Práctica 2: El laboratorio del IMC

Programación 2

Curso 2024-2025

Esta práctica del curso consiste en el desarrollo de un sistema de gestión de analíticas. **Para resolver esta práctica puedes usar los elementos de C++ vistos en las transparencias del *Tema 1*, *Tema 2* y *Tema 3* de teoría.**

Condiciones de entrega

- La fecha límite de entrega para esta práctica es el **viernes 28 de marzo**, hasta las **23:59**
- Debes entregar un único fichero llamado `prac2.cc` con el código de todas las funciones

Código de honor

Si se detecta copia (total o parcial) en tu práctica, tendrás un **0** en la entrega y se informará a la dirección de la Escuela Politécnica Superior para que adopte medidas disciplinarias

Está bien discutir con tus compañeros posibles soluciones a las prácticas
Está bien apuntarte a una academia si sirve para obligarte a estudiar y hacer las prácticas

Está mal copiar código de otros compañeros o pedirle a ChatGPT que te haga la práctica
Está mal apuntarte a una academia para que te hagan las prácticas

Si necesitas ayuda acude a tu profesor/a
No copies

Normas generales

- Debes entregar la práctica exclusivamente a través del servidor de prácticas del Departamento de Lenguajes y Sistemas Informáticos (DLSI). Se puede acceder a él de dos maneras:
  - Página principal del DLSI (`https://www.dlsi.ua.es`), opción "ENTREGA DE PRÁCTICAS"
  - Directamente en la dirección `https://pracdlsi.dlsi.ua.es`
- Cuestiones que debes tener en cuenta al hacer la entrega:
  - El usuario y la contraseña para entregar prácticas son los mismos que utilizas en UACloud
  - Puedes entregar la práctica varias veces, pero sólo se corregirá la última entrega
  - No se admitirán entregas por otros medios, como el correo electrónico o UACloud
  - No se admitirán entregas fuera de plazo
- Tu práctica debe poder ser compilada sin errores con el compilador de C++ existente en la distribución de Linux de los laboratorios de prácticas

- Si tu práctica no se puede compilar su calificación será 0
- Al comienzo de todos los ficheros fuente entregados debes incluir un comentario con tu NIF (o equivalente) y tu nombre. Por ejemplo:

prac2.cc

```
// DNI 12345678X GARCIA GARCIA, JUAN MANUEL
...
```

- Superar **todas** las pruebas del autocorrector te da derecho a presentarte al examen de prácticas. Si falla alguna prueba no podrás presentarte al examen y tu calificación en esta práctica será 0
- El cálculo de la nota de la práctica y su relevancia en la nota final de la asignatura se detallan en las transparencias de presentación de la asignatura (*Tema 0*)

1 Descripción de la práctica

Se desea desarrollar un sistema de gestión de los informes de analíticas (en este caso de índice de masa corporal) de un laboratorio. En el sistema se darán de alta los pacientes por un lado y los resultados de cada uno de los análisis que se realicen. El sistema permitirá hacer un seguimiento de los datos introducidos y detectará situaciones de riesgo.

2 Detalles de implementación

En el Moodle de la asignatura se publicarán varios ficheros que necesitarás para la correcta realización de la práctica:

- `prac2.cc`. Este fichero contiene un esqueleto de programa sobre el que realizar tu práctica. Descárgalo y añade tu código en él. Este fichero contiene la siguiente información:
  - Los registros (struct) necesarios para hacer la práctica
  - Un tipo enumerado `Error` que contiene todas las posibles clases de error que se pueden dar en esta práctica (por ejemplo, `ERR_OPTION`)
  - Una función `error` que se encarga de mostrar el correspondiente mensaje de error por pantalla en función del parámetro que se le pase. Por ejemplo, cuando la función recibe el parámetro `ERR_OPTION`, mostrará por pantalla el mensaje `ERROR: wrong option`
  - Una función `showMenu` que muestra por pantalla el menú del programa
  - Una función `main` que implementa la gestión del menú principal y llama a las funciones correspondientes dependiendo de la opción elegida por el usuario
- `autocorrector-prac2.tgz`. Contiene los ficheros del autocorrector para evaluar la práctica con distintas pruebas de entrada. La corrección automática de la práctica, tras la entrega, se realizará con estas mismas pruebas y será necesario superarlas todas para poder presentarte al examen de esta práctica
- `prac2`. Fichero ejecutable de la práctica (compilado para máquinas Linux de 64 bits) desarrollado por el profesorado de la asignatura, para que puedas probarlo con las entradas que quieras y ver la salida correcta esperada

3 Funcionamiento del programa

El programa que vas a desarrollar consiste en un sistema de almacenamiento y gestión de las analíticas que se realizan en un laboratorio.

4 Componentes

El programa a desarrollar debe poder gestionar tres elementos fundamentales: los pacientes, las diferentes analíticas que se realizan a cada uno de los pacientes y la base de datos, que es donde se almacena toda la información de los pacientes y de las analíticas. Los siguientes apartados describen la estructura de cada uno de estos componentes en más detalle.

4.1 Paciente

Las estructuras de tipo `Patient` almacenan la información relativa a un paciente del laboratorio. Cada paciente estará identificado por su NIF (`nif`), su nombre (`name`) y su teléfono (`telephone`):

```
struct Patient{
    string nif;
    string name;
    string telephone;
};
```

También utilizaremos una estructura paciente especifica para almacenar los pacientes en el fichero binario (`PatientBin`):

```
struct PatientBin{
    char nif[KMAXNIF];
    char name[KMAXNAME];
    char telephone[KMAXTELEPHONE];
};
```

Las constantes `KMAXINF`, `KMAXNAME` y `KMAXTEPEPHONE` tienen el formato que se indica a continuación:

```
const int KMAXNIF=10;
const int KMAXNAME=50;
const int KMAXTELEPHONE=14;
```

4.2 Analítica

Las estructuras de tipo `Analysis` almacenan información referente a los datos de un análisis realizado a un paciente. Cada analítica estará identificada por un número y contendrá un identificador único (`id`), el NIF del paciente (`nif`), la fecha de la analítica (`dateAnalysis`), el peso (`weight`) y altura (`height`):

```
struct Analysis{
    unsigned int id;
    char nif[KMAXNIF];
    Date dateAnalysis;
    float weight;
    float height;
};
```

El registro `Date` se define de la siguiente manera:

```
struct Date{
    unsigned int day;
    unsigned int month;
    unsigned int year;
};
```

4.3 Database

La estructura `Database` almacena toda la colección de pacientes (`patients`) y analíticas realizadas (`analysis`). Contiene además un campo `nextId` que almacenará el identificador que deberá asignarse al campo `id` del siguiente análisis que se cree. Este campo tendrá valor 1 al inicio del programa y se incrementará de uno en uno para cada nueva analítica. Es decir, la primera analítica que se cree tendrá como identificador 1, la segunda tendrá el 2 y así sucesivamente.

```
struct Database{
    unsigned int nextId;
    vector <Patient> patients;
    vector <Analysis> analysis;
};
```

5 Inicio del programa

El `main` de tu práctica contendrá una variable de tipo `Database` que almacenará en memoria toda la información de los pacientes y analíticas durante la ejecución del programa. Cuando se inicie el programa, el campo `nextId` de esta variable se inicializará a 1.

Lo primero que deberás hacer es cargar en memoria los datos de los pacientes almacenados en el fichero binario `patients.bin`. Este fichero contiene estructuras de tipo `PatientBin`, una detrás de otra, de manera que en cada una se almacena la información de un paciente. Deberás crear una función `loadPatients` que se encargue de leer toda la información de los pacientes y cargarla en el vector `patients` de la variable de tipo `Database` mencionada anteriormente.

- Tu programa sólo gestionará una única variable de tipo `Database` que contendrá todos los datos de pacientes y análisis
- Si al iniciar el programa el fichero `patients.bin` no existe, no debes mostrar ningún tipo de error. El programa iniciará con normalidad dejando el vector `patients` de `Database` vacío
- Asumiremos que el contenido del fichero `patients.bin` es correcto. No hay que hacer ningún tipo de comprobación sobre el contenido durante la carga

6 Menú

Al ejecutar la práctica, y después de cargar la información de los pacientes (si los hay), se mostrará por pantalla el menú principal del programa, quedando a la espera de que el usuario elija una opción:

Terminal

```
1- Add patient
2- View patient
3- Delete patient
4- Save patients
5- Add analysis
6- Export analysis
7- Import analysis
8- Statistics
q- Quit
Option:
```

Las opciones válidas que puede introducir el usuario son los números del `1` al `8` y la letra `q`. Si la opción elegida no es ninguna de éstas (por ejemplo un `9`, una `x` o un `;`), se emitirá el error `ERR_OPTION` llamando a la función `error` con dicho parámetro. Cuando el usuario elige una opción correcta, se debe ejecutar el código asociado a dicha opción. Al finalizarla, volverá a mostrar el menú principal y a pedir otra opción, hasta que el usuario decida salir del programa utilizando la opción `q`.

7 Opciones

En los siguientes apartados se describe el funcionamiento que deberá tener cada una de las opciones que se muestran en el menú principal del programa.

7.1 Add patient

Esta opción permite dar de alta los pacientes de los que podremos introducir análisis en la aplicación. Su código se deberá incluir en una función llamada `addPatient`. La información que se solicitará en primer lugar es el NIF del paciente a dar de alta:

```
Terminal
Enter NIF:
```

Si se introduce un NIF en blanco se deberá volver al menú principal. Se deberá garantizar que se introducen los nueve caracteres del NIF, compuesto de ocho números y una letra. En caso contrario se mostrará el error `ERR_WRONG_NIF` y lo volverá a solicitar.

Se comprobará con la función `searchPatient` si éste existe. Esta función devolverá -1 si el paciente no existe. En caso contrario, devolverá la posición del paciente dentro del vector `patients`. Si ya existiese un paciente con dicho NIF, lanzará el mensaje de error `ERR_PATIENT_EXISTS` y el programa volvería a solicitar el NIF. Si no existe, pedirá el resto de información y añadirá un nuevo registro de paciente. En primer lugar, pedirá el nombre:

```
Terminal
Enter name:
```

Se debe comprobar que en el campo nombre se introducen al menos tres caracteres. En caso contrario se se mostrará el mensaje `ERR_WRONG_NAME` y se volvería a solicitar. A continuación se solicitará el teléfono:

```
Terminal
Enter telephone:
```

Se debe comprobar que el número telefónico empieza por el símbolo + (para el prefijo telefónico) y va seguido por entre 10 y 12 números. Si no fuese así, se mostraría el mensaje de error `ERR_WRONG_TELEPHONE` y se volvería a pedir el número de teléfono.

- Para saber si un carácter es un número, puedes usar la función `isdigit()` de la librería `cctype`
- Para saber si un carácter es una letra, puedes usar la función `isalpha()` de la librearía `cctype`

7.2 View patient

Esta opción permite visualizar la información de un paciente. Se deberá implementar una función llamada `viewPatient`. Solicitará en primer lugar el NIF con el mensaje:

```
Terminal
Enter NIF:
```

En caso de no tener un paciente con dicho NIF (se deberá utilizar la función `searchPatient` para localizarlo), emitirá el mensaje de error `ERR_PATIENT_NOT_EXISTS` y se volverá a solicitar el NIF. Si se introduce la cadena vacía, se volverá al menú principal sin hacer nada.

Si el paciente existe en la base de datos, se visualizarán sus datos y los resultados de todas las analíticas que se le hayan realizado, una en cada línea, siguiendo este formato:

Terminal

```
NIF: 11111111D
Name: Pedro Pi
Telephone: +34612345689
Id  Date        Height  Weight
23  11/11/2027  188     95
45  17/02/2028  188     93
```

Las columnas `Id`, `Date`, `Height` y `Weight` están separadas por tabuladores (carácter `'\t'`). Si no hay análisis para ese paciente, se mostrarán solo sus datos, sin poner `Id`, `Date`, etc.

7.3 Delete patient

Esta opción permite borrar la información de un paciente. Se deberá implementar una función llamada `deletePatient`. En primer lugar solicitará el NIF con el siguiente mensaje:

Terminal

```
Enter NIF:
```

Si se introduce un NIF en blanco se deberá volver al menú principal. En el caso de no tener información de un paciente con dicho NIF lanzará el error `ERR_PATIENT_NOT_EXISTS` y se volverá a solicitar el NIF.

En el caso de que el paciente exista se borrará del vector `patients` de la base de datos y se eliminarán todas las analíticas que de dicho paciente se tengan almacenadas en el vector `analysis`.

7.4 Save patients

Esta opción guarda todos los pacientes de la base de datos en el fichero `patients.bin`. Si ya existiese un fichero con dicho nombre lo sobreescribirá con las información actual de la base de datos. Se deberá implementar una función `savePatients` para ello.

Para guardar la información de una paciente (`Patient`) en el fichero binario, se deberá primero transformar en una estructura de tipo (`PatientBin`). Esto implica que tal vez tengas que recortar el nombre del paciente para que no supere el tamaño máximo `KMAXNAME`.

7.5 Add analysis

Esta opción permite añadir resultados de un análisis para un paciente. Se implementará en una función `addAnalysis`. En primer lugar solicitará el NIF del paciente:

Terminal

```
Enter NIF:
```

Si se introduce un NIF en blanco se deberá volver al menú principal. Si no existiese un paciente con dicho NIF, emitiría el error `ERR_PATIENT_NOT_EXISTS` y se volvería a solicitar de nuevo. Si existe, solicitará la fecha de la analítica con el siguiente mensaje:

Terminal

```
Enter date (day/month/year):
```

El usuario deberá introducir la fecha con el formato indicado en el mensaje. Por ejemplo: `12/5/2027`. Asumiremos que el usuario siempre introduce el formato correcto. No hace falta comprobarlo.

Lo que sí se deberá comprobar es que la fecha es válida. Para ello, el día deberá estar comprendido entre los valores 1 y 31, el mes entre 1 y 2, y el años entre 2025 y 2050, inclusive. No es necesario comprobar

que el número de días es correcto para un mes concreto. Asumismo que todos los meses pueden tener 31 días (por ejemplo, el 30 de febrero sería correcto). Esta información deberá almacenarse en los campos `day`, `month` y `year` de `dateAnalysis`. En caso de error mostrará el mensaje `ERR_WRONG_DATE` y se pedirá de nuevo que se introduzca la fecha. A continuación se pedirá el peso (en kilogramos) con el siguiente mensaje:

Terminal
```
Enter weight:
```

Se deberá verificar que el campo peso sea positivo. En caso contrario se lanzará el error `ERR_WRONG_NUMBER` y se volverán a solicitar. Finalmente, se pedirá la altura (en centímetros):

Terminal
```
Enter height:
```

Al igual que antes, se deberá verificar que el campo altura sea positivo. En caso contrario se lanzará el error `ERR_WRONG_NUMBER` y se volverá a solicitar.

Por último, se asignará al campo `id` el valor de `nextId` de la base de datos y se incrementará este valor para la siguiente analítica.

7.6 Export analysis

Almacena en un archivo que se llama `analysis.bin` todos los registros de analíticas almacenados en la base de datos. Se deberá implementar una función `exportAnalysis` para ello. En el caso de que exista el fichero, se deberá sobreescribir.

7.7 Import analysis

Leerá la información del fichero `analysis.bin`. Se deberá implementar en una función `importAnalysis`. Si no existiese el fichero se deberá lanzar el mensaje de error `ERR_FILE_NOT_EXISTS` y se volverá a la función principal sin hacer nada más.

Para cada una de las analíticas almacenadas en el fichero, se deberá comprobar previamente que existe un paciente con dicho NIF en el vector `patients` de la base de datos. En caso contrario, escribirá el NIF de dicho paciente en el fichero de texto `wrong_patients.txt` (uno en cada línea) y no se cargará dicha analítica en el sistema. Este fichero no se sobreescribirá en caso de que existiese, sino que se añadirá información a continuación. Si el fichero `wrong_patients.txt` no se puede abrir, se mostrará también el error `ERR_FILE_NOT_EXISTS` y se volverá al menú principal sin hacer nada más.

Para cada analítica que se cargue en el sistema se le deberá asignar el `id` que le corresponda como si se estuvieran insertando desde la opción `Add analysis` de analítica que corresponda según el valor almacenado en la variable `Database`. Es decir, se ignora el `id` que tuviera la analítica cuando se almacenó en el fichero binario. Las analíticas que pudieran existir antes de la carga de fichero no se borrarán.

7.8 Statistics

Esta opción recorrerá el fichero de analíticas y mostrará el grado de obesidad para cada una de las analíticas de los pacientes. Un paciente puede tener múltiples analíticas, así que se mostrarán todas ellas. Se implementará en una función llamada `statistics`.

Una medida de la obesidad se determina mediante el índice de masa corporal (IMC), que se calcula con la siguiente fórmula:

$$IMC = \frac{\text{peso (kg)}}{[\text{altura (m)}]^2}$$

Por ejemplo, una persona de 1,85 metros de altura y 94 kilogramos de peso, tendría un IMC de $94/1,85^2 = 27,5$. Según el Instituto Nacional del Corazón, los Pulmones y la Sangre de los Estados

Unidos (NHLBI), el sobrepeso se define como un IMC de 25 o más. Se considera que una persona es obesa si su IMC es superior o igual a 30. La siguiente tabla muestra los distintos tipos de composición corporal y su correspondiente IMC:

Composición corporal	Índice de masa corporal (IMC)
Underweight	Menos de 18.5
Healthy	18.5 – 24.9
Overweight	25.0 – 29.9
Obesity	Igual o mayor de 30.0

De cada analítica mostraremos el NIF del paciente, fecha de la analítica (en formato `dd/mm/yyyy`), peso, altura y composición corporal (`Underweight`, `Healthy`, `Overweight` u `Obesity`). Todos estos campos se mostrarán separados por un `';'`:

Terminal

```
11111111D;02/05/2025;110;180;Obesity
11111111D;04/07/2025;115;180;Obesity
22222222G;13/11/2025;75;186;Healthy
```

Solo se mostrará la información de los pacientes que al menos tengan una analítica. La misma información que se muestra en pantalla y con ese mismo formato (separando cada campo con `';'`), se guardará en un fichero de texto llamado `statistics.txt`. Si el fichero ya existiera, se sobreescribirá.

- Al mostrar la información por pantalla y guardarla en fichero, si el número que representa al mes o al año es inferior a 10, se deberá incluir un 0 delante para que el formato cumpla con dos dígitos para el día y dos para el mes

8 Argumentos del programa

En esta práctica el programa debe permitir al usuario indicar algunas acciones a realizar mediante el paso de argumentos por línea de comando. Deberás utilizar los parámetros `int argc` y `char *argv[]` de la función `main` para poder gestionarlos. El programa admitirá los siguientes argumentos por línea de comando:

- `-f <fichero texto>`. Permite importar analíticas de un fichero de texto, cuyo nombre deberá indicarse tras el argumento `-f`, iniciando el programa a continuación. El formato del fichero de texto deberá ser el mismo que el que se genera con la opción `Statistics` (no hace falta comprobarlo). Deberá ignorarse el IMC de la analítica leída de fichero y guardar todo el resto de información en registros de tipo `Analysis` en el vector `analysis` de la base de datos. Para cada analítica que se cargue en el sistema se le deberá asignar el `id` que le corresponda como si se estuvieran insertando desde la opción `Add analysis`. Al igual que sucedía en la opción `Import analysis` del menú, si un NIF no existe en la base de datos, se escribirá el NIF de dicho paciente en el fichero `wrong_patients.txt`.

  Si el fichero pasado como argumento no existe o el fichero `wrong_patients.txt` no se puede abrir, se emitirá el error `ERR_FILE_NOT_EXISTS` y se iniciará el programa con normalidad sin cargar ninguna analítica. Ten en cuenta que antes de gestionar el fichero de análisis el programa cargará en memoria la información de `patients.bin` para que existan pacientes a los que asociar los análisis.

  El siguiente ejemplo importará los datos almacenados en el fichero de texto `blood_tests.txt`:

  Terminal

  ```
  $ prac2 -f blood_tests.txt
  ```

- `-s`. Muestra las estadísticas, igual que si el usuario seleccionara la opción `Statistics` del menú, saliendo del programa a continuación sin llegar a mostrar el menú principal. Esta opción solo tiene sentido si el usuario ha introducido también el argumento `-f`, ya que si no no existiría ninguna información que mostrar. Por esta razón, la siguiente llamada sería incorrecta:

```
Terminal
$ prac2 -s
```

Los dos argumentos pueden aparecer al mismo tiempo en una llamada al programa y además en cualquier orden. Independientemente del orden en el que aparezcan en la línea de comando, el orden en el que se procesarán los argumentos será el siguiente:

1. En primer lugar se procesará la opción `-f` para cargar datos del fichero
2. En segundo lugar se ejecutará la opción `-s` para mostrar las estadísticas

En el siguiente ejemplo, en primer lugar se cargarán los datos del programa almacenados en el fichero `blood_tests.txt` y a continuación se mostrarán las estadísticas, saliendo a continuación del programa:

```
Terminal
$ prac2 -s -f blood_tests.txt
```

Otro ejemplo, cuyos parámetros son válidos, sería el siguiente:

```
Terminal
$ prac2 -f -f -s
```

Aunque a primera vista los parámetros pudieran parecer incorrectos, en realidad no lo son. El programa recibe un primer parámetro `-f` que va seguido de un fichero que se llama `-f`. A continuación recibe el parámetro `-s` para mostrar las estadísticas. Esta secuencia de parámetros es, por tanto, correcta.

En caso de que se produzca un error en los argumentos de entrada (por ejemplo, que se introduzca una opción que no existe o se meta una duplicada), se deberá emitir el error `ERR_ARGS` y se finalizará el programa sin llegar a mostrar el menú principal. Si todos los argumentos son correctos, se procesarán las opciones introducidas por el usuario y después se mostrará el menú principal de programa de la forma habitual (siempre que no se haya pasado la opción `-s`, que muestra la información y finaliza el programa).

- Recuerda, independientemente de si se pasan argumentos del programa o no, siempre se deberá cargar primero el fichero `patients.bin` al inicio del programa
- Los dos argumentos son opcionales y pueden no aparecer en la llamada al programa
- Recuerda, si está el parámetro `-s` obligatoriamente debe estar `-f`. Si apareciese sola, deberá de emitir el error `ERR_ARGS`
- Un determinado argumento sólo puede aparecer una vez en la llamada, de lo contrario se considerará que los argumentos son incorrectos
- Si aparece algún argumento diferente a los mencionados (por ejemplo, `-n`) se considerará que los argumentos son incorrectos
- Si detrás de `-f` no aparece el nombre del fichero, los argumentos serán incorrectos también
- Los ficheros de texto no tienen por qué tener necesariamente la extensión `.txt`, al igual que los ficheros binario no tienen por qué tener la extensión `.bin`. Utilizamos estas extensiones en los ejemplos para que quede más claro que estamos trabajando con ficheros de texto y binarios