Examen Práctico de Programación 1 - 27/enero/2017

- Tiempo máximo para realizar el trabajo de programación propuesto: 110 minutos
- Entrega del trabajo a través de la plataforma *Moodle2*.

Especificación del trabajo a desarrollar en el turno 1º (15:00 horas)

Se han de diseñar las funciones **`organizar`**`(L)` [5.0 puntos] y **`buscarSiguiente`**`(L,i)` [5.0 puntos]. Es posible que se opte por un diseño descendente apoyado en alguna o algunas funciones auxiliares. En tal caso, en el documento presentado se incluirá la especificación y el código de dichas funciones auxiliares.

El diseño de ambas funciones debe apoyarse al máximo en las funciones visibles de los módulos **`persona`** y **`listaPersonas`** desarrolladas en la práctica sexta.

Por razones metodológicas está absolutamente prohibido acceder desde el código de ambas funciones a los detalles internos de representación de los tipos de dato **`Persona`** y **`ListaPersonas`**.

```
/*
 * Pre: L = < p_1, ... , p_K >, con K >= 0. De los K miembros de L,
 *      nM son mujeres y nH son hombres, verificando que K = nM + nH
 * Post: Ha permutado las personas de la lista L de forma que ahora sus
 *       nM primeros miembros son mujeres:
 *               L = < p_1, ... , p_nM, ... >
 *       y sus nH últimos miembros son hombres:
 *               L = < ..., p_(nM+1), ... , p_(nM+nH) >
 */
void organizar (ListaPersonas & L) {
    // ... escribir aquí el código de esta función ...
}

/*
 * Pre: Sea N el número de miembros de la lista L, con N >= 2, y el valor de i
 *      está comprendido entre 1 y N.
 * Post: Devuelve una persona miembro de la lista L cuya posición en la lista L
 *       no sea i, cuyo cumpleaños se celebre el mismo día de cada año que el
 *       del i-ésimo miembro de la lista L o en la fecha posterior más próxima.
 */
Persona buscarSiguiente (const ListaPersonas L, const int i) {
    // ... escribir aquí el código de esta función ...
}
```

Presentación del trabajo

Cada una de las dos funciones pedidas se valorará sobre 5.0 puntos. Se evaluará, en primer lugar, el correcto comportamiento de ambas funciones. En caso de que su comportamiento sea el esperado, se valorará la calidad de su diseño y la legibilidad del código tanto de las funciones pedidas como, en su caso, de sus funciones auxiliares.

Cada alumno presentará un fichero, de nombre **`examenPractico01.cc`**, a través de la plataforma **Moodle2** (https://moodle2.unizar.es/) antes de las 16:50. El contenido del fichero será el siguiente: 1) un comentario con nombre y apellidos, 2) la especificación y el código de las funciones auxiliares, si las hubiera, y 3) la especificación y el código de las dos funciones pedidas.

Ejemplos ilustrativos

Por si queda alguna duda, supongamos que se ha definido una lista, **`L`**, de personas integrada por los siguientes miembros:

```
Miembros de la lista L
======================
 1. Hombre nacido el 20/07/1987
 2. Hombre nacido el 27/06/1985
 3. Mujer nacida el 31/08/1990
 4. Mujer nacida el 21/12/1992
 5. Hombre nacido el 10/04/1995
 6. Mujer nacida el 02/02/1989
 7. Mujer nacida el 14/01/1999
 8. Mujer nacida el 09/12/1989
 9. Hombre nacido el 21/12/1990
10. Mujer nacida el 07/01/1989
```

Tras ejecutar la invocación **`organizar`**`(L)`, la composición de la lista **`L`** ha variado. Una posible composición de ella es la que se presenta a continuación (las mujeres preceden a los hombres en la lista):

```
Miembros de la lista L
======================
 1. Mujer nacida el 07/01/1989
 2. Mujer nacida el 09/12/1989
 3. Mujer nacida el 31/08/1990
 4. Mujer nacida el 21/12/1992
 5. Mujer nacida el 14/01/1999
 6. Mujer nacida el 02/02/1989
 7. Hombre nacido el 10/04/1995
 8. Hombre nacido el 27/06/1985
 9. Hombre nacido el 21/12/1990
10. Hombre nacido el 20/07/1987
```

Al ejecutar reiteradamente a continuación la invocación **`buscarSiguiente`**`(L, i)` para valores de **`i`** comprendidos ente 1 y 10, la función devuelve, en cada caso, la persona de la lista **`L`** cuyos datos se detallan en la siguiente tabla:

Invocación a la función	Cumpleaños del miembro i-ésimo de L	Persona devuelta por la función
buscarSiguiente(L,1)	07/01	Mujer nacida el 14/01/1999
buscarSiguiente(L,2)	09/12	Mujer nacida el 21/12/1992
buscarSiguiente(L,3)	31/08	Mujer nacida el 09/12/1989
buscarSiguiente(L,4)	21/12	Hombre nacido el 21/12/1990
. . .	...	. . .
buscarSiguiente(L,8)	27/06	Hombre nacido el 20/07/1987
buscarSiguiente(L,9)	21/12	Mujer nacida el 21/12/1992
buscarSiguiente(L,10)	20/07	Mujer nacida el 31/08/1990

Examen Práctico de Programación 1 - 27/enero/2017

- Tiempo máximo para realizar el trabajo de programación propuesto: 110 minutos
- Entrega del trabajo a través de la plataforma *Moodle2*.

Especificación del trabajo a desarrollar en el turno 2º (17:00 horas)

Se han de diseñar las funciones **`seleccionar`**`(L,desdeMes,hastaMes)` [5.0 puntos] y **`buscar`**`(L,fecha)` [5.0 puntos]. Es posible que se opte por un diseño descendente apoyado en alguna o algunas funciones auxiliares. En tal caso, en el documento presentado se incluirá la especificación y el código de dichas funciones auxiliares.

El diseño de ambas funciones debe apoyarse al máximo en las funciones visibles de los módulos **`persona`** y **`listaPersonas`** desarrolladas en la práctica sexta.

Por razones metodológicas está absolutamente prohibido acceder desde el código de ambas funciones a los detalles internos de representación de los tipos de dato **`Persona`** y **`ListaPersonas`**.

```
/*
 * Pre: Los valores de desdeMes y hastaMes corresponden a números del 1 al 12
 *      ya que representan meses del año.
 *      Denominaremos K al número de miembros de la lista L:
 *          L = < p_1, ... , p_K >, con K >= 0
 *      y denominaremos NUM_MES al número de personas miembros de L cuyo mes
 *      de nacimiento sea uno de los del intervalo de meses [desdeMes,hastaMes]
 * Post: Ha permutado las personas de la lista L de forma que ahora sus
 *       NUM_MES primeros miembros son personas nacidas en un mes perteneciente
 *       al intervalo de meses [desdeMes,hastaMes]:
 *              L = < p_1, ... , p_NUM_MES, ... >
 *       y los restantes K-NUM_MES últimos miembros de L son personas nacidas
 *       en un mes no perteneciente al intervalo de meses [desdeMes,hastaMes]:
 *              L = < ..., p_(NUM_MES+1), ... ,p_K >
 */
void seleccionar (ListaPersonas& L, const int desdeMes, const int hastaMes) {
    // ... escribir aquí el código de esta función ...
}

/*
 * Pre: fecha define una fecha que, escrita en base 10, tiene la forma
 *      aaaammdd donde aaaa es el año, mm el mes y dd el día de la
 *      fecha y la lista L tiene un miembro como mínimo
 * Post: Devuelve la persona miembro de la lista L cuyo cumpleaños
 *       se celebra cada año el mismo día y mes que alguien que
 *       que hubiera nacido en fecha o, en su defecto, se celebra
 *       el día posterior más próximo
 */
Persona buscar (const ListaPersonas L, const int fecha) {
    // ... escribir aquí el código de esta función ...
}
```

Presentación del trabajo

Cada una de las dos funciones pedidas se valorará sobre 5.0 puntos. Se evaluará, en primer lugar, el correcto comportamiento de ambas funciones. En caso de que su comportamiento sea el esperado, se

valorará la calidad de su diseño y la legibilidad del código tanto de las funciones pedidas como, en su caso, de sus funciones auxiliares.

Cada alumno presentará un fichero, de nombre **`examenPractico02.cc`**, a través de la plataforma **Moodle2** (https://moodle2.unizar.es/) antes de las 18:50. El contenido del fichero será el siguiente: 1) un comentario con nombre y apellidos, 2) la especificación y el código de las funciones auxiliares, si las hubiera, y 3) la especificación y el código de las dos funciones pedidas.

Ejemplos ilustrativos

Por si queda alguna duda, supongamos que se ha definido una lista, **`L`**, de personas integrada por los siguientes miembros:

```
 Miembros de la lista L
 ======================
  1. Hombre nacido el 20/07/1987
  2. Hombre nacido el 27/06/1985
  3. Mujer nacida el 31/08/1990
  4. Mujer nacida el 21/12/1992
  5. Hombre nacido el 10/04/1995
  6. Mujer nacida el 02/02/1989
  7. Mujer nacida el 14/01/1999
  8. Mujer nacida el 09/12/1989
  9. Hombre nacido el 21/12/1990
 10. Mujer nacida el 07/01/1989
```

Tras ejecutar la invocación **`seleccionar`**`(L, 2, 6)`, la composición de la lista **`L`** ha variado. Una posible composición de ella es la que se presenta a continuación (las personas nacidas en los meses del intervalo [2,6] preceden en la lista a las restantes personas):

```
 Miembros de la lista L
 ======================
  1. Mujer nacida el 02/02/1989
  2. Hombre nacido el 27/06/1985
  3. Hombre nacido el 10/04/1995
  4. Mujer nacida el 21/12/1992
  5. Mujer nacida el 31/08/1990
  6. Hombre nacido el 20/07/1987
  7. Mujer nacida el 14/01/1999
  8. Mujer nacida el 09/12/1989
  9. Hombre nacido el 21/12/1990
 10. Mujer nacida el 07/01/1989
```

Al ejecutar a continuación reiteradamente la invocación **`buscar`**`(L,fecha)`, la función devuelve, en cada caso, la persona de la lista **`L`** cuyos datos se detallan en la siguiente tabla:

Invocación a la función	Dia de esa fecha	Persona devuelta por la función
buscar(L,19500101)	01/01	Mujer nacida el 07/01/1989
buscar(L,20160413)	13/04	Hombre nacido el 27/06/1985
buscar(L,19631210)	10/12	Mujer nacida el 21/12/1992
buscar(L,20161221)	21/12	Mujer nacida el 21/12/1992
buscar(L,20021222)	22/12	Mujer nacida el 07/01/1989
buscar(L,19821229)	29/12	Mujer nacida el 07/01/1989

Una solución del problema propuesto en el turno 1º

```
/*
 * Pre: L = < p_1, ... , p_K >, con K >= 0. De los K miembros de L,
 *      nM son mujeres y nH son hombres, verificando que K = nM + nH
 * Post: Ha permutado las personas de la lista L de forma que ahora sus
 *       nM primeros miembros son mujeres:
 *              L = < p_1, ... , p_nM, ... >
 *       y sus nH últimos miembros son hombres:
 *              L = < ..., p_(nM+1), ... , p_(nM+nH) >
 */
void organizar ( ListaPersonas & L) {
    // Algoritmo de distribución de los elementos de una tabla
    int inf = 1, sup = numPersonas(L);
    Persona infPersona = consultar (L, inf ),
            supPersona = consultar (L,sup );
    while ( inf < sup) {
        if (esMujer( infPersona )) {
            // La persona inf-ésima de la lista L está bien ubicada
            inf = inf + 1; infPersona = consultar (L, inf );
        }
        else if (!esMujer(supPersona)) {
            // La persona sup-ésima de la lista L está bien ubicada
            sup = sup - 1; supPersona = consultar (L,sup );
        }
        else {
            // Las personas inf-ésima y sup-ésima de la lista L van a
            // ser intercambiadas
            eliminar (L, inf ); insertar (L, supPersona, inf );
            eliminar (L, sup ); insertar (L, infPersona , sup );
            inf = inf + 1; sup = sup - 1;
            infPersona = consultar (L, inf );
            supPersona = consultar (L,sup );
        }
    }
}
```

```
/*
 * Pre: Sea N el número de miembros de la lista L con N >= 2 y el valor de i está
 *      comprendido entre 1 y N.
 * Post: Devuelve una persona miembro de la lista L cuya posición en la lista L
 *       no sea i, cuyo cumpleaños se celebre el mismo día de cada año que el
 *       del i−ésimo miembro de la lista L o,, en su defecto, en la fecha posterior
 *       más próxima
 */
Persona buscarSiguiente (const ListaPersonas L, const int i) {
    // De la fecha de nacimiento [aaaammdd] de la i−ésima persona de L
    // solo interesa el mes y el día [mmdd]
    int diaFecha = nacido( consultar (L,i )) % 10000;
    // Preselecciona un primer candidato: el primer miembro
    // de la lista L, salvo que i sea 1, en cuyo caso preselecciona
    // el segundo miembro de L
    Persona candidato;
    int primero;
    if (i == 1) { primero = 2; }
    else { primero = 1; }
    candidato = consultar (L,primero);
    // Calcula el día de su cumpleaños [mmdd]
    int diaCumple = nacido( candidato ) % 10000;
    // Calcula diferencia, teniendo en cuenta que depende del orden
    // en un mismo año entre diaCumple y diaFecha
    int diferencia ;
    if (diaCumple >= diaFecha) { diferencia = diaCumple − diaFecha; }
    else { diferencia = 1231 − diaFecha + diaCumple; }
    // Analiza los miembros de L ubicados en las posiciones posteriores
    // a primero en busca de un candidato mejor
    int nP = numPersonas(L);
    for (int j = primero + 1; j <= nP; ++j) {
        if (j != i) {
            // Procede a analizar al miembro j−ésimo de L
            Persona nueva = consultar (L,j );
            // Calcula el dia de su cumpleaños [mmdd]
            int diaCumpleNueva = nacido(nueva) % 10000;
            // Calcula la diferencia entre el día de su cumpleaños
            // y el día definido por fecha
            int diferenciaNueva;
            if (diaCumpleNueva >= diaFecha) { diferenciaNueva = diaCumpleNueva − diaFecha; }
            else { diferenciaNueva = 1231 − diaFecha + diaCumpleNueva; }
            if (diferenciaNueva < diferencia ) {
                // la persona nueva es un candidato mejor que el seleccionado
                // previamente
                diferencia = diferenciaNueva;
                candidato = nueva;
            }
        }
    }
    return candidato;
}
```

Una solución del problema propuesto en el turno 2º

```
/*
 * Pre: Los valores de desdeMes y hastaMes corresponden a números del 1 al 12
 *      ya que representan meses del año.
 * Post: Devuelve true si y solo si el mes de nacimiento de la persona p
 *       está en el intervalo de meses [desdeMes,hastaMes]
 */
bool esSuMes (const Persona p, const int desdeMes, const int hastaMes) {
    // Calcula el mes de nacimiento de la persona p
    int mes = nacido(p) / 100 % 100;
    // Comprueba si mes está o no está en [desdeMes,hastaMes]
    return desdeMes <= mes && mes <= hastaMes;
}

/*
 * Pre: Los valores de desdeMes y hastaMes corresponden a números del 1 al 12
 *      ya que representan meses del año.
 *      Denominaremos K al número de miembros de la lista L:
 *          L = < p_1, ... , p_K >, con K >= 0
 *      y denominaremos NUM_MES al número de personas miembros de L cuyo mes
 *      de nacimiento sea uno de los del intervalo de meses [desdeMes,hastaMes]
 * Post: Ha permutado las personas de la lista L de forma que ahora sus
 *       NUM_MES primeros miembros son personas nacidas en un mes perteneciente
 *       al intervalo de meses [desdeMes,hastaMes]:
 *              L = < p_1, ... , p_NUM_MES, ... >
 *       y los restantes K-NUM_MES últimos miembros de L son personas nacidas
 *       en un mes no perteneciente al intervalo de meses [desdeMes,hastaMes]:
 *              L = < ..., p_(NUM_MES+1), ... ,p_K >
 */
void seleccionar (ListaPersonas& L, const int desdeMes, const int hastaMes) {
    // Algoritmo de distribución de los elementos de una tabla
    int inf = 1, sup = numPersonas(L);
    Persona infPersona = consultar (L, inf),
            supPersona = consultar (L,sup);
    while (inf < sup) {
        if (esSuMes(infPersona, desdeMes, hastaMes)) {
            // La persona inf-ésima de la lista L está bien ubicada
            inf = inf + 1; infPersona = consultar (L, inf);
        }
        else if (!esSuMes(supPersona, desdeMes, hastaMes)) {
            // La persona sup-ésima de la lista L está bien ubicada
            sup = sup - 1; supPersona = consultar (L,sup);
        }
        else {
            // Las personas inf-ésima y sup-ésima de la lista L van a
            // ser intercambiadas
            eliminar (L, inf); insertar (L, supPersona, inf);
            eliminar (L, sup); insertar (L, infPersona, sup);
            inf = inf + 1; sup = sup - 1;
            infPersona = consultar (L, inf);
            supPersona = consultar (L,sup);
        }
    }
}
```

```
/*
 * Pre: fecha define una fecha que, escrita en base 10, tiene la forma
 *      aaaammdd donde aaaa es el año, mm el mes y dd el día de la
 *      fecha y la lista L tiene un miembro como mínimo
 * Post: Devuelve la persona miembro de la lista L cuyo cumpleaños
 *       se celebra cada año el mismo día y mes que alguien que
 *       que hubiera nacido en fecha o, en su defecto, se celebra
 *       el día posterior más próximo
 */
Persona buscar (const ListaPersonas L, const int fecha) {
    // De fecha [aaaammdd] solo interesa el mes y el día [mmdd]
    int diaFecha = fecha % 10000;
    // Preselecciona como candidato al primer miembro de la lista L
    Persona candidato = consultar (L,1);
    // Calcula el día de su cumpleaños [mmdd]
    int diaCumple = nacido(candidato) % 10000;
    // Calcula diferencia, teniendo en cuenta que depende del orden
    // en un mismo año entre diaCumple y diaFecha
    int diferencia;
    if (diaCumple >= diaFecha) { diferencia = diaCumple - diaFecha; }
    else { diferencia = 1231 - diaFecha + diaCumple; }
    // Analiza los miembros de L ubicados en las posiciones 2 a nP
    // en busca de un candidato mejor
    int nP = numPersonas(L);
    for (int i = 2; i <= nP; ++i) {
        // Procede a analizar al miembro i-ésimo de L
        Persona nueva = consultar (L,i);
        // Calcula el dia de su cumpleaños [mmdd]
        int diaCumpleNueva = nacido(nueva) % 10000;
        // Calcula la diferencia entre el día de su cumpleaños
        // y el día definido por fecha
        int diferenciaNueva;
        if (diaCumpleNueva >= diaFecha) { diferenciaNueva = diaCumpleNueva - diaFecha; }
        else { diferenciaNueva = 1231 - diaFecha + diaCumpleNueva; }
        if (diferenciaNueva < diferencia) {
            // la persona nueva es un candidato mejor que el seleccionado
            // previamente
            diferencia = diferenciaNueva; candidato = nueva;
        }
    }
    return candidato;
}
```