Si el puntero apunta a una palabra sin contestar no es necesario avanzar.
iii. Muestra la Letra, la Consigna y se queda esperando el ingreso de la respuesta del jugador.
iv. El jugador responde, puede responder: “pp” significa PasaPalabra, de lo contrario el texto es su respuesta y se compara con la palabra buscada, si es igual significa “Acertada”, de lo contrario es “Errada”.
v. Si el texto es “pp”, quedará la palabra como [Pendiente] y se avanza el puntero a la siguiente palabra, se pasa al otro jugador y se vuelve al punto i.
vi. Si el texto no corresponde a la palabra se marca [Errada], se avanza el puntero a la siguiente palabra, se pasa al otro jugador y se vuelve al punto i. Si no se marca [Acertada], se avanza el puntero a la siguiente palabra y se vuelve al punto i.

d. Cuando se sale del ciclo se terminó la partida, si hay un jugador con más aciertos que otro se suma 1(uno) a la cantidad de jugadas ganadas en el jugador tanto en el árbol como en el archivo.

5. **Borrar un jugador**: (sólo **PROMOCION**) Se pide el nombre del jugador, se verifica que exista en el árbol, se **elimina el nodo** en el árbol y se marca con eliminado=true en el archivo. Si no existe se le avisa al usuario.
6. **Salir**: Sale del programa, no necesita guardar nada porque todo fué guardado en su momento.

Notas:

Dado que el pascal tiene problemas con el readln de un char recomendamos implementar el menú utilizando una variable de tipo string para la lectura del número de la opción.

Archivos utilizados

jugadores.dat

Cada registro tiene el nombre del jugador y un integer con la cantidad de partidas ganadas. Este archivo debe ser creado y actualizado por ustedes únicamente.
Tengan en cuenta que los archivos van a estar todos en el mismo disco y por lo tanto deben diferenciar los suyos con los del resto para no romperlos sin querer, por eso proponemos que todos sus archivos estén prefijados con un mismo texto, por ejemplo apellido y nombre, luego el nombre del archivo, por ejemplo: podría quedar assign (archivo, '/ip2/juanperez-jugadores.dat');

(**PROMOCION** Se agrega un atributo booleano en el registro llamado (eliminado), cuando se borra un jugador se pone eliminado=true. Y cuando se cargan los jugadores al árbol no se toman los eliminados.)

palabras.dat

La cátedra brinda, almacenadas en este archivo, 5 Sets o juegos distintos de palabras para que los jugadores no tengan siempre las mismas preguntas, cuando se cargan las preguntas para un jugador en una partida se determina en forma aleatoria un número de 1 a 5 y se toma ese juego de preguntas para él. Cada set tiene 26 registros, uno por letra del abecedario.
Para obtener este número aleatorio (de 1 a 5) debe, en primer término y al principio de la ejecución invocar `Randomize;` (esta invocación genera una nueva secuencia)
Luego, cada vez que requiera un número aleatorio entre 1 y 5 lo obtiene ejecutando: `Random(5)+1;`
Con ese resultado se debe asegurar que es el número que tiene el atributo nro_set para leer todas las palabras de ese conjunto, la definición del registro es la siguiente:

```
type reg_palabra = record
        nro_set : integer;
        letra : char;
        palabra : string;
        consigna : string
    end;
```

Los datos que tiene el archivo se pueden ver aquí.
Este archivo está dado por la cátedra, debe ser accedido sólo de lectura (es decir, solo usando reset() y read()), y está accesible para todos.
El acceso al archivo se hace mediante `assign (archivo, '/ip2/palabras.dat');`

Estructuras dinámicas en memoria

jugadores

árbol con nombre y cantidad de partidas ganadas, ordenado por nombre, cargado en el inicio de ejecución del programa y actualizado en tres oportunidades:

- cuando se agrega un jugador (con cantidad igual cero).
- cuando un jugador gana una partida, le suma uno a la cantidad.
- cuando se elimina un jugador (**PROMOCION**)

diez_maximos_ganadores (sólo **PROMOCION**)

arreglo auxiliar de diez punteros a nodos del árbol jugadores. Es utilizado para cargar los diez jugadores que mayor cantidad de partidas ganadas tienen. Se utiliza sólo para ese proceso.

partida

arreglo de dos elementos, cada uno posee (nombre del jugador [string-20] y puntero a lista circular [rosco]).

rosco (estructura interna, accesible mediante el arreglo partida)

lista circular con el estado de las respuestas de un jugador. Cada nodo contiene: letra, palabra, consigna y respuesta del jugador (si la respuesta está [Pendiente, Acertada, Errada]).

Entregas

Primer entrega:

Subir por moodle en la Entrada que se habilitará oportunamente el diagrama de estructuras con las definiciones de constantes y tipos que utilizarán. Fecha límite 11/11/2020 23:55hs.

Si no entregó en la fecha estipulada(11/11) se restará 1(uno) punto a la nota final de su trabajo y podrá realizar la entrega hasta el 18/11 a las 23:00hs. No se aceptarán entregas fuera de esta fecha.

Tanto el diagrama de estructura como las definiciones pueden ser posteriormente modificados en base a las correcciones del docente, y será entregados nuevamente a la entrega definitiva

Entrega definitiva:

Esta entrega constará de la documentación e implementación completa del Sistema. Por lo tanto, la entrega constará de:

- El código ejecutando en el propio entorno.
- Un documento (pdf) que debe contener el Diagrama de Estructura definitivo del trabajo completo y toda documentación que considere de interés.
- El conjunto de archivos quedan en el propio entorno como para probar el sistema. Tengan en cuenta que los archivos van a estar todos en el mismo disco y por lo tanto deben diferenciar los suyos con los del resto para no romperlos sin querer, por eso proponemos que todos sus archivos estén prefijados con un mismo texto, por ejemplo apellidos y nombre, luego el nombre del archivo, por ejemplo: podría quedar **assign (archivo, '/ip2/juanperez-jugadores');**

Fecha de entrega definitiva: - 26/11 hasta las 23:55 hs. para TODOS

Importante:

Si no entregó en la fecha estipulada(26/11) se restarán 4(cuatro) puntos a la nota final de su trabajo y podrá realizar la entrega hasta el 2/12 a las 23:00hs. No se aceptarán entregas fuera de esta fecha.

Defensa del trabajo:

La defensa es **obligatoria** constará de una reunión de evaluación individual con el docente en la clase práctica del 2/12 (Cada docente comunicará el horario por maill).

La defensa del trabajo es la instancia de evaluación final y definitiva del trabajo especial; razón por la cual, cada uno deberá prepararse para la misma con todos los elementos que considere necesarios para defender adecuadamente el trabajo realizado a lo largo del cuatrimestre.

La nota final del trabajo depende de la evaluación del trabajo entregado y de la defensa. En esta defensa, el docente realizará preguntas individuales a cada alumno sobre el trabajo realizado.

Aclaraciones:

- En toda práctica podrán realizarse consultas sobre el trabajo a su docente asignado (recomendado) o a cualquier otro ayudante de la cátedra.
- No todos los trabajos llegarán a la instancia de la defensa oral, dado que el docente puede determinar que el trabajo entregado no cumple con las características mínimas requeridas para aprobar.

Otras razones por las cuales se podrá desaprobar el trabajo (y por lo tanto la cursada)

- Si su docente considera que el trabajo no está en condiciones para ser aprobado
- **Si se detecta que existe otro trabajo con código parcial o totalmente copiado (aunque el otro trabajo acepte que lo copió del suyo).**
- Si no asiste a la defensa del trabajo en el lugar y fecha que corresponde,
- Si durante la defensa final del trabajo el docente considera que usted no posee los conocimientos suficientes para aprobar, a pesar que el trabajo fue realizado exitosamente.
- Si realiza una entrega parcial, es decir que le falta información.