Práctica 1
Dime qué día de la semana fue, es o será...

Fecha límite de entrega: 27 de noviembre de 2016

El día 1 de enero de 1900 fue lunes. Con este simple dato de referencia se puede averiguar el día de la semana asociado a cualquier fecha posterior. En esta práctica vas a desarrollar, de forma incremental, un programa que sea capaz de eso y de más cosas.

Versión 1 del programa.- En primer lugar debes desarrollar un programa que averigüe el día de la semana para una fecha proporcionada por el usuario a través de teclado. El programa debe verificar la validez de la fecha proporcionada, solicitándola tantas veces como sea preciso. Además tiene que seguir un algoritmo concreto, que es el siguiente: en primer lugar, calcular el número de días transcurridos desde el 1 de enero de 1900 hasta la fecha problema, y en segundo lugar, utilizar el hecho de que la fecha de referencia era lunes para saber a qué día de la semana desplazan esos días transcurridos.

Requisitos de implementación de la versión 1

Tu solución debe incluir la implementación de, al menos, los siguientes subprogramas:

- ✓ **`int solicitaAnyo()`**: solicita, valida y devuelve un año introducido por el usuario a través de teclado. Debe ser 1900 o posterior.
- ✓ **`int solicitaMes()`**: solicita, valida y devuelve un mes introducido por el usuario a través de teclado.
- ✓ **`int solicitaDia(int mes, int anyo)`**: solicita, valida y devuelve un día introducido por el usuario, correcto para el mes y el año proporcionados.
- ✓ **`long int diasTranscurridos (int dia, int mes, int anyo)`**: devuelve el número de días transcurridos desde el 1/enero/1900 hasta la fecha `dia/mes/anyo`.
- ✓ **`long int diasAnyosCompletos (int anyo)`**: devuelve el número de días que han transcurrido desde el 1/enero/1900 hasta el 1/enero/`anyo`.

- ✓ **`int diasEsteAnyo (int dia, int mes, int anyo`**): devuelve el número de días transcurridos desde la fecha 1/enero/`anyo` hasta la fecha `dia/mes/anyo`.
- ✓ **`int contarBisiestos (int anyoInicio, int anyoFinal)`**: devuelve el número de años bisiestos existentes en el intervalo [`anyoInicio, anyoFinal`].
- ✓ **`int diasMes (int mes, int anyo)`**: devuelve el número de días del mes y año dado, teniendo en cuenta en el caso de ser el mes de febrero si el año es bisiesto o no.
- ✓ **`bool esBisiesto (int anyo)`**: devuelve un valor booleano que indica si el año `anyo` es o no bisiesto. Según el calendario gregoriano un año es bisiesto si se cumple una de las dos siguientes condiciones:
  - Es múltiplo de 4 pero no es múltiplo de 100.
  - Es múltiplo de 400.
- ✓ **`int diaSemana (long int numDias)`**: devuelve una representación entera del día de la semana que es después de haber transcurrido `numDias` días desde el 1/enero/1900 (0 si es lunes, 1 si es martes,…, 6 si es domingo).
- ✓ **`string nombreDia (int representacionDia)`**: recibe un entero que representa un día de la semana (0 si es lunes, 1 si es martes,…, 6 si es domingo) y devuelve una cadena con el nombre del día.

Versión 2 del programa.- Ahora tienes que ampliar la funcionalidad del programa de manera que permita elegir entre las opciones que figuran a continuación y finalizar sólo cuando el usuario decida salir:

```
1 – Calcular el día de la semana para una fecha dada
2 – Obtener la fecha correspondiente al primer domingo de un año
3 – Obtener los domingos de un año
4 – Obtener los posibles puentes de un año
0 - Salir
```

La opción 1 se corresponde con la versión 1 del programa.

La opción 2 debe solicitar un año por teclado (que deber ser igual o posterior a 1900) y calcular la fecha correspondiente al primer domingo de ese año. Por ejemplo, para el año 2017, el programa deberá mostrar por pantalla:

El primer domingo del año 2017 es el día: 1 de enero

La opción 3 debe presentar por pantalla el día y el mes de todos los domingos de un año que el usuario introduce por teclado (igual o posterior a 1900). Por ejemplo, para el año 2017 el programa debe mostrar por pantalla

Domingos de 2017

1 de enero

8 de enero

15 de enero

...

19 de marzo

26 de marzo

...

5 de noviembre

12 de noviembre

...

24 de diciembre

31 de diciembre

Número total de domingos: 53

La opción 4 debe leer del archivo `fiestas.txt` los días festivos de un cierto año (igual o posterior a 1900) que no caen en domingo y localizar cuáles de estos días festivos dan lugar a "puentes". Se considera que un día festivo da lugar a un puente cuando cae en martes o en jueves. Los días festivos del archivo `fiestas.txt` que dan lugar a puente hay que escribirlos en el archivo `puentes.txt`.

El archivo `fiestas.txt` tiene una primera línea en la que figura el año y una línea por cada fecha que corresponde a un día festivo donde figuran dos números separados por blancos (el primero es el día y el segundo el mes). El archivo finaliza con un centinela que son dos 0 separados por blancos.

El archivo `puentes.txt` deberá contener una primera línea donde figure el año y una línea por cada día de `fiestas.txt` que da lugar a puente en la que aparezcan, en primer lugar la fecha en el mismo formato que se ha establecido para `fiestas.txt` y en segundo lugar, separado por un blanco, el día de la semana en que cae (martes o jueves). Finaliza con un centinela que son dos 0 separados por blancos y XXX como nombre del día.

Requisitos de implementación de la versión 2

Tu solución debe incluir la implementación de, al menos, los siguientes subprogramas:

- ✓ **`int menu()`**: muestra por pantalla un menú con las distintas opciones y solicita, valida y devuelve la opción elegida.
- ✓ **`string diaDeLaSemana(int dia, int mes, int anyo)`**: devuelve una cadena con el nombre del día de la semana en que cae la fecha `dia/mes/anyo`.
- ✓ **`int primerDomingoMes(int mes, int anyo)`**: devuelve el día del mes que corresponde al primer domingo del `mes/anyo` dados.
- ✓ **`int domingosAnyo(int anyo)`**: muestra por pantalla los domingos del año `anyo` y devuelve el número de domingos mostrados.
- ✓ **`bool puentes()`**: localiza los posibles puentes asociados a las fiestas de un año, guardadas en el archivo `fiestas.txt`, y guarda en el archivo `puentes.txt` aquellas fiestas que dan lugar a puente de acuerdo con el criterio y formato indicado anteriormente en el enunciado de la práctica.

Versión 3 del programa (<u>opcional</u>; cuenta en el apartado de "actividad adicional" del método de evaluación).- Finalmente, de manera opcional, puedes añadir una opción más en el menú:

`5 – Obtener los posibles puentes de un año "mejorado"`

Esta opción hace esencialmente lo mismo que la opción 4 pero debe mejorar el criterio de detección de posibles puentes. La opción 4 detecta un posible puente cuando la fiesta X cae en martes o en jueves pero no tiene en cuenta que el lunes inmediatamente anterior o el viernes inmediatamente posterior, respectivamente, podrían ser también fiesta y por tanto la fiesta X no generaría puente. Un ejemplo habitual en la Comunidad de Madrid es la fiesta de jueves santo, que no genera puente porque el viernes santo también es fiesta.

Entrega de la práctica

La práctica se entregará en el Campus Virtual por medio de la tarea **Entrega de la Práctica 1**, que permitirá subir el archivo `main.cpp` con el código fuente de la versión 2 (o de la 3, si haces la parte opcional). Uno de los dos miembros del grupo será el encargado de subirlo, no lo suben los dos.

Recordad poner el nombre de los miembros del grupo en un comentario al principio del archivo de código fuente.

Fecha límite de entrega: **27 de noviembre de 2016.**