Metodología de la Programación
Grado en Informática
Guión de Prácticas Nº 1

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 los 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

CUADERNO DIGITAL DEL PROFESOR (CDP)

El objetivo de la práctica es realizar la implementación de una versión simplificada de un cuaderno del profesor, en formato digital, que intente sustituir al tradicional cuaderno, en papel, del profesor de secundaria. Se pretende que actúe como único cuaderno para todo el profesorado y no exista un ejemplar individual por cada profesor del centro, como ocurre con el cuaderno en papel.
En este cuaderno se llevará el control académico de todo el alumnado, como por ejemplo las calificaciones, faltas de asistencia, etc., además de sus datos personales. También se usará para almacenar las materias y grupos que imparte el profesorado, así como los horarios, etc.

El programa dispondrá de dos perfiles de usuarios:

- Un perfil de **profesor**, que accederá a los datos personales de los alumnos a los que imparte clases, calificaciones, faltas de asistencia, anotaciones, partes de incidencia, etc.
- Un perfil **administrador**, que realizará tareas de configuración del programa tales como tratamiento de alumnos, usuarios, materias, matrículas de alumnos en materias, horarios, etc.

Para conservar toda la información y volverla a utilizar en posteriores ejecuciones del programa, todos los datos del Cuaderno Digital del Profesor (CDP) estarán almacenados en ficheros. De esta forma, al iniciar el CDP, es necesario que dicha información se vuelque a las estructuras de datos correspondientes en memoria, y al cerrar el programa, se vuelvan a almacenar todos los datos actualizados en los ficheros, realizándose así el funcionamiento del CDP en memoria principal.

Los ficheros que van contener dicha información son:

- **Usuarios.txt**, almacenará la información de los usuarios del sistema con los siguientes campos separados por guiones:
  - Identificador del usuario (Id_usuario), tres dígitos
  - Nombre completo del usuario (Nomb_usuario), 20 carácteres máximo
  - Perfil del usuario (Perfil_usuario): “administrador” o “profesor”
  - Nombre de usuario (Usuario) para acceder al sistema, 5 caracteres
  - Contraseña para acceder al sistema (Contraseña), 8 caracteres

  Ejemplo:

  001-Juan Pérez-administrador-admin-jp123456
  002-Guillermo Gómez-profesor-prof1-gg125431
  003-Manuel López- profesor –prof2-ml909876
  …….

- **Alumnos.txt**, almacenará la información de los alumnos del centro con los siguientes campos separados por guiones:

    - Identificador escolar (Id_alum), seis dígitos
    - Nombre del alumno (Nombre_alum), 20 carácteres máximo
    - Dirección del alumno (Direc_alum), 30 carácteres máximo
    - Localidad del alumno (Local_alum), 30 caracteres máximo
    - Curso al que pertenece (Curso), 30 caracteres máximo
    - Grupo al que pertenece, (Grupo), 10 caracteres máximo

  Ejemplo:

123456-Dolores Martín-C/diego montes nº 4-Cádiz-1ºBachillerato Ciencias Sociales-1ºHCSA
342312-Pedro Lima-Avda Colón S/N-Puerto Santa María- 4ºESO-4ºB
…….

- **Materias.txt**, almacenará la información relativa a las asignaturas impartidas en el centro educativo con los siguientes campos separados por guiones:
    - Identificador de la materia (Id_materia), 4 dígitos
    - Nombre de la materia (Nombre_materia), 50 carácteres máximo
    - Abreviatura del nombre de la materia (Abrev_materia), 3 carácteres

  Ejemplo:

  0001-Matemáticas Aplicadas a las Ciencias Sociales-MCS
  0002-Historia del Arte-HAR
  …….

- **Matriculas.txt**, almacenará la información relativa a las asignaturas en las que se encuentran matriculados los alumnos con los siguientes campos separados por guiones:
    - Identificador de la materia (Id_materia), 4 dígitos (debe coincidir con el Id_materia de alguna materia del fichero Materias.txt)
    - Identificador escolar (Id_alum), seis dígitos (debe coincidir con el Id_alum de algún alumno del fichero Alumnos.txt)

  Ejemplo:

  0001-342312
  0002-123456
  …….

- **Calificaciones.txt**, almacenará la información de todas las calificaciones que reciben los alumnos por parte de los profesores con los siguientes campos separados por guiones:
    - Fecha a la que corresponde la calificación (Fecha_calif), en formato fecha
    - Descripción de la calificación (Descrip_calif), 30 caracteres máximo

- Identificador de la materia (Id_materia) a la que corresponde la calificación , 4 dígitos (debe coincidir con el Id_materia de alguna materia del fichero Materias.txt)
- Identificador escolar del alumno (Id_alum) que recibe la calificación, seis dígitos (debe coincidir con el Id_alum de algún alumno del fichero Alumnos.txt)
- Calificación (Valor_calif), valor numérico de la calificación obtenida entre 0 y 10

Ejemplo:

25/02/2017-Examen unidad 1-0001-342312-7
25/02/2017-Examen unidad 4-0002-123456-4
……..…….

- **Faltas.txt**, almacenará la información de todas las faltas de asistencia que tienen los alumnos con los siguientes campos separados por guiones:
  - Fecha a la que corresponde la falta de asistencia (Fecha_falta), en formato fecha
  - Hora (Hora_falta), valor numérico de 1 a 6 para reflejar el tramo horario en el que ha faltado
  - Descripción de la falta (Descrip_falta), 30 caracteres máximo
  - Estado de la falta (Estado_falta), indica en qué estado se encuentra la falta: Injustificada, Justificada, Retraso
  - Identificador escolar del alumno (Id_alum) al que corresponde la falta de asistencia, seis dígitos (debe coincidir con el Id_alum de algún alumno del fichero Alumnos.txt)

Ejemplo:

25/02/2017-4-enfermedad-injustificada-342312
22/02/2017-2-actividad extraescolar-justificada-123456
……..…….

- **Horarios.txt**, almacenará la información de las materias que imparte cada profesor con los siguientes campos separados por guiones:
  - Identificador del profesor (Id_profesor), tres dígitos (debe coincidir con un identificador de usuario, Id_usuario, con perfil profesor del fichero Usuarios.txt)
  - Día (Día_clase), valor numérico de 1 a 5 que indica el día de la semana que imparte clase de la materia el profesor
  - Hora (Hora_clase), valor numérico de 1 a 6 que indica el tramo horario en el que imparte clase de la materia el profesor
  - Identificador de la materia (Id_materia), 4 dígitos (debe coincidir con el Id_materia de alguna materia del fichero Materias.txt)
  - Grupo (Grupo) al que imparte clase, 10 caracteres máximo (debe coincidir con algún grupo de los que aparecen en el fichero Alumnos.txt)

Ejemplo:

003-1-2-0002-1ºHCSA (El profesor Manuel López imparte clases de Historia del Arte al grupo 1ºHCSA los lunes a 2ª hora)

002-5-3-0001-4ºB (El profesor Guillermo Gómez imparte clases de Matemáticas aplicadas a las ciencias sociales al grupo 4º B los viernes a 3ª hora)

……..…….

La ejecución del programa comienza con un mensaje inicial solicitando el usuario y contraseña para acceder al CDP. Los datos introducidos se deberán contrastar con los datos previamente cargados en memoria (prodentes de **Usuarios.txt**). Si se corresponde con un usuario válido hay que comprobar el perfil:

- Si corresponde al perfil **Profesor**:
  Aparecerá una lista con todos los grupos y materias a los que imparte clases en el día actual con el objetivo de que el profesor seleccione con qué grupo quiere trabajar en ese momento. A continuación aparecerá el siguiente menú en pantalla, suponiendo ese ejemplo de selección de grupo y materia:

  GRUPO 1ºHCSA MATERIA HAR
  1.- Lista de alumnos
  2.- Cambiar de grupo

  1.- La primera opción mostrará la lista de todos los alumnos del grupo seleccionado que están matriculados en la materia correspondiente, pudiendo seleccionar alguno de ellos. Una vez hecha la selección se permitirán las siguientes acciones sobre el mismo:

  ALUMNO: ----------
  1. Ficha del alumno.
  2. Calificaciones del alumno
  3. Faltas de asistencia.
  4. Volver.

  1.- Ficha del alumno
  Esta opción mostrará la ficha con los datos personales del alumno y una opción que permita al profesor la edición para modificar esos datos.
  2.- Calificaciones del alumno
  Esta opción mostrará la lista de las calificaciones que tiene actualmente el alumno en esa materia, y un menú de opciones para modificar una calificación de la lista, borrarla o añadir una nueva al alumno actual.
  3.- Faltas de asistencia
  Esta opción mostrará la lista de las faltas de asistencia que tiene actualmente el alumno en esa materia, y un menú de opciones para modificar una falta de la lista, borrarla o añadir una nueva al alumno actual.
  4.- Volver
  Esta opción permitirá al usuario volver al menú anterior.

2.- La segunda opción permitirá al profesor cambiar de grupo y materia mostrando de nuevo la lista inicial de selección.

- Si corresponde al perfil **Administrador**:
  En el caso de que sea un usuario administrador, el menú que aparecerá en pantalla será el siguiente:

  1.- Usuarios
  2.- Alumnos
  3.- Materias
  4.- Horarios

  1.- Usuarios
  Permitirá al administrador gestionar los usuarios del sistema pudiendo dar de alta, baja, modificar y listar usuarios.
  2.- Alumnos
  Permitirá al administrador gestionar los alumnos del sistema pudiendo dar de alta, baja, modificar y listar alumnos. Además, para un alumno seleccionado, se permitirá mostrar la lista de materias en las que se encuentra matriculado, realizar cambios de matrícula a otras materias, eliminar matrícula en alguna materia y crear nuevas matrículas.
  3.- Materias
  Permitirá al usuario administrador gestionar todas las materias impartidas en el centro pudiendo dar de alta, baja, modificar y listar materias.
  4.- Horarios
  Permitirá al administrador gestionar todos los horarios de profesores pudiendo añadir horas de clase a un profesor concreto, eliminarlas, modificarlas y listar horarios de cada profesor.