Examen escrito de Programación 1. Jueves 5 de febrero de 2015

- Se debe disponer de un documento de identificación con fotografía sobre la mesa.
- Se debe comenzar a resolver cada uno de los dos problemas del examen en una hoja de papel diferente, para facilitar su calificación por profesores diferentes. Escribir en cada hoja de papel nombre y apellidos y utilizar, en su caso, ambas caras de la hoja.
- Tiempo máximo para realizar este examen: 3 horas

Problema 1º (3.5 puntos)

Este es un esquema del fichero de interfaz de un módulo de biblioteca denominado **reloj**.

```
/* Fichero reloj.h de interfaz del módulo reloj */

/* Un dato de tipo Reloj representa la hora de un reloj */
struct Reloj {
   . . .
};

/* Pre : . . .
 * Post: ...                                                    */
void ponerEnHora (Reloj& r, int h, int m, int s);

/* Pre : . . .
 * Post: ...                                                    */
void adelantar (Reloj& r, int s);

/* Pre : . . .
 * Post: ...                                                    */
int hora (Reloj r);

/* Pre : . . .
 * Post: ...                                                    */
int minuto (Reloj r);

/* Pre : . . .
 * Post: ...                                                    */
int segundo (Reloj r);

/* Pre : . . .
 * Post: ...                                                    */
void mostrar (Reloj r);
```

La función ***ponerEnHora(r,h,m,s)*** permite definir la hora, `h:m:s`, del reloj *r*. Ejemplos:

ponerEnHora(r,0,0,0)	El reloj *r* ha sido puesto a las 00:00:00
ponerEnHora(r,7,45,30)	El reloj *r* ha sido puesto a las 07:45:30
ponerEnHora(r,23,59,59)	El reloj *r* ha sido puesto a las 23:59:59

La función ***adelantar(r,s)*** permite adelantar o retrasar la hora del reloj *r*. Si $s \geq 0$ la hora del reloj *r* se adelantará *s* segundos y si $s \leq 0$ la hora del reloj *r* se retrasará -*s* segundos. Ejemplos:

adelantar(r,45)	Adelanta la hora de *r* 45 segundos
adelantar(r,90)	Adelanta la hora de *r* un minuto y medio
adelantar(r,1800)	Adelanta la hora de *r* en media hora
adelantar(r,-25)	Retrasa la hora de *r* 25 segundos
adelantar(r,-3615)	Retrasa la hora de *r* una hora y 15 segundos

Las funciones *hora(r)*, *minuto(r)* y *segundo(r)* devuelven, respectivamente, los valores de la hora del día, de los minutos transcurridos desde dicha hora y de los segundos transcurridos desde dicho minuto. Así, por ejemplo,si la hora del reloj **r** pasa 72 segundos del mediodía, *hora(r)* devuelve 12 (la hora), *minuto(r)* devuelve 1 (los minutos que pasan de las 12) y *segundo(r)* devuelve 12 (los segundos que pasan de las 12:01).

La función ***mostrar**(r)* muestra por pantalla la hora del reloj *r* escribiendo una secuencia de 8 caracteres (sin acabar la línea). Dependiendo de la hora del reloj *r*, escribirá por pantalla las siguientes secuencias de 8 caracteres:

Ejemplos:

`00:00:00`	Si la hora de *r* corresponde a la medianoche
`11:58:00`	Si a la hora de *r* le faltan dos minutos para el mediodía
`12:01:30`	Si la hora de *r* es 90 segundos después del mediodía
`23:59:59`	Si a la hora de *r* le falta un segundo para la medianoche

Se pide:

- Escribir completo el fichero de interfaz del módulo de biblioteca **reloj** incluyendo la definición del tipo **Reloj** y la especificación de las seis funciones visibles del módulo. [1.5 puntos]
- Reescribir la especificación y escribir el código C++ de tres de las funciones del módulo **reloj**: ***ponerEnHora**(r,h,m,s)*, ***adelantar**(r,s)* y ***mostrar**(r)*. Se incluirá también, en su caso, la especificación y el código de las funciones auxiliares en las que se apoye el diseño de las tres funciones anteriores.

  En cambio, **no se debe escribir** ni presentar el código de las restantes funciones visibles del módulo **reloj**, es decir, **no se debe escribir** el código de ***hora**(r)*, ***minuto**(r)* y ***segundo**(r)*. [2.0 puntos]

Problema 2º

Se han de especificar y diseñar tres funciones que trabajan con datos de tipo **Reloj** y están ubicadas en otro módulo distinto al módulo **reloj**. Se asume que en el fichero en el que se escriban estas tres funciones se han definido las clásulas ***#include*** que dan visibilidad a otros módulos utilizados por las funciones pedidas.

Primera función [1.5 puntos]. Especificar y diseñar el código de la función ***leerRelojes****(nombreF, T, n)* sabiendo que:

- `nombreF` es el nombre de un fichero binario que almacena una secuencia de datos de tipo **Reloj**.
- El valor final del parámetro `n` ha de ser igual al número de datos de tipo **Reloj** que almacena el fichero `nombreF`.
- El vector `T[0,n-1]` ha de almacenar finalmente una secuencia de datos de tipo **Reloj** igual a la almacenada en el fichero `nombreF`.

```
/*
 * Pre: . . .
 * Post: . . .
 */
void leerRelojes (const char nombreF[], Reloj T[], int& n);
```

Segunda función [2.0 puntos]. Especificar y diseñar el código de la función ***reorganizar****(v, n)* que permute los elementos del vector v[0,n-1] de forma que los relojes cuya hora sea anterior al mediodía precedan a los relojes cuya hora sea igual o posterior al mediodía, es decir, que a los primeros les corresponda un índice inferior en el vector v[0,n-1] que a los segundos. Se valorará la corrección y, muy especialmente, la eficiencia del algoritmo propuesto.

```
/*
 * Pre: . . .
 * Post: . . .
 */
void reorganizar (Reloj v[], int n);
```

Tercera función [3.0 puntos]. Especificar y diseñar el código de la función ***masProximo****(TR, n, relojReferencia)* sabiendo que:

- Los datos almacenados en *TR[0,n-1]* están ordenados cronológicamente: la hora del reloj *TR[i]* es igual o anterior a la del *TR[i+1]*, para todo índice i comprendido entre 0 y $n-2$.
- El valor del parámetro `n` es positivo.
- La función devuelve un dato de tipo **Reloj** cuya hora coincide con la de uno de los datos de *TR[0,n-1]*, aquel cuya hora sea la más próxima a la del reloj `relojReferencia`.

Se valorará la corrección y, muy especialmente, la eficiencia del algoritmo propuesto.

```
/*
 * Pre: . . .
 * Post: . . .
 */
Reloj masProximo (Reloj TR[], int n, Reloj relojReferencia );
```

Una solución del problema 1º

```
/* Fichero reloj.h de interfaz del módulo reloj */

/* Un dato de tipo Reloj representa la hora de un reloj */
struct Reloj {
    int h;                  // Hora marcada por el reloj entre 0 y 23
    int m;                  // Minuto marcado por el reloj entre 0 y 59
    int s;                  // Segundo marcado por el reloj entre 0 y 59
};

/* ¡
 * Pre : h>=0 y h <24 y m>=0 y m<60 y s<=0 y s<24
 * Post: Asigna al reloj r la hora h:m:s
 */
void ponerEnHora (Reloj& r, int h, int m, int s);

/*
 * Pre : ---
 * Post: Si s>=0 entonces adelanta s segundos la hora del reloj r y
 *       si s<=0 entonces retrasa -s segundos la hora del reloj r
 */
void adelantar (Reloj& r, int s);

/*
 * Pre : El reloj r marca una hora igual a H:M:S
 * Post: Devuelve H, es decir, las horas del tiempo que marca el reloj r
 */
int hora (Reloj r);

/*
 * Pre : El reloj r marca una hora igual a H:M:S
 * Post: Devuelve M, es decir, los minutos del tiempo que marca el reloj r
 */
int minuto (Reloj r);

/*
 * Pre : El reloj r marca una hora igual a H:M:S
 * Post: Devuelve S, es decir, los segundos del tiempo que marca el reloj r
 */
int segundo (Reloj r);

/*
 * Pre : ---
 * Post: Escribe por pantalla una secuencia de 8 caracteres para definir la
 *       hora del reloj r: HH:MM:SS, donde el par de caracteres HH define las
 *       horas, el par de caracteres MM los minutos y el par de caracteres SS
 *       los segundos. Ejemplo: 17:05:00
 */
void mostrar (Reloj r);
```

```
/*
 * Pre : h>=0 y h <24 y m>=0 y m<60 y s<=0 y s<24
 * Post: Asigna al reloj r la hora h:m:s
 */
void ponerEnHora (Reloj& r, int h, int m, int s) {
    r.h = h;
    r.m = m;
    r.s = s;
}

/*
 * Pre: --
 * Post: Devuelve los segundos transcurridos desde las 00:00:00 horas
 *       hasta la hora marcada por el reloj r
 */
int aSegundos(Reloj r) {
    return 3600*r.h + 60*r.m + r.s;
}

/*
 * Pre : ---
 * Post: Si s>=0 entonces adelanta s segundo la hora del reloj r y
 *       si s<=0 entonces retrasa -s segundo la hora del reloj r
 */
void adelantar (Reloj& r, int s) {
    const int SEGUNDOS_DIA = 86400;
    int segundos = (aSegundos(r) + s ) % SEGUNDOS_DIA;
    if (segundos < 0) {
        segundos = segundos + SEGUNDOS_DIA;
    }
    r.h = segundos / 3600;
    r.m = (segundos % 3600) / 60;
    r.s = segundos % 60;
}

/*
 * Pre : ---
 * Post: Escribe por pantalla una secuencia de 8 caracteres para definir la
 *       hora del reloj r: HH:MM:SS, donde el par de caracteres HH define las
 *       horas, el par de caracteres MM los minutos y el par de caracteres SS
 *       los segundos. Ejemplo: 17:05:00
 */
void mostrar (Reloj r) {
    cout << setfill ('0') << setw(2) << r.h << ':' << setw(2) << r.m << ':'
         << setw(2) << r.s << flush;
}
```

Una solución del problema 2º

```
/*
 * Pre:  «nombreF» es el nombre de un fichero binario que almacena una secuencia
 *       de datos de tipo «Reloj»
 * Post: El valor del parámetro «n» es igual al número de datos de tipo «Reloj»
 *       que almacena el fichero «nombreF» y el vector T[0, n − 1] almacena una
 *       secuencia de datos de tipo «Reloj» igual a la almacenada en el fichero
 *       «nombreF».
 */
void leerRelojes (char nombreF[], Reloj T[], int& n) {
    ifstream f;
    f.open(nombreF, ios :: binary );
    if (f.is_open ()) {
        n = 0;
        f.read( reinterpret_cast <char*>(&T[n]), sizeof(T[n ]));
        while (!f.eof ()) {
            n++;
            f.read( reinterpret_cast <char*>(&T[n]), sizeof(T[n ]));
        }
        f.close ();
    }
    else {
        cerr << "Error al abrir el fichero \"" << nombreF << "\"" << endl;
        n = −1;
    }
}
```

```
/*
 * Pre:  r1 = R1 y r2 = R2.
 * Post: r1 = R2 y r2 = R1.
 */
void intercambia(Reloj& r1, Reloj& r2) {
    Reloj aux = r1;
    r1 = r2;
    r2 = aux;
}


/*
 * Pre:  n = 0 y «v» tiene al menos «n» componentes.
 * Post: Ha permutado los elementos inciales del vector v[0, n − 1] de forma que
 *       los relojes cuya hora sea anterior al mediodía preceden a los relojes
 *       cuya hora es igual o posterior al mediodía.
 */
void reorganizar (Reloj v[], int n) {
    int inf = 0;
    int sup = n − 1;
    while (inf < sup) {
        if (hora(v[inf ]) < 12) {
            inf++;
        }
        else if (hora(v[sup]) >= 12) {
            sup−−;
        }
        else {
            intercambia(v[inf ], v[sup ]);
            inf++;
            sup−−;
        }
    }
```

```
}
```

```
/*
 * Pre:  ---
 * Post: Ha devuelto el número de segundos del reloj «r» transcurridos desde la
 *       medianoche.
 */
int segundosTotales(Reloj r) {
    return hora(r) * 3600 + minuto(r) * 60 + segundo(r);
}


/*
 * Pre:  ---
 * Post: Ha deuvelto el valor absoluto de la diferencia de tiempo entre r1 y r2.
 */
int diferenciaAbsoluta (Reloj r1, Reloj r2) {
    const int SEGUNDOS_DIA = 24 * 60 * 60;

    int diferencia = segundosTotales(r1) - segundosTotales(r2);
    if (diferencia < 0) {
        diferencia = -diferencia;
    }

    return min(diferencia, SEGUNDOS_DIA - diferencia);
}


/*
 * Pre:  ---
 * Post: Ha devuelto true si y solo si r1 es posterior en el tiempo a r2.
 */
bool esPosterior (Reloj r1, Reloj r2) {
    return segundosTotales(r1) > segundosTotales(r2);
}


/*
 * Pre:  Los datos almacenados en TR[0,n-1] están ordenados cronológicamente y
 *       el valor del parámetro «n» es positivo.
 * Post: Ha devuelto un dato cuya hora coincide con la de uno de los datos de
 *       TR[0,n-1], aquel cuya hora es la más próxima a la del reloj
 *       «referencia».
 */
Reloj masProximo(Reloj TR[], int n, Reloj referencia) {
    // Búsqueda del primer reloj posterior o igual a «referencia»,
    // utilizando el esquema de búsqueda binaria
    int inf = 0;
    int sup = n - 1;
    while (inf < sup) {
        int med = (inf + sup) / 2;
        if (esPosterior(referencia, TR[med])) {
            inf = med + 1;
        }
        else {
            sup = med;
        }
    }
    // inf >= sup.
    // TR[inf] es igual o posterior a referencia, excepto si inf == n-1,
    // cuando TR[inf] puede ser anterior, posterior o igual a referencia.
```

```
    // Determinación de los índices en los que puede estar el reloj más próximo.
    // Caso general:
    int indiceAnterior = inf - 1;
    int indicePosterior = inf;

    // Casos particulares:
    if (inf == 0) {
        indiceAnterior = n - 1;
    }
    else if (inf == n - 1 && esPosterior(referencia, TR[inf])) {
        indiceAnterior = 0;
    }

    // Determinación y devolución del reloj más próximo:
    if (diferenciaAbsoluta(TR[indiceAnterior], referencia)
            < diferenciaAbsoluta(referencia, TR[indicePosterior])) {
        return TR[indiceAnterior];
    }
    else {
        return TR[indicePosterior];
    }
}
```