Metodología de la Programación
Grado en Informática
Guión de Prácticas Nº 2

DISEÑO MODULAR

Objetivos

- Aprender a resolver problemas descomponiéndolos en diferentes módulos (ficheros) y realizar compilación separada.
- Aprender a realizar una descomposición adecuada del problema en distintos módulos y que cada uno de ellos sea lo más independiente posible.
- Aprender a documentar programas.
- Realizar el desarrollo de un programa trabajando en equipo de manera organizada.

En las prácticas de este tema se formarán equipos de trabajo de cuatro personas, dónde uno de los componentes actuará como representante y se encargará de coordinar el grupo. Deben hacer una descomposición adecuada del problema en distintos módulos y un reparto de dichos módulos entre los distintos componentes del equipo, obteniendo así la solución al problema que se les plantea.

En todos los problemas los alumnos deben elaborar la documentación completa relativa a la aplicación desarrollada: documentar y describir las estructuras de datos utilizadas, redactar guías de uso de la aplicación,…

Implementación y uso de módulos en C

En el lenguaje C el código de un programa se puede escribir completo en un solo fichero o bien, cuando se trate de un programa grande, repartirlo entre varios ficheros. Es decir, en C un **fichero** que contiene una parte de un programa se corresponde con el concepto de **módulo**.

Los ficheros en que se divide un programa pueden ser de dos tipos:

- **Ficheros de código** (con extensión **.c**): Pueden contener declaraciones y/o definiciones de constantes, tipos, variables y funciones.
- **Ficheros de cabecera** (con extensión **.h**, del inglés *header* = cabecera): Pueden contener también cualquier parte de un programa, pero no conviene incluir en ellos nada que genere código al ser compilados. Es decir, pueden contener definiciones de tipos y constantes o declaraciones de variables y funciones (prototipos), pero no deben incluir definiciones de variables o funciones.

Para implementar en C un módulo de un algoritmo escrito en pseudocódigo haremos lo siguiente:

1. Creamos un fichero **`NombreModulo.h`** con las **declaraciones de todos los elementos públicos** o exportables, es decir, todos los elementos de la sección de exportación del módulo.

   Hay que aclarar que en C no es posible exportar constantes y tipos de datos ocultando su implementación. Por tanto, debemos incluir también en este fichero las **definiciones de constantes y tipos** que se quieran hacer **públicos**.

2. Creamos otro fichero **`NombreModulo.c`** con la **definición o implementación de todas las funciones**, tanto públicas como privadas.

   Si no se indica al compilador lo contrario, todas las funciones de un fichero son públicas, es decir, se pueden utilizar en cualquiera de los ficheros que componen un programa. Para hacer que una **función** sea **privada**, o sea, de uso exclusivo dentro del módulo en que está definida, su definición debe ir precedida de la palabra clave **`static`**.

   Puesto que las constantes y tipos públicos están definidos en el fichero de cabecera (.h) y tienen que ser utilizados en el fichero de código (.c), debemos incluir estas definiciones en este segundo fichero. Para ello debemos añadir la directiva **`#include "NombreModulo.h"`** al principio del fichero de código (`NombreModulo.c`).

Para hacer **uso de un módulo** en un fichero del programa (es decir, cuando un fichero importe algunos de los elementos exportados por otro módulo) debemos indicar al compilador dónde se encuentran las declaraciones de los elementos públicos del módulo que deseamos utilizar. Esto se hace también utilizando la directiva **`#include "NombreModulo.h"`** en dicho fichero.

Para **compilar un programa** compuesto de varios módulos y obtener el programa ejecutable debemos indicar al *enlazador* (*linker*) cuáles son estos módulos. Para ello, creamos un **proyecto** en el que debemos **incluir los ficheros de código (.c) correspondientes a los módulo utilizados**. En caso de que utilicemos módulos precompilados debemos incluir en el proyecto los ficheros de código objeto (.o) correspondientes.

Ficheros de texto

En ésta y en sucesivas prácticas se deben emplear ficheros de texto para leer y escribir datos. Daremos una breve explicación de cómo hacer uso de esta clase de ficheros en C.

Un fichero de texto consta de una secuencia de caracteres ASCII que finaliza con el **carácter de fin de fichero** (representado por la macro **`EOF`**) y se organiza en líneas terminadas por un carácter de **fin de línea (`'\n'`)**.

El fichero de cabecera `stdio.h` contiene las definiciones de constantes y tipos y las declaraciones de las funciones necesarias para manejar ficheros. Por tanto, cualquier módulo de un programa que haga uso de ficheros debe incluir esta cabecera estándar mediante la directiva **`#include <stdio.h>`**.

Declaración de una variable de tipo fichero:

```
FILE *var_fichero;
```

Apertura de un fichero:

Antes de poder leer o escribir datos en un fichero hay que abrirlo mediante la función

FILE *fopen (const char *nombre_fichero, const char *modo);

donde *nombre_fichero* es un puntero a una cadena de caracteres que contiene el nombre del fichero (puede incluir el nombre del directorio) y *modo* es otro puntero a una cadena de caracteres que indica cómo se debe abrir el fichero.

Modo	
r	Abre un fichero de texto para lectura
w	Crea un fichero de texto para escritura
a	Abre un fichero de texto para añadir
r+	Abre un fichero de texto para lectura/escritura
w+	Crea un fichero de texto para lectura/escritura
a+	Abre para añadir o crea un fichero de texto para lectura/escritura

La función *fopen()* devuelve un puntero al fichero abierto. Ese puntero es nulo (`NULL`) si por alguna razón se produce un error al intentar abrirlo.

Ejemplo:

```
FILE *fich;
if ((fich = fopen(“prueba.txt”, “w”)) == NULL) {
   printf(“No se puede abrir el fichero.\n”);
   exit(1);
}
```

Cierre de un fichero:

Una vez que un fichero deja de ser necesario en el programa hay que cerrarlo mediante la función

int fclose (FILE *f);

que devuelve 0 si la operación se ha llevado a cabo con éxito. Esta función hace que se escriba toda la información que todavía se encuentre en el buffer del disco. Para evitar posibles pérdidas de datos es muy importante cerrar todos los ficheros abiertos antes de que finalice la ejecución del programa.

Lectura:

int fgetc (FILE *f);

int getc (FILE *f);

Son dos funciones idénticas que devuelven el carácter del fichero *f* situado en la posición actual y avanzan el indicador de posición del fichero al siguiente carácter. Devuelven `EOF` si se ha llegado al final del fichero.

char *fgets (char *s, int tam, FILE *f);

Esta función lee caracteres y los copia en la cadena apuntada por *s* hasta llegar a un carácter de fin de línea (`'\n'`), un `EOF` o hasta leer *tam-1* caracteres. Después pone un carácter nulo (`'\0'`) al final de la cadena. También devuelve un puntero a la cadena leída.

int fscanf (FILE *f, const char *formato, ...);

Funciona exactamente igual que *scanf()* excepto que lee los datos del fichero *f* en lugar de hacerlo de *stdin* (entrada estándar, normalmente el teclado).

Escritura:

int fputc (int c, FILE *f);

int putc (int c, FILE *f);

Ambas funciones escriben el carácter *c* en el fichero *f* y avanzan el indicador de posición. Devuelven `EOF` si se produce un error y si no, el carácter escrito.

int fputs (const char *s, FILE *f);

Escribe la cadena de caracteres apuntada por *s* en el fichero *f*. El carácter nulo de terminación (`'\0'`) no se escribe. Devuelve `EOF` si se produce un error y un valor negativo en caso contrario.

int fprintf (FILE *f, const char *formato, ...);

Funciona igual que *printf()*, pero escribiendo los datos en el fichero *f* en vez de escribirlos en *stdout* (salida estándar, normalmente la pantalla).

Problema

LIGA FANTASTICA MP

En esta práctica se va a crear un pequeño programa de simulación de fútbol donde todos los datos van a estar almacenados en ficheros, garantizando la conservación de la información y la posibilidad de volverla a utilizar en posteriores ejecuciones del programa.

La idea de este programa es la creación de una liga de fútbol virtual en la que cada **participante** configure sus propias plantillas de fútbol con los futbolistas disponibles de equipos reales de primera división. Al final de cada jornada, un **cronista** valorará a cada uno de los futbolistas en función de cómo hayan jugado en la vida real. De manera que, cada futbolista tendrá asociada una puntuación y cada plantilla acumulará a su puntuación general la suma de todas las valoraciones de cada uno los futbolistas que forman dicha plantilla. Así, el ganador de la liga será el participante al que pertenezca el equipo con mayor puntuación acumulada.

El programa dispondrá de varios perfiles de usuario:

- Un perfil de usuario **participante**: tendrá la posibilidad de configurar plantillas de fútbol con los diferentes futbolistas disponibles.
- Un perfil de usuario **cronista:** se encargará de establecer la valoración (0-10) de los futbolistas en cada jornada.
- Un perfil **administrador**: podrá realizar tareas de configuración del programa como asignar el presupuesto por defecto de los participantes, el nº máximo de plantillas a crear por cada participante, etc.

Se trabajará con los ficheros de texto que se listan a continuación, el contenido de cada uno de ellos se cargará inicialmente en memoria al ejecutar el programa. Estos datos serán modificados a lo largo de la ejecución, por lo que, al salir del programa, se deberá volcar de nuevo los datos actualizados a los correspondientes ficheros:

- Fichero **Equipos.txt**, almacenará la información de los equipos reales de la primera división con los siguientes campos separados por guiones:
  - Identificador del equipo con dos dígitos
  - Nombre del equipo con 20 caracteres máximo

  Ej: 01-Real Madrid
  02-Barcelona
  …….

- Fichero **Futbolistas.txt**, almacenará la información relativa a los futbolistas de los equipos de primera división con los siguientes campos separados por guiones:
  - Identificador del futbolista con dos dígitos

- Identificador del equipo al que pertenece con dos dígitos (debe coincidir con un identificador de equipo existente en el fichero **Equipos.txt**)
- Nombre del jugador con 20 caracteres máximo
- Precio del jugador en millones de €
- Valoración actual del jugador por parte del **cronista**. Este valor (0-10) se establecerá a cero inicialmente cuando se cree al futbolista. En cada jornada un usuario **cronista** podrá modificar dichos valores en función de cómo considere que cada futbolista haya jugado en la realidad.

Ej: 01-01-Iker Casillas-35-0
02-01-Cristiano Ronaldo-90-0
03-02-Sergio Busquets-25-0
04-02-Lionel Messi-90-0
…….

- Fichero **Usuarios.txt**, almacenará la información de los usuarios del sistema con los siguientes campos separados por guiones:
  - Identificador del usuario con dos dígitos
  - Nombre completo del usuario con 20 dígitos máximo
  - Perfil del usuario que será "administrador", "participante" o "cronista"
  - Usuario para acceder al sistema con 5 caracteres
  - Contraseña para acceder al sistema con 8 caracteres.

Ej: 01-Juan Pérez-administrador-admin-jp123456
02-Pedro López-cronista-croni-pl124312
03-Guillermo Gómez-participante-part1-gg125431
04-Manuel López-participante-part2-ml909876
……..…….

Para facilitar el trabajo, este fichero se creará inicialmente con al menos un usuario **administrador** y un usuario **cronista.**

- Fichero **Plantillas.txt**, almacenará la relación de plantillas configuradas por cada participante con los siguientes campos separados por guiones:
  - Identificador del propietario de la plantilla con 2 dígitos (debe coincidir con un identificador válido de un usuario participante en el fichero **Usuarios.txt**)
  - Identificador de la plantilla con tres dígitos
  - Nombre de la plantilla con 30 caracteres máximo
  - Presupuesto disponible del participante, en millones de €, que será asignado por defecto al registrar la plantilla con el valor configurado por el administrador en el fichero **configuración.txt** y que irá modificándose automáticamente después de que el participante realice cualquier alta o baja de futbolistas en dicha plantilla.
  - Puntuación acumulada de la plantilla. Este valor será establecido por defecto a 0 al crear inicialmente la plantilla.

Ej: 03-001-Real Garete-80-0
03-002-Futbol Club La Isla-80-0
04-003-Club Atlético Malacatón-80-0

……..…

- Fichero **Jugadores_Plantillas.txt**, almacenará la relación de jugadores asignados a cada una de las plantillas del fichero anterior con los siguientes campos separados por guiones:
    - Identificador del jugador con 2 dígitos (debe coincidir con un identificador válido de un futbolista existente en el fichero **futbolistas.txt**)
    - Identificador de la plantilla con tres dígitos (debe coincidir con un identificador válido de una plantilla existente en el fichero **plantillas.txt**).

    Ej: 01-001
    02-001
    01-002
    ……..…

- Fichero **Configuracion.txt**, almacenará los datos relativos a la configuración general del programa con los siguientes campos separados por guiones:
    - Campo de configuración que indica qué se va a almacenar, con 30 caracteres máximo
    - Valor que toma el campo anterior

    Ej: nº_maximo_equipos-3
    Presupuesto_defecto-80
    Nº_maximo_jugadores_equipo-11
    ……..

Inicialmente aparecerá un menú principal para acceso a la LFMP si el usuario está registrado o bien para registrarse en el sistema:

Ej:
MENU LFMP
1.- Registro
2.- Acceso al sistema

1. **Registro**
   En caso de seleccionar la opción **1**, se permitirá realizar un registro nuevo en el sistema como usuario **participante**. Se pedirán los datos correspondientes al nuevo usuario que quedarán guardados con el perfil **participante**.

2. **Acceso al sistema**
   Cuando se selecciona la opción **2** se deberá pedir un identificador de usuario y contraseña de acceso al sistema. Los datos introducidos se deberán contrastar con los datos previamente cargados en memoria, y que corresponden al contenido del fichero **usuarios.txt.** Si el usuario es válido, habrá que comprobar qué tipo de perfil tiene asociado:

1. Perfil **Participante**
   En caso de que el usuario sea un **participante** debe aparecer un menú:

MENU PARTICIPANTE
1.- Crear Plantilla
2.- Configurar Plantilla
3.- Listar Plantillas
4.- Eliminar Plantilla
5.- Ranking
6.- Salir del programa

1. Crear plantilla. Debe dar la posibilidad de introducir los datos de la nueva plantilla. Tener en cuenta siempre que se debe respetar el valor de configuración del número máximo de plantillas por participante.

2. Configurar Plantilla. Mostrará la lista de las plantillas actuales de las que el participante es propietario para que pueda elegir cuál configurar. Una vez seleccionada la plantilla, se mostrará en pantalla el presupuesto disponible y el siguiente menú:

1.- Lista de jugadores en plantilla
2.- Lista de jugadores disponibles
3.- Añadir jugador a plantilla
4.- Eliminar jugador de plantilla
5.- Volver

1.- Opción que visualiza la lista de jugadores que forman parte de la plantilla actual junto con su valoración.
2.- Visualiza la lista de futbolistas de primera división junto con su precio.
3.- Opción que permite al participante **añadir** el futbolista deseado a la plantilla actual, por ejemplo, introduciendo su identificador. Una vez seleccionado un futbolista, automáticamente quedará asignado a su plantilla y el presupuesto del participante se verá reducido en la cantidad indicada por el precio del jugador seleccionado. Sólo se podrá seleccionar aquel jugador que no pertenezca ya a la plantilla y cuyo precio sea inferior o igual al presupuesto con el que cuenta el participante para dicha plantilla. Siempre se debe respetar el nº máximo de jugadores en plantilla atendiendo a la configuración general.
4.- El participante podrá **eliminar** el jugador deseado de la plantilla actual, por ejemplo, introduciendo su identificador. El presupuesto del participante para la plantilla actual se verá incrementado en la cantidad indicada por el precio del jugador eliminado.
5.- Volver al menú anterior.

3. Listar Plantillas. Mostrará la lista de las plantillas actuales de las que el participante es propietario, junto con su puntuación acumulada.

4. Eliminar Plantilla. El participante podrá eliminar la plantilla deseada previa selección de la misma.
5. Ranking. Mostrará el listado de las tres plantillas con mayor puntuación acumulada.
6. Salir del programa. Permitirá salir del programa.

2. Perfil **Cronista**

En caso de que el usuario sea un **cronista** debe aparecer un menú:

MENU CRONISTA
1.- Listar Equipos
2.- Valorar Equipos
3.- Salir del programa

1. Listar Equipos. Visualiza la lista de Equipos de primera división.

2. Valorar Equipos. Permitirá al cronista actualizar las valoraciones de los futbolistas. Para ello, el cronista seleccionará un identificador de un equipo y posteriormente le aparecerá la lista de futbolistas de dicho equipo junto con su valoración actual. A continuación, el cronista irá seleccionando futbolistas e introduciendo sus nuevas valoraciones (0-10).

3. Salir del programa. Permitirá salir del programa. En el caso de que el cronista haya realizado alguna valoración se deberán actualizar todas las puntuaciones de todas las plantillas existentes en el sistema acumulando las valoraciones realizadas.

4. Perfil **Administrador**

Para el caso en el que el usuario sea un **administrador** debe aparecer un menú:

MENU ADMINISTRADOR
1.- Equipos
2.- Usuarios
3.- Configuración
4.- Salir del programa

1. Equipos. Debe mostrar un menú al administrador para listar, modificar, añadir y eliminar equipos de primera división al igual que sus futbolistas.

2. Usuarios. Debe mostrar un menú al administrador para listar, modificar, añadir y eliminar usuarios del sistema.

3. Configuración. Debe permitir editar los valores de configuración general del programa.

4. Salir del programa. Permitirá salir del programa.

Nota.- Como se ha comentado anteriormente, al iniciar el programa se realizará la carga inicial en memoria de la información existente en los ficheros. Todas las operaciones que se realicen durante la ejecución del programa se deben realizar sobre los datos en memoria, y una vez seleccionada la opción de *Salir del programa* habrá que volcar la información actual a los ficheros para conservar los cambios. Esta opción se mostrará y realizará esta operación para todos los perfiles de usuarios.