Facultad de Informática – Universidad Complutense

Fundamentos de la programación – Grupos D,F,G,H,DG

Curso 2014–2015

Examen final –10 de septiembre de 2015

Tiempo disponible: 3 horas

Se pide construir un programa modular que permita a un estudiante realizar exámenes de autoevaluación, con respuesta numérica, a partir de una lista maestra de enunciados que se almacena en un fichero. El programa constará al menos de tres módulos: *ListaEnunciados*, *Examen* y módulo principal

Módulo *ListaEnunciados* (3 puntos)

Máx. 50 preguntas

Declara un tipo de estructura `tEnunciado` con tres campos: identificador, texto del enunciado y solución (un número). Declara también un tipo de estructura `tListaEnunciados` para la lista maestra de enunciados disponibles. Esta lista, de tamaño variable, estará implementada con un **array estático de punteros a variables dinámicas**. Además, estará **ordenada** por el identificador.

Implementa, al menos, los siguientes subprogramas:

- ✓ `buscar()`: Dada una lista de enunciados y un identificador, si hay un enunciado en la lista con ese identificador devuelve cierto y la posición del enunciado en la lista. En otro caso devuelve falso y la posición que debería ocupar.
- ✓ `insertar()`: Dada una lista de enunciados y un enunciado, si el identificador de éste no se encuentra ya en la lista, lo inserta en su lugar y devuelve cierto. En otro caso no se inserta y devuelve falso. Si no cabe devuelve falso.
- ✓ `cargar()`: Dada una lista de enunciados le añade los enunciados del archivo `enunciados.txt`. El archivo termina con el centinela XXX y cada enunciado utiliza tres líneas: identificador, texto del enunciado y solución. Ejemplo de archivo:

```
MD01
¿Cuántos datos se pueden codificar con 4 bits?
16
CB00
Escribe 8 en binario
1000
...
XXX
```

Módulo *Examen* (5 puntos)

Máx. 50 preguntas

Declara un tipo de estructura `tPregunta` con tres campos: identificador, respuesta y resultado (incorrecta, correcta o pendiente). Declara también un tipo `tExamen` para listas de tamaño variable de preguntas, implementadas con **array dinámico**. Además de la lista de preguntas guardará el número total de aciertos.

Implementa, al menos, los siguientes subprogramas:

- ✓ `iniciar()`: Devuelve un examen, sin preguntas, adecuadamente inicializado.
- ✓ `nuevaPregunta()`: Dada la lista de enunciados y un examen, selecciona aleatoriamente un enunciado de la lista de enunciados que no forme parte del examen; y a continuación asigna el identificador del enunciado obtenido a la nueva pregunta, pide una respuesta y valida si ésta es correcta o incorrecta. Devuelve la nueva pregunta con los campos actualizados.
- ✓ `insertar()`: Dado un examen y una pregunta la inserta al final del examen, actualizando todos los campos del examen adecuadamente.
- ✓ `mostrar()`: Dado un examen y la lista de enunciados muestra en la pantalla el número de preguntas, la lista de preguntas (los enunciados con las respuestas y si son correctas o no) y el número total de aciertos. Ejemplo:

```
Preguntas: 3
1.- Escribe 8 en hexadecimal
    8 (Correcta)
2.- Escribe 8 en binario
    101 (Incorrecta)
3.- ¿Cuántos datos se pueden codificar con 4 bits?
    16 (Correcta)
Aciertos: 2
```

- ✓ `guardar()`: Dado un examen guarda en el archivo `examen.txt` el número de preguntas y las preguntas del examen. Debe implementarse de forma **recursiva** una función `guardarPreguntas()`, con parámetros adicionales, que guarda las preguntas del examen (cada pregunta en una línea: identificador, respuesta y resultado). Ejemplo de archivo:

```
3
CB10 8 1
CB00 101 0
MD01 16 1
```

guardarPreguntas()

Módulo principal (1 punto)

Carga los enunciados del archivo `enunciados.txt` en la lista maestra de enunciados, inicia un examen vacío y muestra al usuario un menú con tres opciones: `Nueva pregunta` (añade una pregunta al examen), `Mostrar examen` (muestra el estado actual del examen) y `Salir`. El menú deberá mostrarse sucesivamente hasta que el usuario seleccione la opción `Salir`. Antes de terminar guarda el examen en el archivo `examen.txt`.

Instrucciones adicionales

Se valorará la estructuración y el uso adecuado de los esquemas de recorrido y búsqueda.

Se valorará la legibilidad, así como la comunicación entre subprogramas y uso adecuado de los módulos y de la memoria (1 punto).

Entrega el código del programa en la tarea `Entrega Examen Septiembre` del Campus Virtual (sólo .cpp y .h, comprimidos en un ZIP). ¡Asegúrate de entregar una versión sin errores de compilación!

Ejemplo de ejecución

Ejemplo de ejecución

```
Examen de autoevaluación
 1-Nueva pregunta
 2-Mostrar examen
 0-Salir
Opción: 2
 Preguntas: 0
 Aciertos: 0
Examen de autoevaluación
 1- Nueva pregunta
 2- Mostrar examen
 0- Salir
Opción: 1
 Escribe 8 en hexadecimal
 Respuesta: 8
   Correcta
Examen de autoevaluación
 1-Nueva pregunta
 2-Mostrar examen
 0-Salir
Opción: 1
 Escribe 8 en binario
 Respuesta: 101
   Incorrecta
Examen de autoevaluación
 1-Nueva pregunta
 2-Mostrar examen
 0-Salir
Opción: 1
 ¿Cuántos datos se pueden codificar con 4 bits?
 Respuesta: 16
   Correcta
Examen de autoevaluación
 1-Nueva pregunta
 2-Mostrar examen
 0-Salir
Opción: 2
 Preguntas: 3
 1.- Escribe 8 en hexadecimal
      8 (Correcta)
 2.- Escribe 8 en binario
     101 (Incorrecta)
 3.- ¿Cuántos datos se pueden codificar con 4 bits?
     16 (Correcta)
 Aciertos: 2
Examen de autoevaluación
…
```