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. Crearemos 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. Crearemos 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 reservada **`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

ESI-SHARE

ESI-SHARE es un sistema que facilita la compartición de coches entre los miembros de la comunidad universitaria de la Escuela Superior de Ingeniería, con el objetivo de reducir los costes diarios de desplazamiento hacia nuestra escuela. Este sistema se basa en los archiconocidos como BlaBlaCar, etc. Con esta práctica se pretende implementar una versión simplificada de dichos sistemas adaptada a nuestras necesidades.

El sistema dispondrá de dos perfiles de usuarios:

- Un perfil de **usuario,** que tendrá la posibilidad de acceder a los datos de su perfil, realizar publicaciones de viajes, gestión de vehículos compartidos, etc.
- Un perfil de **administrador,** que podrá realizar tareas de configuración del sistema como tratamiento de usuarios, trayectos, etc.

Inicialmente, todos los datos de ESI-SHARE estarán almacenados en ficheros, lo que permite la conservación de la información en el tiempo, así como su reutilización en posteriores ejecuciones del programa. Al iniciar el sistema ESI-SHARE dicha información se volcará en memoria a las estructuras de datos correspondientes, con las que se trabajará hasta la finalización de su ejecución, cuya información actualizada se almacenará de nuevo en los ficheros.

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), con 4 dígitos
  - Nombre completo del usuario (Nomb_usuario), con 20 caracteres máximo
  - Población (Localidad), con 20 caracteres máximo, para indicar la ubicación desde la que sale el usuario con su vehículo compartido
  - Perfil del usuario(Perfil_usuario): Administrador o Usuario
  - Usuario (User) para acceder al sistema, con 5 caracteres
  - Contraseña (Login) para acceder al sistema, con 8 caracteres
  - Estado: Activo o Bloqueado

  0001-Juan Pérez-San Fernando-administrador-admin-jp123456-activo
  0002-Guillermo Gómez-Cádiz-usuario-usua1-gg125431-activo
  0003-Manuel López-Chiclana–usua2-ml909876-activo

- **Vehículos.txt**, almacenará la información de los vehículos que el usuario haya dado de alta en el sistema y que utilizará para compartir en sus viajes:
  - Matrícula del vehículo (Id_mat), con 7 caracteres
  - Identificador del usuario (Id_usuario) propietario del vehículo, con 4 dígitos
  - Nº de Plazas (Num_plazas), con 1 dígito, para indicar el número de plazas de las que dispone el coche, sin contar la plaza del conductor.
  - Descripción del vehículo (Desc_veh), con 50 caracteres, para indicar los datos del vehículo como marca, modelo, color, etc..

2321GJK-0002-4-Citroen MX Rojo
1234HLD-0003-5-Peugeot 806 Verde
7886JJP-0003-4-Seat Arona Blanco

- **Viajes.txt,**almacenará la información de los viajes publicados por los usuarios del sistema, con los siguientes campos separados por guiones:
    - Identificador del viaje (Id_viaje), con 6 dígitos
    - Matrícula del vehículo (Id_mat) que se comparte, con 7 caracteres
    - Fecha del viaje (F_inic), con formato día/mes/año
    - Hora de inicio (H_inic), con formato hora:minutos, usando el sistema horario de 24 horas
    - Hora de llegada (H_fin), con formato hora:minutos, usando el sistema horario de 24 horas
    - Plazas libres (Plazas_libre), número de plazas que aún quedan sin ocupar, con 1 dígito
    - Viaje: Ida o Vuelta. Para simplificar, cada viaje puede ser únicamente de ida o vuelta, pero no ambos.
    - Importe total del viaje
    - Estado del viaje (Estado), para indicar si el viaje está:
        - Abierto, con posibilidad de añadir usuarios al viaje
        - Cerrado, indicando que ya no quedan plazas disponibles
        - Iniciado, indicando que ya ha comenzado el viaje pero que permite añadir más usuarios en el caso de haber plazas libre
        - Finalizado, indicando que ya ha finalizado el viaje
        - Anulado, indicando que el viaje no se ha realizado

    000001-2321GJK-05/03/2018-10:30-11:30-4-ida-5€-abierto
    000002-1234HLD-06/03/2018-08:00-09:00-5-ida-4€-abierto
    000003-7886JJP-04/03/2018_08:30-09:30-4-vuelta-3€-cerrado

- **Pasos.txt**, almacenará la información de las poblaciones por las que, durante un viaje, está dispuesto a pasar un usuario, con su vehículo compartido, con objeto de recoger a otros usuarios. Contendrá los siguientes campos separados por guiones:

    - Identificador del viaje (Id_viaje), con 6 dígitos
    - Población de paso (Población), con 20 caracteres máximo

        000001-San Fernando
        000001-Puerto Real
        000002-Puerto Real

- **Incidencias.txt**, almacenará la información relativa a las incidencias que hayan podido ocurrir durante cualquiera de los trayectos. Después de que se haya dado por finalizado un viaje, cada usuario que participa de ese viaje tiene la posibilidad de abrir una incidencia.

    - Identificador del viaje (Id_viaje), con 6 dígitos.

- Identificador del usuario (Id_us_registra) que registra la incidencia, con 4 dígitos
- Identificador del usuario (Id_us_incidencia) sobre el que recae la incidencia
- Descripción de la incidencia (Desc_indicencia) con 100 caracteres
- Estado de la incidencia (Est_incidencia):
  - Abierta, indicará que la incidencia aún no ha sido vista por el administrador
  - Validada, indicará que la incidencia ha sido validada por el administrador
  - Cerrada, indicará que la incidencia ha sido cerrada por el administrador

000001-0004-0001-No ha pasado a recogerme-Abierta
000001-0002-0005-Se niega a pagar el trayecto-Cerrada

La ejecución del programa comienza con un mensaje inicial solicitando el usuario y contraseña para acceder al ESI-SHARE. Los datos introducidos se deberán contrastar con los datos previamente cargados en memoria (procedentes de **Usuarios.txt**). Si los datos introducidos no se encuentran en el sistema, éste debe dar la posibilidad de registrarse con el perfil de usuario, dando acceso directo al sistema. Si los datos introducidos sí se corresponden con un usuario válido hay que comprobar el perfil:

1. Si corresponde al perfil **Usuario**:
   Aparecerá una pantalla con los datos del usuario y las correspondientes opciones:

   NOMBRE Y APELLIDOS DEL USUARIO

   1.- Perfil
   2.-Vehículos
   3.- Viajes
   4.- Incidencias

   1. Perfil. Esta primera opción mostrará los datos del perfil del usuario con posibilidad de edición, por si deseara realizar alguna modificación.
   2. Vehículos. Permitirá al usuario acceder a la lista de vehículos que tiene dados de alta, así como la posibilidad de dar de alta nuevos vehículos, eliminar o modificar alguno de ellos. Lógicamente, habrá usuarios que no tengan ningún vehículo asociado, son aquellos que hacen uso del sistema para viajar con acceso a vehículos compartidos.
   3. Viajes. Aparecerá por defecto la lista de los viajes que haya abiertos en ese momento, siendo posible editarlos si el usuario es el mismo que lo ha publicado. Esta modificación sólo será posible si no hay ningún usuario que se haya incorporado al viaje, de ser así, no será factible ningún cambio. También debe dar la posibilidad de publicar nuevos viajes.
      - Para publicar un nuevo viaje el sistema debe controlar que el usuario tenga algún vehículo dado de alta, en caso contrario no se permitirá el alta. De entre los vehículos dados de alta, el sistema dará la opción de elegir el vehículo con el que va a realizar el viaje. El número de plazas libres quedará establecido por defecto al número de plazas de las que dispone el vehículo y el estado se establecerá a abierto. El usuario deberá introducir la fecha y horas

de inicio y llegada, así como el importe total del viaje. El sistema controlará que la fecha introducida no haya transcurrido ya, y que la horas de inicio y fin sean correctas.

- Si el usuario quisiera incorporarse a un viaje, se le dará la opción de que el sistema le muestre únicamente los viajes que tengan previsto pasar por, o se inicien, o terminen en su población junto con el importe que tendrá que pagar (que se calculará en función del importe establecido por el propietario y el número actual de usuarios añadidos al viaje). La decisión de mostrar el resto de datos del viaje se dejará a criterio del programador. Lógicamente, desde el mismo momento que un usuario se añade a un viaje, se debe actualizar el nº de plazas libres de dicho viaje y el estado en caso de quedar completado. En todo momento, un usuario que se haya añadido a un viaje podrá anular la decisión siempre que el viaje no se haya iniciado.
- Otra opción que debe permitir el sistema es poder visualizar detalladamente cada viaje, mostrando la lista de poblaciones por las que pasará durante su trayecto.
- Un viaje quedará finalizado de forma automática pasada una hora después de la hora prevista de llegada. Sería conveniente que al arrancar, el sistema hiciera las comprobaciones pertinentes y las correspondientes actualizaciones.

4. Incidencias. Para velar por el buen funcionamiento de ESI-SHARE, existe la posibilidad de que cualquiera de los usuarios pueda poner una incidencia referente a algún viaje y usuario concreto. La lista de incidencias podrá ser consultada por cualquier usuario en cualquier momento. Cuando se pulse esta opción aparecerá por defecto la lista de incidencias que afecten directamente al usuario en cuestión. No se podrán realizar incidencias sobre un viaje finalizado. O sea, que las incidencias son recomendables realizarlas en la hora siguiente a la hora prevista de llegada.

2. Si corresponde al perfil **Administrador**:
   Aparecerá en pantalla el siguiente menú:

   1.- Usuarios
   2.- Vehículos
   3.- Viajes
   4.- Incidencias

1. Usuarios. Permitirá al administrador gestionar los usuarios del sistema pudiendo dar de alta, baja, modificar y listar usuarios. Al listar los usuarios se permitirá ver el número de incidencias que recaen sobre los mismos. El administrador será el único que podrá bloquear manualmente a un usuario debido a las incidencias registradas. No se tendrán en cuenta las incidencias ya cerradas.
2. Vehículos. Permitirá al administrador gestionar todos los vehículos del sistema pudiendo dar de alta, baja, modificar y listar vehículos. Se dará la

posibilidad de, dado un vehículo concreto, mostrar la lista de todos los viajes realizados.

3. Viajes. Permitirá al administrador gestionar todos los viajes. Podrá publicar, eliminar, modificar y listar viajes.
4. Incidencias. Permitirá al administrador gestionar todas las incidencias del sistema pudiendo crear, eliminar, modificar y listar las mismas. El sistema le dará la opción de validar las incidencias, una vez comprobada su veracidad. El administrador debe bloquear manualmente a un usuario con tres incidencias validadas.