Altas/Bajas/Modificaciones (ABM)

Algoritmos y Programación I (95.11/75.02)

4 de octubre de 2017

1. Objetivo del TP

El objetivo del presente trabajo consiste en la realización de un conjunto de aplicaciones en modo consola, escritos en lenguaje ANSI-C89, que permitan manipular un conjunto de archivos que contengan datos que deben ser actualizados utilizando Altas/Bajas/Modificaciones (ABM).

2. Alcance del TP

Mediante el presente TP se busca que el/la estudiante adquiera y aplique, conocimientos sobre los siguientes temas, además de los vistos anteriormente:

- Arreglos/Vectores
- Memoria dinámica
- Argumentos en línea de órdenes (CLA)
- Estructuras
- Archivos
- Modularización

Fecha de entrega: 18 de octubre de 2017

3. Introducción

3.1. Archivos CSV

> Los archivos CSV (del inglés comma-separated values) son un tipo de documento en formato abierto sencillo para representar datos en forma de tabla, en las que las columnas se separan por comas y las filas por saltos de línea. Fuente: Wikipedia/CSV

Cada fila de un archivo CSV se denomina *registro*. Cada columna, es decir, el texto sepado por comas, se denomina *campo*. Un CSV está compuesto por uno o más registros que a su vez están compuestos por uno o más campos. La cantidad de campos por registro es invariable, todos tienen la misma cantidad, aunque posean campos vacíos. Los espacios cuentan como dato, por lo que los campos vacíos se distinguen por haber dos comas consecutivas. Un archivo CSV puede contener líneas con comentarios, que comienzan con un carácter determinado.

Así como existe el CSV, existen genéricamente los DSV, que vienen del inglés *valores separados por delimitadores* (*delimiter-separated values*). En los mismos, el carácter separador no es necesariamente la coma ni debe ser el mismo que el carácter que marca los comentarios. Un ejemplo de un DSV se ve en la figura 1, con # como carácter de marca de comentarios.

```
# Id, Nombre, Desarrollador, Plataforma, Release, Puntaje, Reseñas
12658,Persona 5,Atlus,PS4,2017-04-04,94,73
68536,Zelda: Breath of the Wild,Nintendo,Switch,2017-03-03,97,10
68537,Zelda: Breath of the Wild,Nintendo,Wii U,2017-03-03,96,13
```

Figura 1: Ejemplo de archivo DSV (en particular, CSV)

3.2. Altas/Bajas/Modificaciones

ABM (en inglés *CRUD*) es el acrónimo que se utiliza para referirse a las operaciones básicas de las bases de datos. Estas operaciones son: *crear* (altas), *leer*, *actualizar* (modificaciones), *borrar* (bajas). En el presente trabajo práctico se debe crear una aplicación que manipule un archivo que contiene registros de datos (en binario), realizando las operaciones mencionadas. Este archivo con registros es una base de datos, rudimentaria.

4. Desarrollo del TP

En el presente trabajo se debe desarrollar una aplicación que gestione una base de datos con puntajes. Para ello, se contará con archivos binarios con la información de la base de datos. En ésta se almacenan estructuras que contienen la información provista en la tabla 1.

La aplicación sólo debe contemplar el uso de un único tipo de estructuras, a elección del grupo de desarrolladores/as.

Juegos	Discos	Películas	Libros	Dato
id	id	id	id	size_t
nombre	nombre	titulo	titulo	char[200]
desarrolador	banda	guion	autor	char[200]
plataforma	genero	director	genero	char[200]
fecha	fecha	fecha	fecha	time_t
puntaje	puntaje	puntaje	puntaje	double
reseñas	reseñas	reseñas	reseñas	size_t

Tabla 1: Estructuras según información del puntaje

Para ello, será necesario primero crear una base de datos. Por otro lado, para la visualización de la misma, será necesario desarrollar otra aplicación. Estas 2 aplicaciones, y la aplicación principal, se detallan en las secciones 4.1, 4.2 y 4.3.

Nota Se puede asumir, en todas las aplicaciones, que los archivos se encuentran ordenados según el campo id.

Nota Los archivos pueden tener comentarios en cualquier línea. De haber un comentario, el carácter delimitador del mismo se encuentra siempre al comienzo de la línea.

4.1. Creación de la base de datos

Para la creación de la base de datos se debe desarrollar una aplicación distinta a la anterior. La misma debe ser invocable por línea de comandos del siguiente modo

```
./crear_base entrada salida
```

donde:

entrada es un archivo csv, como el de la figura 2, que se debe procesar para generar las estructuras.

salida es un archivo binario que contiene todos los datos del csv pero en estructuras.

Se puede asumir que el CSV se encuentra ordenado según el id y el mismo no presenta errores de formato.

Para la creación de la base de datos, se recomienda proceder del siguiente modo:

```
Mientras pueda leer lineas del archivo:
    1. extraer (separar) campos de la línea
    2. ¿Se pudo procesar la línea? SI/NO
    3. cargar los datos en la estructura
    4. ¿Se pudo cargar los datos? SI/NO
    5. guardar la estructura en el archivo
    6. ¿Se pudo guardar la estructura? SI/NO
```

```
# Id, Nombre, Desarrollador, Plataforma, Fecha, Puntaje, Reseñas
1,NetHack,The NetHack DevTeam,PC,1987-07-28,95,1530
180,Myst,Cyan Inc.,PC,1993-09-24,80,724
236,Team Fortress Classic,Valve Corporation,PC,1999-04-07,91,865
863,Diablo II,Blizzard North,PC,2000-06-29,99,2652
1530,Uplink,Introversion Software,PC,2001-10-01,87,1203
5398,Mirror's Edge,EA DICE,PC,2008-11-11,89,974
12658,Persona 5,Atlus,PS4,2017-04-04,94,730
68536,Zelda: Breath of the Wild,Nintendo,Switch,2017-03-03,97,1001
68537,Zelda: Breath of the Wild,Nintendo,Wii U,2017-03-03,96,1101
```

Figura 2: Ejemplo de archivo CSV para base de datos

Pensar qué hacer en caso de que falle alguna etapa.

4.2. Desbinarización de la base de datos

Se debe crear una nueva aplicación que genere una salida en formato CSV, sobre el flujo de salida estándar, a partir del contenido de la base de datos. El mismo debe ser invocable según el siguiente ejemplo.

```
./deco_base entrada
```

donde:

entrada es un archivo binario compuesto por estructuras.

Se puede asumir que el archivo de entrada se encuentra ordenado según el id y el mismo no está corrupto.

4.3. Gestión de la base de datos

La aplicación desarrollada debe ser invocable por línea de comandos de acuerdo con el siguiente ejemplo

```
./puntaje operacion -if db -f data -log log
```

donde:

operacion indica si se quiere realizar una alta, baja o modificación. La misma puede ser [A]LTA, [B]AJA o [M]ODIFICACION.

-if indica el archivo que contiene la base de datos, cuyo nombre es `db` –en el ejemplo–, es decir, debe recibir el nombre por CLA.

-f indica el archivo que contiene los registros a modificar.

-log indica un archivo donde almacenar los mensajes de error.

4.3.1. Altas

En el proceso de ALTAS, se debe modificar la base de datos original agregando los registros nuevos según la información del archivo de datos. Registros duplicados conforman un error y se debe almacenar únicamente el original.

Ejemplo En todos los casos, se muestran representaciones en CSV de los archivos binarios correspondientes.

db (base de datos)
```
1,NetHack,The NetHack DevTeam,PC,1987-07-28,95,1530
236,Team Fortress Classic,Valve Corpo,PC,1999-04-07,91,865
```

data (registros a dar de alta)
```
180,Myst,Cyan Inc.,PC,1993-09-24,80,724
863,Diablo II,Blizzard North,PC,2000-06-29,99,2652
```

db final (base de datos modificada)
```
1,NetHack,The NetHack DevTeam,PC,1987-07-28,95,1530
180,Myst,Cyan Inc.,PC,1993-09-24,80,724
236,Team Fortress Classic,Valve Corpo,PC,1999-04-07,91,865
863,Diablo II,Blizzard North,PC,2000-06-29,99,2652
```

4.3.2. Bajas

En el proceso de BAJAS, se debe modificar la base de datos original eliminando de la ella los registros que se encuentran en el archivo de datos. La inexistencia en la base de datos de un registro presente en el archivo de bajas, es un error y se debe informar.

Ejemplo En todos los casos, se muestran representaciones en CSV de los archivos binarios correspondientes.

db (base de datos)
```
1,NetHack,The NetHack DevTeam,PC,1987-07-28,95,1530
180,Myst,Cyan Inc.,PC,1993-09-24,80,724
236,Team Fortress Classic,Valve Corpo,PC,1999-04-07,91,865
863,Diablo II,Blizzard North,PC,2000-06-29,99,2652
```

data (registros a dar de baja)
```
180,Myst,Cyan Inc.,PC,1993-09-24,80,724
863,Diablo II,Blizzard North,PC,2000-06-29,99,2652
```

db final (base de datos modificada)
```
1,NetHack,The NetHack DevTeam,PC,1987-07-28,95,1530
236,Team Fortress Classic,Valve Corpo,PC,1999-04-07,91,865
```

4.3.3. Modificaciones

En el proceso de MODIFICACIONES, se deben cargar las estructuras del archivo de datos y con ellas modificar el contenido de las estructuras almacenadas en la base.

Ejemplo

db (base de datos)
```
1,NetHack,The NetHack DevTeam,PC,1987-07-28,95,1530
180,Myst,Cyan Inc.,PC,1993-09-24,80,724
236,Team Fortress Classic,Valve Corpo,PC,1999-04-07,91,865
863,Diablo II,Blizzard North,PC,2000-06-29,99,2652
```

data (registros a modificar)

```
863,Diablo II,Blizzard North,PC,2000-06-29,97,2943
```

db final (base de datos modificada)

```
1,NetHack,The NetHack DevTeam,PC,1987-07-28,95,1530
180,Myst,Cyan Inc.,PC,1993-09-24,80,724
236,Team Fortress Classic,Valve Corpo,PC,1999-04-07,91,865
863,Diablo II,Blizzard North,PC,2000-06-29,97,2943
```

4.3.4. Consideraciones

- En todos los casos, se puede asumir que todos los archivos de entrada se encuentran ordenados según el campo `id`.
- Si, durante la ejecución del proceso de ALTAs, se encuentra un `id` repetido, es un caso de *logueo*, pero no de interrupción del programa.
- Si, durante la ejecución del proceso de BAJAs, no se encuentra un `id`, es un caso de *logueo*, pero no de interrupción del programa.
- Si, durante la ejecución del proceso de MODIFICACIONES, no se encuentra un `id`, es un caso de *logueo*, pero no de interrupción del programa.

5. Restricciones

La realización de los programas pedidos está sujeta a las siguientes restricciones:

- Debe realizarse en grupos de **3 (tres)** integrantes.
- No está permitida la utilización de `scanf()`, `gets()`[1], `fflush(stdin)`[2],la biblioteca `conio.h`[3] (`#include <conio.h>`), etc.
- Debe recurrirse a la utilización de funciones mediante una adecuada parametrización.
- No está permitido en absoluto tener hard-codings:

```c
...
char c;
...
if (c == 'R')                    /* ¡¡hard-coded!!*/
    printf("%s","xxxxxxxx");     /* ¡¡hard-coded!!*/
else if(c == 'A')                /* ¡¡hard-coded!!*/
    printf("%s","yyyyyyyy");     /* ¡¡hard-coded!!*/
...
```

[1] obsoleta en C99 [3], eliminada en C11 [4] por fallas de seguridad en su uso.
[2] comportamiento indefinido para flujos de entrada ([3],[4]). Definida en estándar POSIX.
[3] biblioteca no estándar, con diferentes implementaciones y licencias, y no siempre disponible.

sino que debe recurrirse al uso de `ETIQUETAS`, `CONSTANTES SIMBÓLICAS`, `MACROS`, etc.

Los ejemplos no son exhaustivos, sino que existen otros hard-codings y tampoco son aceptados.

- Hay ciertas cuestiones que no han sido especificadas intencionalmente en este Requerimiento, para darle al/la desarrollador/a la libertad de elegir implementaciones que, según su criterio, resulten más convenientes en determinadas situaciones. Por lo tanto, se debe explicitar cada una de las decisiones adoptadas, y el o los fundamentos considerados para las mismas.

6. Entrega del Trabajo Práctico

La fecha de entrega del trabajo práctico es: 18 de octubre de 2017 o antes.

No se requiere entrega en papel, pero se acepta. Deberá realizarse una entrega digital, a través del campus de la materia, de un único archivo cuyo nombre debe seguir el siguiente formato:

```
YYYYMMDD_apellido1-apellido2-apellido3-N.tar.gz
```

donde `YYYY` es el año (2017), `MM` el mes y `DD` el día en que uno de los integrantes sube el archivo, `apellido-1a3` son los apellidos de los integrantes ordenados alfabéticamente, `N` indica el número de vez que se envía el trabajo (`1`, `2`, etc.), y `.tar.gz` es la extensión, que no necesariamente es `.tar.gz`.

El archivo comprimido debe contener los siguientes elementos:

- La correspondiente documentación de desarrollo del TP (en formato pdf), siguiendo la numeración siguiente, incluyendo:

  1. Carátula del TP. Incluir una dirección de correo electrónico.
  2. Enunciado del TP.
  3. Estructura funcional de los programas desarrollados.
  4. Explicación de cada una de las alternativas consideradas y las estrategias adoptadas.
  5. Resultados de la ejecución (corridas) de los programas, captura de las pantallas, bajo condiciones normales e inesperadas de entrada.
  6. Reseña sobre los problemas encontrados en el desarrollo de los programas y las soluciones implementadas para subsanarlos.
  7. Bibliografía (ver aparte).
  8. Indicaciones sobre la compilación de lo entregado para generar la aplicación.
     NOTA: Si la compilación del código fuente presenta mensajes de aviso (warning), notas o errores, los mismos deben ser comentados en un apartado del informe.

NOTA: El Informe deberá ser redactado en *correcto* idioma castellano.

- Códigos fuentes en formato de texto plano (.c y .h), *debidamente documentados*.

NOTA: Todos los integrantes del grupo deben subir el *mismo* archivo.
NOTA: Se debe generar y subir un único archivo (comprimido) con todos los elementos de la entrega digital. **NO usar RAR.** La compresión RAR no es un formato libre, en tanto sí se puede utilizar *ZIP*, *GUNZIP*, u otros (soportados, por ejemplo, por la aplicación de archivo *TAR*).

Si no se presenta cada uno de estos ítems, será rechazado el TP.

7. Bibliografía

Debe incluirse la referencia a toda bibliografía consultada para la realización del presente TP: libros, artículos, URLs, etc., citando:

- Denominación completa del material (Título, Autores, Edición, Volumen, etc.).
- Código ISBN del libro (opcional: código interbibliotecario).
- URL del sitio consultado. No poner Wikipedia.org o stackexchange.com, sino que debe incluírse un enlace al artículo, hilo, etc. consultado.

Utilizando LaTeX, la inclusión de citas/referencias es trivial. Los editores de texto gráficos de las suites de ofimática, como LibreOffice Write o MS Word, admiten plugins que facilitan la inclusión.

Ejemplo de referencias

[1] B.W. Kernighan y D.M. Ritchie. *The C Programming Language*. 2.ª ed. Prentice-Hall software series. Prentice Hall, 1988. ISBN: 9780131103627.

[2] P. Deitel y H. Deitel. *C How to Program*. 7.ª ed. Pearson Education, 2012. ISBN: 9780133061567.

[3] ISO/IEC. *Programming Languages – C*. ISO/IEC 9899:1999(E). ANSI, dic. de 1999, págs. 270-271.

[4] ISO/IEC. *Programming Languages – C*. INCITS/ISO/IEC 9899:2011. INCITS/ISO/IEC, 2012, pág. 305.