SUBPROGRAMAS Y ESPECIFICACIÓN

TITLE: PROGRAMMING II LABS

SUBTITLE: Práctica 2

AUTHOR 1: Raúl Fernández del Blanco LOGIN 1: r.delblanco

AUTHOR 2: Armando Martínez Noya LOGIN 2: a.mnoya

Date: 07/05/21

Raúl Fernández del Blanco
Armando Martínez Noya
1ºGEI
Grupo: 2.4.

En el programa principal del proyecto (***main.c***) encontraremos unas funciones las cuales son declaradas y definidas en él, estas funciones fueron diseñadas por los autores de este proyecto con la finalidad de conseguir el correcto funcionamiento de éste. Estas funciones son:

- **void New ( );**
- **void Delete ( );**
- **void Play ( );**
- **void Add ( );**
- **void Stats ( );**
- **void Remove ( );**

Estos subprogramas ejecutan las ordenes indicadas en los ficheros de entrada de la siguiente manera:

void New(char *commandNumber, char command, char *param1, char *param2, tUserList *lista);

1. **Objetivo:**

   Es una función generadora que se encarga de recibir la información y llamar a las funciones pertinentes para crear el nuevo usuario con la información recibida.

2. **Entradas:**

   - CommandNumber: indica el número de operación que se ejecuta.
   - Command: indica la operación a ejecutar.
   - Param1: indica el nickname del nuevo usuario.
   - Param2: indica la categoria de dicho usuario.
   - Lista: indica la lista donde se añadirá el nuevo usuario.

3. **Salidas:**

   La lista con un nuevo usuario implementado alfabéticamente.

4. **PreCondición:**

   - La lista de usuarios tiene que estar previamente inicializada.
   - El param2 debe ser válido.

5. **PostCondición:**

   Las posiciones de los elementos de la lista de usuarios han podido variar.

Raúl Fernández del Blanco
Armando Martínez Noya
1ºGEI
Grupo: 2.4.

void Delete(char *commandNumber, char command, char *param1, tUserList *lista);

1. **Objetivo:**

   Es una función destructora que se encarga de eliminar de la lista el usuario que tenga el nickname indicado.

2. **Entradas:**

   - CommandNumber: indica el número de operación que se ejecuta.
   - Command: indica la operación a ejecutar.
   - Param1: indica el nickname del usuario a eliminar.
   - Lista: indica la lista de usuarios donde se eliminará el usuario.

3. **Salidas:**

   La lista de usuarios modificada sin el elemento borrado anteriormente.

4. **PreCondición:**

   La lista de usuarios tiene que estar previamente inicializada.

5. **PostCondición:**

   Las posiciones de los elementos de la lista de usuarios posteriores a la del elemento eliminado han podido variar.

void Play(char *commandNumber, char command, char *param1, char *param2, char *param3, tUserList *lista);

1. **Objetivo:**

   Es una función modificadora que se encarga de aumentar el tiempo de reproducción del video indicado y mostrar el tiempo total de reproducción del usuario (en minutos).

2. **Entradas:**

   - CommandNumber: indica el número de operación que se ejecuta.
   - Command: indica la operación a ejecutar.
   - Param1: indica el nickname del usuario.
   - Param2: indica el nombre del video a reproducir.
   - Param3: indica el tiempo de reproducción del vídeo.
   - Lista: indica la lista en la que se encuentra el usuario introducido.

3. **Salidas:**

   La lista con el nuevo tiempo de reproducción actualizado.

4. **PreCondición:**

   Las listas de usuarios y vídeos tienen que estar previamente inicializadas.

UNIVERSIDADE DA CORUÑA

Raúl Fernández del Blanco
Armando Martínez Noya
1ºGEI
Grupo: 2.4.

void Add(char *commandNumber, char command, char *param1, char *param2, tUserList *lista);

1. **Objetivo:**

   Es una función modificadora que se encarga de añadir un video a la lista de vídeos del usuario indicado.

2. **Entradas:**

   - CommandNumber: indica el número de operación que se ejecuta.
   - Command: indica la operación a ejecutar.
   - Param1: indica el nickname del usuario.
   - Param2: indica el nombre del video a insertar.
   - Lista: indica la lista donde se encuentra el usuario.

3. **Salidas:**

   La lista modificada con el contenido del usuario actualizado.

4. **PreCondición:**

   Las listas de usuarios y vídeos tienen que estar previamente inicializadas.

5. **PostCondición:**

   Las posiciones de los elementos de la lista de videos posteriores a la del elemento insertado han podido variar.

void Stats(char *commandNumber, char command, tUserList lista);

1. **Objetivo:**

   Es una función observadora que tiene como objetivo calcular y mostrar por pantalla de manera ordenada las estadísticas de la lista.

2. **Entradas:**

   - CommandNumber: indica el número de operación que se ejecuta.
   - Command: indica la operación a ejecutar.
   - Lista: indica la lista a la que pertenecen los datos.

3. **Salidas:**

   Tiene como salida por pantalla la lista completa de usuarios actuales incluyendo todos sus vídeos, así como el número de usuarios, el número de vídeos reproducidos, la media de reproducciones para cada categoría y la cantidad total de tiempo reproducido.

4. **PreCondición:**

   Las listas de usuarios y vídeos tienen que estar previamente inicializadas.

UNIVERSIDADE DA CORUÑA

Raúl Fernández del Blanco
Armando Martínez Noya
1ºGEI
Grupo: 2.4.

void Remove(char *commandNumber, char command, char *param1,

tUserList *lista)

1. **Objetivo:**

   Es una función destructora que se encarga de eliminar de la lista a todo usuario de categoría standard que haya excedido el tiempo máximo de reproducción.

2. **Entradas:**

   - CommandNumber: indica el número de operación que se ejecuta.
   - Command: indica la operación a ejecutar.
   - Param1: indica el tiempo máximo de reproducción.
   - Lista: indica la lista de usuarios donde se eliminarán los usuarios.

3. **Salidas:**

   La lista de usuarios modificada sin los usuarios standard que superen el límite.

4. **PreCondición:**

   La lista de usuarios tiene que estar previamente inicializada.

5. **PostCondición:**

   Las posiciones de los elementos de la lista de usuarios posteriores a la de los elementos eliminados han podido variar.

Además, en nuestro programa principal podremos encontrarnos con tres funciones las cuales se encargan de leer los ficheros de entrada, llamar a nuestros subprogramas definidos anteriormente y optimizar el código. Estas son:

int main (int nargs, char **args);

La función main de nuestro programa se encarga de seleccionar y enviar el documento que leerá posteriormente.

void readTasks (char *filename);

La función recibe como entrada el fichero que queremos que utilice el programa. Esta, además de inicializar la lista, se encarga de leer línea a línea dicho fichero y extraer las variables necesarias para el correcto funcionamiento del programa. Las variables mencionadas anteriormente, son las introducidas en la siguiente función:

Raúl Fernández del Blanco
Armando Martínez Noya
1ºGEI
Grupo: 2.4.

void processCommand (char *commandNumber, char command, char *param1, char *param2, char *param3, tList *list);

Esta función es muy sencilla, ya que simplemente se encarga de elegir cuál de los subprogramas previamente analizados se va a ejecutar y de enviarles la información pertinente para su correcto funcionamiento.

void Header(char *commandNumber, char command, char dosP, char *nick_maxtime, char *param1, char *video_cat, char *param2, char *minut, char *param3);

Por último, la función Header se encarga de mostrar por pantalla la cabecera necesaria para cada una de las funciones principales, optimizando así el programa y evitando la repetición de código.