Práctica 2: Gestor de preguntas y respuestas

Programación 2

Curso 2023-2024

Esta práctica del curso consiste en el desarrollo de un sistema de gestión de preguntas y respuestas sobre tu asignatura favorita. Los conceptos necesarios para desarrollar esta práctica se trabajan en los *Temas 1*, *2* y *3* de teoría.

Condiciones de entrega

- La fecha límite de entrega para esta práctica es el **viernes 19 de abril**, hasta las **23:59**
- Debes entregar dos ficheros: `preguntas.txt` con un conjunto de preguntas de ejemplo y `prac2.cc` con el código de todas las funciones

Código de honor

Si se detecta copia (total o parcial) en tu práctica, tendrás un **0** en la entrega y se informará a la dirección de la Escuela Politécnica Superior para que adopte medidas disciplinarias

Está bien discutir con tus compañeros posibles soluciones a las prácticas
Está bien apuntarte a una academia si sirve para obligarte a estudiar y hacer las prácticas

Está mal copiar código de otros compañeros o pedirle a ChatGPT que te haga la práctica
Está mal apuntarte a una academia para que te hagan las prácticas

Si necesitas ayuda acude a tu profesor/a
No copies

Normas generales

- Debes entregar la práctica exclusivamente a través del servidor de prácticas del Departamento de Lenguajes y Sistemas Informáticos (DLSI). Se puede acceder a él de dos maneras:
  - Página principal del DLSI (`https://www.dlsi.ua.es`), opción "ENTREGA DE PRÁCTICAS"
  - Directamente en la dirección `https://pracdlsi.dlsi.ua.es`
- Cuestiones que debes tener en cuenta al hacer la entrega:
  - El usuario y la contraseña para entregar prácticas son los mismos que utilizas en UACloud
  - Puedes entregar la práctica varias veces, pero sólo se corregirá la última entrega
  - No se admitirán entregas por otros medios, como el correo electrónico o UACloud
  - No se admitirán entregas fuera de plazo

- Tu práctica debe poder ser compilada sin errores con el compilador de C++ existente en la distribución de Linux de los laboratorios de prácticas
- Si tu práctica no se puede compilar su calificación será 0
- El 70% de la nota de la práctica dependerá de la corrección automática, por lo que es imprescindible que respetes estrictamente los textos y los formatos de salida que se indican en este enunciado. El otro 30% dependerá de la revisión manual del código que haga tu profesor de prácticas, por lo que debes también ajustarte a la guía de estilo de la asignatura. Se tendrá en cuenta también para la nota el contenido del fichero de preguntas solicitado. Como en prácticas anteriores, también se valorará que hayas actualizado tu código en *GitHub* todas las semanas
- Al comienzo de todos los ficheros fuente entregados debes incluir un comentario con tu NIF (o equivalente) y tu nombre. Por ejemplo:

prac2.cc

```
// DNI 12345678X GARCIA GARCIA, JUAN MANUEL
...
```

- El cálculo de la nota de la práctica y su relevancia en la nota final de la asignatura se detallan en las transparencias de presentación de la asignatura (*Tema 0*)

1 Descripción de la práctica

Se desea desarrollar un sistema de resolución de preguntas y respuestas sobre los temas de la asignatura *Programación 2*. El objetivo del sistema es el que los estudiantes puedan realizar consultas sobre un tema concreto y que los profesores dados de alta en el programa puedan contestarlas. La herramienta permitirá gestionar tanto las preguntas y respuestas como los perfiles de los profesores.

2 Detalles de implementación

En el Moodle de la asignatura se publicarán varios ficheros que necesitarás para la correcta realización de la práctica:

- `prac2.cc`. Este fichero contiene un esqueleto de programa sobre el que realizar tu práctica. Descárgalo y añade tu código en él. Este fichero contiene la siguiente información:
  - Los registros (`struct`) necesarios para hacer la práctica
  - Un tipo enumerado `Error` que contiene todas las posibles clases de error que se pueden dar en esta práctica (por ejemplo, `ERR_OPTION`)
  - Una función `error` que se encarga de mostrar el correspondiente mensaje de error por pantalla en función del parámetro que se le pase. Por ejemplo, cuando la función recibe el parámetro `ERR_OPTION`, mostrará por pantalla el mensaje `ERROR: wrong menu option`
  - Una función `showMenu` que muestra por pantalla el menú del programa
  - Una función `main` que implementa la gestión del menú principal y llama a las funciones correspondientes dependiendo de la opción elegida por el usuario
- `autocorrector-prac2.tgz`. Contiene los ficheros del autocorrector para probar la práctica con algunas pruebas de entrada. La corrección automática de la práctica se realizará con un programa similar, con estas pruebas y otras más definidas por el profesorado de la asignatura
- `prac2`. Fichero ejecutable de la práctica (compilado para máquinas Linux de 64 bits) desarrollado por el profesorado de la asignatura, para que puedas probarlo con las entradas que quieras y ver la salida correcta esperada

- En la corrección de la práctica se introducirán siempre datos del tipo correcto, aunque con valores que pueden ser incorrectos. Por ejemplo, si se pide la puntuación de un profesor, se probará siempre con un valor entero, que podría ser `-1237` o `0`, pero nunca se introducirá un valor de otro tipo (carácter, `string`, número real, ...)
- Para evitar problemas con la codificación de caracteres, en ninguno de los textos que se introduzcan para evaluar esta práctica (preguntas, respuestas, nombres de profesores, etc.) se emplearán acentos, diéresis, interrogante de apertura o la letra ñ. Solo se usarán los caracteres del alfabeto inglés
- El resto de decisiones en la implementación de la práctica quedan a tu criterio, pero ten en cuenta que el código fuente será revisado por tu profesor de prácticas siguiendo la guía de estilo publicada en el Moodle de la asignatura. Parte de la nota de la práctica depende de dicha revisión

3 Funcionamiento del programa

El programa que vas a desarrollar consiste en un sistema de tutorías que te permitirá gestionar preguntas, respuestas y profesores, almacenándolos para su uso posterior utilizando ficheros. Los usuarios de la aplicación que vas a desarrollar serán estudiantes, que serán los que dan de alta las preguntas, y los profesores, que serán los que contesten dichas dudas.

4 Componentes

El programa a desarrollar debe poder gestionar tres elementos fundamentales: las preguntas formuladas por los estudiantes, los profesores, que son cada uno de los docentes encargados de contestar a las preguntas, y la base de datos, que es donde se almacena toda la información de las preguntas y los profesores. Los siguientes apartados describen la estructura de cada uno de estos componentes en más detalle.

4.1 Question

Las estructuras de tipo `Question` almacenan la información relativa a las preguntas realizadas por los estudiantes.

Cada pregunta contendrá un identificador numérico (`id`), número de tema al que corresponde (`unit`), el texto de la pregunta (`question`) y el texto de la respuesta (`answer`). Esta información se almacenará en un registro con el siguiente formato:

```
struct Question{
    unsigned int id;
    unsigned int unit;
    string question;
    string answer;
};
```

4.2 Teacher

Las estructuras de tipo `Teacher` almacenan la información relativa a los profesores. Cada profesor tendrá un nombre (`name`), su contraseña encriptada (`password`) y el número de preguntas que han contestado en el programa (`answered`). Esta información se almacenará en un registro con el siguiente formato:

```
struct Teacher{
    char name[KMAXNAME];
    char password[KMAXPASSWORD];
    unsigned int answered;
};
```

`KMAXNAME` es una constante entera con el valor `50` y `KMAXPASSWORD` es también una constante entera con el valor `5`.

4.3 Database

La estructura `Database` almacena toda la colección de preguntas y profesores. Contiene además un campo `nextId` que almacenarán el identificador que deberá asignarse al campo `id` de la siguiente pregunta (`Question`) que se cree. Este campo tendrá valor `1` al inicio del programa y se incrementará de uno en uno para cada nueva pregunta. Es decir, la primera pregunta que se cree tendrá como identificador `1`, la segunda tendrá el `2` y así sucesivamente.

```
struct Database{
  unsigned int nextId;
  vector<Question> questions;
  vector<Teacher> teachers;
};
```

5 Inicio del programa

El `main` de tu práctica contendrá una variable de tipo `Database` que almacenará en memoria toda la información de las preguntas y profesores durante la ejecución del programa. Cuanto se inicie el programa, el campo `nextId` de esta variable se inicializará a `1`.

Lo primero que deberás hacer es cargar en memoria los datos de los profesores almacenados en el fichero binario `teachers.bin`. Este fichero contiene estructuras de tipo `Teacher`, una detrás de otra, de manera que en cada una se almacena la información de un profesor. Deberás crear una función `loadTeachers` que se encargue de leer todos la información de los profesores y cargarla en el campo `teachers` de la variable de tipo `Database` mencionada anteriormente.

- Tu programa sólo gestionará una única variable de tipo `Database` que contendrá todos los datos del preguntas y profesores
- Si al iniciar el programa el fichero `teachers.bin` no existe no debes mostrar ningún tipo de error. El programa iniciará con normalidad dejando el vector `teachers` de `Database` vacío
- Asumiremos que el contenido del fichero `teachers.bin` es correcto. No hay que hacer ningún tipo de comprobación sobre el contenido durante la carga

6 Menú

Al ejecutar la práctica, y después de cargar la información de los profesores (si la hay), se mostrará por pantalla el menú principal del programa, quedando a la espera de que el usuario elija una opción:

Terminal

```
1- Add question
2- Batch add questions
3- Delete question
4- Add teacher
5- Add answers
6- View answers
7- View statistics
8- Export questions
q- Quit
Option:
```

Las opciones válidas que puede introducir el usuario son los números del 1 al 8 y la letra q. Si la opción elegida no es ninguna de éstas (por ejemplo un 9, una x o un ;), se emitirá el error ERR_OPTION llamando a la función error con dicho parámetro. Cuando el usuario elige una opción correcta, se debe ejecutar el código asociado a dicha opción. Al finalizarla, volverá a mostrar el menú principal y a pedir otra opción, hasta que el usuario decida salir del programa utilizando la opción q.

7 Opciones

En los siguientes apartados se describe el funcionamiento que deberá tener cada una de las opciones que se muestran en el menú principal del programa.

7.1 Add question

Esta opción permite añadir una pregunta nueva al programa. Se activa cuando el usuario elige la opción 1 del menú principal. Se deberá crear una función addQuestion para gestionarla. En ella se pedirá al usuario que introduzca el número de tema mostrando el siguiente mensaje:

Terminal

```
Enter unit:
```

Si se introduce la cadena vacía se mostrará el error ERR_EMPTY y se volverá al menú principal sin guardar ningún tipo de información de la pregunta. En caso contrario, deberá comprobarse que el valor introducido esté entre el 1 y el 5. Si no es así, se emitirá mensaje ERR_UNIT y se volverá a pedir al usuario que introduzca el número mostrando el mismo mensaje.

Si el número es correcto, se pedirá el texto de la pregunta mostrando el siguiente mensaje:

Terminal

```
Enter question:
```

Se deberá validar que el texto introducido no contenga ningún carácter de barra vertical (|) para evitar problemas a la hora de leer y escribir el fichero de texto que se explicará en la siguiente sección. En caso de contener una barra vertical, se mostrará el error ERR_CHAR y se volverá a pedir la pregunta mostrando el mismo mensaje anterior. En caso de que la cadena introducida esté vacía, se mostrará el error ERR_EMPTY y se volverá al menú principal sin guardar nada.

Una vez leído el número de tema y la pregunta correctamente, se creará una nueva pregunta (Question) con el campo answer vacío, ya que todavía no tendrá una respuesta asociada.

A cada nueva pregunta se le asignará de manera automática un identificador único que se almacenará en el campo id de Question. Este identificador vendrá dado por el valor que haya en ese momento almacenado en el campo nextId de la estructura Database que contiene toda la información del programa. Como se indicó en la Sección 4.3, este identificador único comenzará con el valor 1 y se incrementará de uno en uno para cada nueva pregunta. Por lo tanto, cada vez que añadas una pregunta, deberás incrementar el campo nextId para que a la siguiente pregunta se le asigne un nuevo identificador correcto.

Tras asignar el identificador, se incorporará la nueva pregunta al vector questions del registro Database del programa y se volverá al menú principal.

- Se va a permitir que existan preguntas duplicadas, por lo que no es necesario comprobar si una pregunta ya existe cuando se introduce

7.2 Batch add questions

Esta opción permite cargar preguntas de un fichero de texto. Se activa cuando el usuario elige la opción 2 del menú principal. Su código se deberá incluir en una función llamada batchAddQuestions. Cada línea del fichero de texto tendrá el siguiente formato:

`Número de tema|Pregunta|Respuesta`

En primer lugar, se solicitará el nombre del fichero de texto que se desea cargar con el siguiente mensaje:

Terminal

```
Enter filename:
```

Si se introduce la cadena vacía se mostrará el error `ERR_EMPTY` y se volverá al menú principal. Si no se ha podido abrir el fichero con el nombre introducido se mostrará el error `ERR_FILE` y se volverá a pedir el nombre mostrando el mismo mensaje anterior.

Si el fichero se ha podido abrir correctamente, se iniciará el proceso de insertar en el sistema las preguntas. Se deberá comprar que las preguntas son correctas haciendo las siguientes validaciones:

- El número de tema deberá ser un valor entre 1 y 5. No puede estar vacío
- El texto de la pregunta no puede estar vacío
- El texto de la respuesta puede estar vacío. En ese caso, el formato correcto de la línea sería `Número de tema|Pregunta`. Es decir, hay que comprobar que no hay una barra vertical detrás de la pregunta
- No pueden haber más de dos barras verticales separadoras, porque implicaría que hay más campos de los esperados y el formato no sería correcto
- Tendrá que haber al menos una línea vertical para que el formato sea correcto

Si alguna de estas validaciones falla se mostrará el mensaje `Error line X`, donde `X` es el número de línea del fichero, y se ignorará el contenido de la línea leída pasando automáticamente a leer la siguiente sin hacer más comprobaciones. Es decir, sólo se mostrará un error por pregunta, aunque haya más de uno en la misma línea (por ejemplo, el tema puede estar mal y la pregunta vacía).

Si la línea del fichero leída está vacía se ignorará sin mostrar ningún error. Por cada línea correcta se deberá crear una nueva pregunta en la base de datos, incrementando el identificador `nextId` tal y como se explicaba en la sección anterior.

Al final del proceso se mostrará un mensaje indicando cuántas preguntas tenía el fichero y cuántas han sido correctamente introducidas en el sistema. Por ejemplo, dado el siguiente fichero:

questions.txt

```
1|Que es una matriz?|Un array bidimensional
1|No entiendo lo que es un booleano|

2||La pregunta esta vacia
1|Para que sirve break?
5|Como se implementa la herencia|No se estudia en esta asignatura
7|||
```

Se obtendría la siguiente salida por pantalla:

Terminal

```
Error line 2
Error line 4
Error line 7
Summary: 3/6 questions added
```

El error en la línea 2 se produce porque hay una barra vertical innecesaria al final. La línea 3 se ignora simplemente al estar vacía. La línea 4 da error porque el texto de la pregunta está vacío. La línea 7 da error porque no existe el tema 7. Además la pregunta está vacía y hay más líneas verticales de las admitidas, pero solo se muestra una vez el error. Por tanto, en este ejemplo se añadirían a la base de datos 3 de las 6 preguntas que había en el fichero.

Si el fichero estuviera vacío se obtendría la siguiente salida, sin mostrar ningún error:

```
Terminal
SUMMARY: 0/0 questions added
```

7.3 Delete question

Esta opción permite borrar una pregunta dado su identificador. Se activa cuando el usuario elige la opción 3 del menú principal. Se deberá crear una función `deleteQuestion` para manejar esta funcionalidad. En primer lugar se pedirá el identificador de la pregunta con el mensaje:

```
Terminal
Enter question id:
```

Si no existe una pregunta con ese identificador, se emitirá el error `ERR_ID` y se volverá a pedir el identificador mostrando el mismo mensaje. Si se introduce la cadena vacía se mostrará el error `ERR_EMPTY` y se volverá al menú principal. Si el identificador es correcto, se eliminará la pregunta del vector `questions` del registro `Database` y se volverá al menú principal.

7.4 Add teacher

Esta opción permite dar de alta los profesores que responderán las preguntas de los estudiantes. Se activa cuando el usuario elige la opción 4 del menú principal. Su código se deberá incluir en una función llamada `addTeacher`. En primer lugar se pedirá el nombre del profesor con el siguiente mensaje:

```
Terminal
Enter teacher name:
```

Se deberá validar el nombre del profesor siguiendo estas reglas:

- Sólo podrá contener letras mayúsculas (A-Z), minúsculas (a-z) y espacios en blanco. De nuevo, sólo se usarán los caracteres del alfabeto inglés. Una forma de hacer esta comprobación es mediante la función `isalpha` de la librería `cctype`
- Si contiene espacios en blanco al principio o al final del nombre se deberán eliminar
- La longitud mínima del nombre, después de eliminar los espacios al principio y al final, será de 3 caracteres y la máxima de `KMAXNAME-1` (recuerda que hay que dejar un espacio para el carácter '`\0`')

Si no se cumple alguna de estas reglas se mostrará el error `ERR_NAME` y se volverá a pedir el nombre mostrando el mismo mensaje anterior. Si la cadena está vacía, se mostrará el error `ERR_EMPTY` y se volverá al menú principal. Si el nombre es válido se comprobará que no exista ya ningún otro profesor con el mismo nombre. En caso contrario, se mostrará el error `ERR_DUPLICATED` y se volverá a pedir al usuario que introduzca el nombre mostrando el mismo mensaje anterior.

A continuación se pedirá la contraseña que utilizará el profesor para identificarse mostrando el siguiente mensaje:

```
Terminal
Enter password:
```

Sólo se considerarán válidas las contraseñas formadas exclusivamente por 4 dígitos (0-9). En cualquier otro caso se mostrará el error `ERR_PASSWORD` y se volverá a pedir la contraseña mostrando el mismo mensaje. Para hacer esta comprobación puedes usar la función `isdigit` de la librería `cctype`. Si la cadena está vacía, se mostrará el error `ERR_EMPTY` y se volverá al menú principal.

Si la contraseña es válida, antes de guardarla en la información del profesor se encriptará transformando los dígitos a caracteres usando la siguiente tabla de equivalencia:

Dígito	0	1	2	3	4	5	6	7	8	9
Letra	T	R	W	A	G	M	Y	F	P	D

Por ejemplo, la contraseña `0375` se encriptará como `TAFM`. Una vez validados los datos y encriptada la contraseña, se inicializará a `0` el contador de preguntas respondidas del profesor y se guardará al final del vector `teachers` del registro `Database`.

7.5 Add answers

Esta opción permite a los profesores añadir respuestas a las preguntas que todavía no se hayan respondido. Se activa cuando el usuario elige la opción `5` del menú principal. Su código se deberá incluir en una función llamada `addAnswers` que tendrás que implementar.

Para poder añadir respuestas, los profesores deberán acceder usando su nombre y contraseña. Primero se pedirá el nombre del profesor con el siguiente mensaje:

Terminal
```
Enter teacher name:
```

Se deberá comprobar si el nombre del profesor existe en la base de datos (no es necesario validarlo antes) y si no se encuentra se mostrará el error `ERR_NAME` y se volverá a pedir el nombre mostrando el mismo mensaje. Si se introduce la cadena vacía, se mostrará el error `ERR_EMPTY` y se volverá al menú principal. Si el nombre es correcto se pedirá la contraseña con el siguiente mensaje:

Terminal
```
Enter password:
```

¡Recuerda que la contraseña se guardó de manera encriptada! Si la contraseña no coincide con la que hay almacenada, se mostrará el error `ERR_PASSWORD` y se volverá a pedir mostrando el mismo mensaje. Si la cadena introducida está vacía, se mostrará el error `ERR_EMPTY` y se volverá al menú principal.

Si las credenciales del profesor son correctas se iniciará un proceso repetitivo de contestar preguntas. Para ello se visualizarán en pantalla las preguntas que todavía no tienen respuesta con el formato:

```
ID. (Tema) Pregunta
```

y a continuación el siguiente mensaje solicitando el identificador de la pregunta a contestar:

Terminal
```
Enter quetion id:
```

En el siguiente ejemplo puedes ver una posible salida con dos preguntas pendientes de contestar, con identificadores 2 y 5, pertenecientes al tema 1 y al 3, respectivamente:

Terminal

```
2. (1) Para que sirve break?
5. (3) Como se abre un fichero de texto?
Enter question id:
```

Si no hubiera ninguna pregunta sin contestar se mostrará el error `ERR_NO_QUESTIONS` y se volvería al menú principal.

Si el identificador introducido no corresponde a una de las preguntas listadas, se mostrará el error `ERR_ID` y se volverá a solicitar el identificador mostrando de nuevo el mensaje `Enter question id:`, pero sin mostrar de nuevo la lista de preguntas. Si se introduce la cadena vacía, se mostrará el error `ERR_EMPTY` y se volverá al menú principal. Si el identificador es correcto se pedirá la respuesta con el siguiente mensaje:

Terminal

```
Enter answer:
```

La respuesta no podrá contener el carácter de barra vertical (`'|'`), de lo contrario se mostrará el error `ERR_CHAR` y se volverá a mostrar el mensaje anterior para pedir otra vez la respuesta. Si se introduce la cadena vacía, se mostrará el error `ERR_EMPTY` y se volverá al menú principal.

Si la respuesta es válida, se guardará en la pregunta correspondiente y se volverá a mostrar la lista de preguntas pendientes de contestar y a solicitar un identificador. Este proceso terminará cuando se introduzca el carácter `'b'` en lugar de un identificador válido o cuando ya no queden preguntas por contestar. En ambos casos se volverá al menú principal.

7.6 View answers

Esta opción permite visualizar todas las preguntas que ya se hayan resuelto con su respuesta. Se activa cuando el usuario elige la opción `6` del menú principal. Su deberá crear una función `viewAnswers` con el código de esta opción.

Al iniciar esta opción, las preguntas se mostrarán con el siguiente formato:

```
Id. (Tema) Pregunta: Respuesta
```

Por ejemplo:

Terminal

```
2. (1) Para que sirve break?: Por ejemplo, para salir de un bucle
5. (3) Como se abre un fichero de texto?: Mira la transparencia 8 del Tema 3
6. (3) Como se cierra un fichero?: Transparencia 11 del Tema 3
7. (1) Que significa int?: Tipo entero, burro
12. (2) Que es un string?: Una clase para manejar cadenas facilmente
```

Tras esto se volverá al menú principal.

7.7 View statistics

Esta opción permite visualizar estadísticas sobre el número de preguntas y cuántas ha contestado cada profesor. Se activa cuando el usuario elige la opción `7` del menú principal. Se deberá crear una función `viewStatistics` para gestionar esta opción.

A continuación puedes ver un ejemplo de salida por pantalla:

Terminal

```
Total number of questions: 10
Number of questions answered: 7
Grace Hopper: 2
Ada Lovelace: 4
Kim Dotcom: 0
Tim Berners Lee: 1
```

Los profesores se mostrarán en el orden en el que fueron introducidos en el sistema. Si no hay ninguna pregunta, o hay preguntas pero no están respondidas, las líneas correspondientes mostrarán el valor `0`. En este ejemplo no hay preguntas, ni respuestas, ni profesores:

Terminal

```
Total number of questions: 0
Number of questions answered: 0
```

7.8 Export questions

Esta opción permite exportar todas las preguntas de la base de datos a un fichero de texto. Se activa cuando el usuario elige la opción `8` del menú principal. Su código se deberá incluir en una función llamada `exportQuestions` que deberás implementar. En primer lugar se pedirá el nombre del fichero con el siguiente mensaje:

Terminal

```
Enter filename:
```

Si se introduce un nombre de fichero vacío se mostrará el error `ERR_EMPTY` y se volverá al menú principal. Si el fichero introducido no se puede abrir se mostrará el error `ERR_FILE` y se volverá a pedir el nombre mostrando el mismo mensaje anterior.

Una vez abierto el fichero, deberás guardar todas las preguntas de la base de datos, tanto las contestadas como las que no, cada una en una línea siguiendo el formato correcto descrito en la Sección 7.2. Se guardarán en el mismo orden en el que fueron introducidas.

7.9 Fin de la aplicación

Al seleccionar la opción `q` el programa terminará, guardando en el fichero binario el listado actualizado de profesores. Para esto deberás sobreescribir el contenido del fichero `teachers.bin` con el contenido del vector `teachers`, en el mismo orden en que se encuentren en el vector. Si el fichero no se puede abrir, deberá mostrarse el error `ERR_FILE` y salir del programa igualmente.

Esta acción deberá hacerse siempre que termine el programa, aunque no se haya añadido ningún profesor nuevo durante la ejecución.

8 Argumentos de programa

En esta práctica el programa debe permitir al usuario indicar algunas acciones a realizar mediante el paso de argumentos por línea de comando. Deberás utilizar los parámetros `int argc` y `char *argv[]` de la función `main` para poder gestionarlos. El programa admitirá los siguientes argumentos por línea de comando:

- `-f <fichero texto>`. Permite importar preguntas de un fichero de texto, cuyo nombre deberá indicarse tras el argumento `-f`, iniciando el programa a continuación. Se comportará igual que si el usuario seleccionara la opción `Batch add questions` del menú, también a la hora de tratar los errores si alguno de los datos leídos es incorrecto. Por ejemplo, si se lee del fichero una pregunta con el texto vacío, se emitirá el error `ERR_EMPTY`, se ignorarán los datos de esa pregunta y se seguirá procesando el resto del fichero. Si no se puede abrir el fichero se emitirá el error `ERR_FILE` y se iniciará el programa con normalidad sin cargar ningún dato. Si los datos del fichero se cargan correctamente, se mostrará el mensaje mostrando cuántas preguntas se han importado, igual que se hacía en la Sección 7.2. El siguiente ejemplo importará los datos almacenados en el fichero de texto `questions.txt`:

Terminal
```
$ prac2 -f questions.txt
```

- `-s`. Muestra las estadísticas, igual que si el usuario seleccionara la opción `View statistics` del menú, saliendo del programa a continuación sin llegar a mostrar el menú principal

Los dos argumentos pueden aparecer al mismo tiempo en una llamada al programa y además en cualquier orden. Independientemente del orden en el que aparezcan en la línea de comando, el orden en el que se procesarán los argumentos será el siguiente:

1. En primer lugar se procesará la opción `-f` para cargar datos del fichero
2. En segundo lugar se ejecutará la opción `-s` para mostrar las estadísticas

En el siguiente ejemplo, en primer lugar se cargarán los datos del programa almacenados en el fichero `questions.txt` y a continuación se mostrarán las estadísticas, saliendo a continuación del programa.

Terminal
```
$ prac2 -s -f questions.txt
```

Otro ejemplo, cuyos parámetros son válidos, sería el siguiente:

Terminal
```
$ prac2 -f -f -s
```

Aunque a primera vista los parámetros pudieran parecer incorrectos, en realidad no lo son. El programa recibe un primer parámetro `-f` que va seguido de un fichero que se llama `-f`. A continuación recibe el parámetro `-s` para mostrar las estadísticas. Esta secuencia de parámetros es, por tanto, correcta.

En caso de que se produzca un error en los argumentos de entrada (por ejemplo, que se introduzca una opción que no existe o se meta una duplicada), se deberá emitir el error `ERR_ARGS` y se finalizará el programa sin llegar a mostrar el menú principal. Si todos los argumentos son correctos, se procesarán las opciones introducidas por el usuario y después se mostrará el menú principal de programa de la forma habitual (siempre que no se haya pasado la opción `-s`).

- Independientemente de si se pasan argumentos del programa o no, siempre se deberá cargar el fichero `teachers.bin` al inicio del programa, antes de cargar las preguntas y antes de mostrar estadísticas, si fuera el caso
- Los dos argumentos son opcionales y pueden no aparecer en la llamada al programa
- Un determinado argumento sólo puede aparecer una vez en la llamada, de lo contrario se considerará que los argumentos son incorrectos
- Si aparece algún argumento diferente a los mencionados (por ejemplo, `-n`) se considerará que los argumentos son incorrectos
- Si detrás de `-f` no aparece el nombre del fichero, los argumentos serán incorrectos también
- Los ficheros no tienen por qué tener necesariamente la extensión `.txt`. Utilizamos estas extensiones en los ejemplos para que quede más claro que estamos trabajando con ficheros de texto

9 El fichero `preguntas.txt`

Como parte de esta práctica debes entregar un fichero `preguntas.txt` que deberá incluir 10 preguntas realizadas por ti sobre los contenidos de la asignatura `Programación 2`. **Esmérate, porque estas preguntas serán revisadas por el profesorado y se tendrán en cuenta para la puntuación de la corrección manual**. Es decir, no pongas chorradas ni preguntas de ejemplo sacadas de este documento.

Específicamente, deberás realizar dos preguntas sobre cada uno de los cinco temas de la asignatura. El formato del fichero será el siguiente:

```
Tema|Pregunta
```

Las preguntas irán ordenadas por tema, es decir, primero vendrán las dos preguntas del Tema 1, luego las dos del Tema 2 y así sucesivamente. Un ejemplo de las primeras cinco líneas del fichero podría ser este:

```
1|¿Para qué sirve un break?
1|¿En qué orden se ejecutan los parámetros de un bucle for?
2|¿Cómo se copia el contenido de un string en un array de caracteres?
2|¿Para qué sirve srtcat?
3|¿Qué pasa si abro un fichero de escritura con el flag ios::in?
```

- En este fichero la pregunta puede contener los caracteres que quieras (eñes, acentos, interrogante de apertura...) excepto la barra vertical