Examen Práctico de Programación 1 - 6/septiembre/2016

- Tiempo para realizar el trabajo de programación propuesto: 2 horas
- Entrega del trabajo a través de la plataforma *Moodle2*.

Especificación del trabajo a desarrollar

Se ha de completar el desarrollo de un programa, en el área de trabajo **`trabajos`**, ubicada en la carpeta **`practicasPROG1`**.

Cada alumno debe diseñar únicamente dos funciones, escribiendo sus especificaciones y su código. El resto del código del programa se facilita y se puede copiar desde la web de la asignatura (fichero **`codigo.cc`** ubicado en la carpeta **`examenSeptiembre`**).

Las funciones a diseñar son las siguientes.

```
/*
 * Programador del módulo: .. aquí debe escribir su nombre y apellidos ...
 * Fecha de la última revisión : 6/septiembre/2016
 */

#include <iostream>
#include <iomanip>
#include <cstring>

using namespace std;

#include " ../ ModTiempo/tiempo.h"                  // el módulo tiempo
#include " ../ ModSubtitulo/ subtitulo .h"           // el módulo subtitulo
#include " ../ ModHerramientas/herramientas.h"       // el módulo herramientas

/*
 * Pre:  ... escribir aquí su especificación ...
 * Post: ... escribir aquí su especificación ...
 */
void reprogramar ( Subtitulo & S, const int mseg) {
     ... escribir aquí su código ...
}

/*
 * Pre:  ... escribir aquí su especificación ...
 * Post: ... escribir aquí su especificación ...
 */
void presentarEstadistica (const Subtitulo TS[], const int n) {
     ... escribir aquí su código ...
}

/*
 * Este programa realiza las siguiente tareas :
 * 1. Pide el operador que defina el nombre de un fichero de subtítulos con formato
 *    SubRip, que ha de estar ubicado en la carpeta RUTA, y lee su respuesta . Ej:
 *      Un fichero de subtitulos (de practicasPROG1/datos/ subtitulos ): s01. srt
 * 2. Presenta por pantalla una tabla de estadísticas de la duracción de los
 *     subtítulos del fichero
 * 3. Ilustra cómo se puede adelantar y retrasar los tiempos de inicio y fnalización
 *    de un subtítulo
 */
int main () {
```

```
    // Carpeta que almacena los ficheros de subtítulos
    const char RUTA[] = "../../../datos/subtitulos/";
    // Nombre de fichero (sin ruta) y con ruta
    char nombreFichero[64];
    char nombreCompletoFichero[128];
    // Pregunta por el nombre de un fichero de subtítulos
    cout << "Un fichero de subtitulos (en practicasPROG1/datos/subtitulos): " << flush;
    cin >> nombreFichero;
    strcpy(nombreCompletoFichero, RUTA);
    strcat(nombreCompletoFichero, nombreFichero);
    // Asigna a numSubtitulos el número de subtitulos del fichero nombreCompletoFichero
    int numSubtitulos = contarSubtitulos(nombreCompletoFichero);
    if (numSubtitulos > 0) {
        // Lee los subtitulos del fichero nombreCompletoFichero y los almacena en
        // TS[0,numSubtitulos-1]
        const int MAXIMO = 2000;      // Núm. máximo de subtitulos que pueden tratarse
        Subtitulo TS[MAXIMO];      // Tabla de subtítulos
        leerSubtitulos(nombreCompletoFichero, TS);
        // Presenta una estadística de las duraciones de los subtítulos del fichero
        presentarEstadistica(TS, numSubtitulos);
        // Define un número de subtítulo y lo muestra por pantalla
        int numSub = 754;
        mostrar(TS[numSub-1]); cout << endl;
        // Adelanta su tiempo de inicio y de finalización 1.555 segundos y
        // lo vuelve a mostra por pantalla
        reprogramar(TS[numSub-1], -1555);
        mostrar(TS[numSub-1]); cout << endl;
        // Retrasa su tiempo de inicio y de finalización 1.556 segundos y
        // lo vuelve a mostrar por pantalla
        reprogramar(TS[numSub-1], 1556);
        mostrar(TS[numSub-1]); cout << endl;
    }
    else {
        cout << "No se ha podido leer el fichero " << nombreFichero<< endl;
    }
    return 0;
}
```

El comportamiento de las dos funciones cuya especificcaión y cuyo código se han de escribir se detalla en los dos apartados que siguen.

Comportamiento de la función `reprogramar(S,mseg)`

Una invocación **`reprogramar`**`(S,mseg)` modifica el subtítulo **`S`** del siguiente modo:

- Si el valor de `mseg` es negativo entonces modifica los valores de los tiempos de inicio y finalización del subtítulo **`S`** adelantándolos `-mseg` milisegundos.
- Si el valor de `mseg` es positivo entonces modifica los valores de los tiempos de inicio y finalización del subtítulo **`S`** retrasándolos `mseg` milisegundos.
- Si el valor de `mseg` es cero entonces no modifica ninguno de los datos asociados al subtítulo **`S`**.

Si el subtítulo **`S`** almacena la siguiente información:

```
754 00:48:39,770 --> 00:48:42,480
He never had a conviction
except Charlie Kane in his life.
```

Tras ejecutar **reprogramar**(S,-1555) pasará a almacenar:

```
754 00:48:38,215 --> 00:48:40,925
He never had a conviction
except Charlie Kane in his life.
```

Y tras ejecutar de nuevo **reprogramar**(S,1556) pasará a almacenar:

```
754 00:48:39,771 --> 00:48:42,481
He never had a conviction
except Charlie Kane in his life.
```

Comportamiento de la función presentarEstadistica(TS,n)

Una invocación **presentarEstadistica**(TS,n) produce como resultado la presentación por pantalla de la siguiente estadística sobre las duraciones de los subtítulos almacenados en TS[0,n-1]:

```
 DURACION [seg.]   SUBTITULOS     PORCENTAJE
===============    ==========     ==========
 0.000 - 0.999           0           0.00%
 1.000 - 1.999         127          10.17%
 2.000 - 2.999         602          48.20%
 3.000 - 3.999         383          30.66%
 4.000 - 4.999         122           9.77%
 5.000 - 5.999          15           1.20%
                   ----------     ----------
                      1249         100.00%
```

La tabla anterior muestra una estadística de las duraciones de los 1249 subtítulos almacenados en el fichero **sub03.srt** ubicado en la carpeta **datos/subtitulos**. La primera línea informa del número y porcentaje de subtítulos cuya duración es inferior a 1.0 segundos. La última línea de la tabla informa del número y porcentaje de subtítulos cuya duración es igual o superior a 5.0 segundos, pero inferior a 6.0 segundos. Ello denota que en TS[0,n-1] no hay subtítulos cuya duración sea igual o superior a 6.0 segundos.

Si la tabla TS[0,n-1] almacenara los 1510 subtítulos del fichero **sub01.srt**, ubicado en la carpeta **datos/subtitulos**, el resultado de una invocación **presentarEstadistica**(TS,n) produciría como resultado la presentación por pantalla de la siguiente estadística:

```
 DURACION [seg.]   SUBTITULOS     PORCENTAJE
===============    ==========     ==========
 0.000 - 0.999          67           4.44%
 1.000 - 1.999         405          26.82%
 2.000 - 2.999         544          36.03%
 3.000 - 3.999         347          22.98%
 4.000 - 4.999         122           8.08%
 5.000 - 5.999          23           1.52%
 6.000 - 6.999           0           0.00%
 7.000 - 7.999           1           0.07%
 8.000 - 8.999           1           0.07%
                   ----------     ----------
                      1510         100.00%
```

Diseño del código de las dos funciones anteriores

El diseño de las dos funciones se simplifica pudiendo declarar, en caso necesario, datos de los tipos **`Tiempo`** y **`Subtitulo`**, definidos en los módulos **`tiempo`** y **`subtitulo`**, respectivamente, y haciendo uso de las funciones visibles definidas en los módulos **`tiempo`**, **`subtitulo`** y **`herramientas`**.

El programa constará únicamente de un módulo principal, que hará uso intensivo de los recursos definidos en los tres módulos que acaban de ser citados.

Se debe entregar, a través de la plataforma ***Moodle2***, un único fichero, el que almacene el código fuente del módulo principal del programa y cuyas primeras líneas sean un comentario inicial con el nombre y apellidos del autor del trabajo.

Esta prueba de programación se valorará según los siguientes criterios:

- La función **`reprogramar`**`(S,mseg)` debe comportarse de acuerdo a lo especificado anteriormente y tener un buen diseño y un código legible [5 puntos]
- La función **`presentarEstadistica`**`(TS,n)` debe comportarse de acuerdo a lo especificado anteriormente y tener un buen diseño y un código legible [5 puntos]
- Un programa que presente errores al ser compilado o presente errores graves al ser ejecutado será valorado con 0 puntos.

Ejemplo ilustrativo del comportamiento del programa

Se reproduce a continuación, el diálogo que debe mantener el programa con el operador:

```
Un fichero de subtitulos (en practicasPROG1/datos/subtitulos): sub01.srt

DURACION [seg.]    SUBTITULOS     PORCENTAJE
===============    ==========     ==========
 0.000 - 0.999          67           4.44%
 1.000 - 1.999         405          26.82%
 2.000 - 2.999         544          36.03%
 3.000 - 3.999         347          22.98%
 4.000 - 4.999         122           8.08%
 5.000 - 5.999          23           1.52%
 6.000 - 6.999           0           0.00%
 7.000 - 7.999           1           0.07%
 8.000 - 8.999           1           0.07%
                   ----------     ----------
                      1510         100.00%

754 00:48:39,770 --> 00:48:42,480
He never had a conviction
except Charlie Kane in his life.

754 00:48:38,215 --> 00:48:40,925
He never had a conviction
except Charlie Kane in his life.

754 00:48:39,771 --> 00:48:42,481
He never had a conviction
except Charlie Kane in his life.
```

Una solución del problema propuesto

```
/*
 * Pre: ---
 * Post: Modifica los tiempos de inicio y de finalización del
 *       subtítulo S adelantando ambos -mseg milisegunos, en
 *       el caso de que mseg sea negativo y retrasando ambos
 *       mseg milisegunos, en el caso de que mseg sea cero o
 *       positivo
 */
void reprogramar ( Subtitulo & S, const int mseg) {
    // Extrae los diferentes datos que definen un subtítulo
    // (su número, su texto y sus tiempo de comienzo y de
    // finalización )
    int num = numero(S);
    Tiempo t_in = principio (S);
    Tiempo t_fin = fin (S);
    int numLineas;
    char textoSubtitulo [MAX_LINEAS][MAX_LONG_LINEA];
    texto (S, numLineas, textoSubtitulo );
    // Discrimina si corresponde retrasar o adelantar el
    // subtitulo
    if (mseg >= 0) {
        // Retrasa el inicio y la finalización del subtítulo
        // mseg milisegundos
        Tiempo retraso = definir (mseg/1000, mseg %1000);
        t_in = sumar( t_in , retraso );
        t_fin = sumar( t_fin , retraso );
    }
    else {
        // Adelanta el inicio y la finalización del subtítulo
        // -mseg milisegundos
        Tiempo adelanto = definir (-mseg/1000, -mseg %1000);
        t_in = restar ( t_in , adelanto );
        t_fin = restar ( t_fin , adelanto );
    }
    // Redefine los valores del subtítulo S, tras haber modificado
    // sus tiempos de inicio y de finalización
    S = definir (num, numLineas, textoSubtitulo , t_in , t_fin );
}


/*
 * Pre: n >= 0
 * Post: Presenta por pantalla una estadistica de los duracciones de los
 *       subtítulos almacenados en TS[0,n-1]. Cada línea informa de los
 *       subtítulos , en número y porcentaje , cuyas duraciones están
 *       comprendidas en un rango de 1 segundo, tal como se ilustra
 *       a continuación:
 *
 *       DURACION [seg.]   SUBTITULOS   PORCENTAJE
 *       =============== ========== ==========
 *        0.000 - 0.999          0        0.00 %
 *        1.000 - 1.999        127       10.17 %
 *        2.000 - 2.999        602       48.20 %
 *        3.000 - 3.999        383       30.66 %
 *        4.000 - 4.999        122        9.77 %
 *        5.000 - 5.999         15        1.20 %
 *                          ---------- ----------
 *                           1249      100.00 %
 */
```

```
void presentarEstadistica (const Subtitulo TS[], const int n) {
    // DIM: su valor, en segundos, ha de ser igual o mayor que la duración
    //      del subtítulo de mayor duración de TS[0,n-1]
    const int DIM = 20;
    int contador[DIM];
    // Puesta a cero de contadores de subtítulos
    for (int i = 0; i < DIM; ++i) { contador[i] = 0; }
    // Contabiliza, por tramos de duración de 1 segundo, el número
    // de subtitulos cuya duración cae en cada tramo
    for (int i = 0; i < n; ++i) {
        Tiempo t = restar ( fin (TS[i ]), principio (TS[i ]));
        int milesimas = 3600000 * hora(t) + 60000 * minuto(t)
                        + 1000 * segundo(t) + milesima(t);
        int indice = milesimas / 1000;
        contador[ indice ] = contador[ indice ] + 1;
    }
    // Presenta por paantalla los resultados. Comienza con el encabezamiento
    // de la tabla de resultados
    cout << endl << "DURACION [seg.]    SUBTITULOS    PORCENTAJE" << endl;
    cout << endl << "===============    ==========    ==========" << endl;
    // Itera la presentación de tramos, hasta que han sido contabilizados los
    // n subtítulos
    int suma = 0;
    int i = 0;
    while (suma < n) {
        // Por cada tramo presenta el número de subtítulos y su porcentaje
        // respecto del total.
        cout << fixed << setw(6) << setprecision(3) << 1.0 * i << " - " << i + 0.999
             << setw(12) << contador[i] << setprecision(2)
             << setw(15) << contador[i] * 100.0 / n << ' %' << endl;
        suma = suma + contador[i];
        i = i + 1;
    }
    // Presenta una línea con la suma de subtítulos contabilizados y la
    // suma de porcentajes
    cout << endl << "                  ----------    ----------" << endl;
    cout << setw(26) << suma << setprecision(2) << setw(15) << suma * 100.0 / n
         << ' %' << endl << endl;
}
```