Examen de Prácticas de Programación 1 - 5/febrero/2015

- Tiempo máximo para realizar el trabajo de programación propuesto: 90 minutos
- Entrega del trabajo a través de la plataforma *Moodle2*.

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

Se debe diseñar el código de la función **`seleccionarPorMes`**`(nombreFBC, m, ficheroResultados)` donde:

- **`nombreFBC`** define el nombre de un fichero binario que almacena exclusivamente una secuencia de datos de tipo **`Ciudadano`**.
- El parámetro **`m`** define el número de un mes del año: 1 (enero), 2 (febrero), . . . , 12 (diciembre).
- **`ficheroResultados`** define el nombre de un fichero binario que ha de ser creado y almacenará finalmente una secuencia de datos de tipo **`Ciudadano`**, concretamente una copia de todos los datos del fichero **`nombreFBC`** que correspondan a ciudadanos nacidos en el mes número 'm' con la particularidad de que los ciudadanos han de estar almacenados en este fichero según valores de su DNI crecientes.
- La función, además de crear el fichero **`ficheroResultados`** con el contenido descrito, informará por pantalla del número de ciudadnos leídos del fichero **`nombreFBC`** y del número de ciudadanos almacenados en el fichero **`ficheroResultados`**. Por ejemplo:

  Leidos 1247 ciudadanos y creado un fichero con los 97 nacidos en el mes 11

```
/*
 * Pre:  ...
 * Post:  ...
 */
void seleccionarPorMes (const char nombreFBC[], int m, const char ficheroResultados []);
```

Desarrollar la función en un nuevo proyecto denominado **ExamenFebrero2015** ubicado en el área de trabajo **practica5**. En el desarrollo de la función se debe hacer uso de los módulos de biblioteca **ciudadano**, **fecha** y **nif**, cuyos ficheros de interfaz y de especificación se encuentran ubicados en **practicasPROG1/definicionDatos/Tipos**.

Si se desea, puede hacerse uso de los recursos programados en el módulo **gestiones**, ubicado en el proyecto **Gestiones** en el área de trabajo **practica5** para facilitar la realización de pruebas para comprobar el buen funcionamiento de la función pedida. En cambio no debe hacerse uso de los recursos del módulo **gestiones** en el desarrollo de la función pedida ni de sus funciones auxiliares, ya que se pretende hacer un diseño no dependiente de dicho módulo.

Se debe entregar, a través de la plataforma *Moodle2*, un fichero que incluya:

- Un comentario inicial con el nombre y apellidos del autor del trabajo.
- Las especificaciones y el código de la funciones auxiliares que se hayan desarrollado para facilitar el diseño de la función **`seleccionarPorMes`**`(nombreFBC, m, ficheroResultados)`.
- La especificación y el código de la función **`seleccionarPorMes`**`(nombreFBC, m, ficheroResultados)`

Examen de Prácticas de Programación 1 - 5/febrero/2015

- Tiempo máximo para realizar el trabajo de programación propuesto: 90 minutos
- Entrega del trabajo a través de la plataforma *Moodle2*.

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

Se debe diseñar el código de la función **`seleccionarPorLetra`**`(nombreFBC, c, fResultados)` donde:

- **`nombreFBC`** define el nombre de un fichero binario que almacena exclusivamente una secuencia de datos de tipo **`Ciudadano`**.
- El valor de **`c`** define una de las 23 letras mayúsculas que pueden estar asociadas a un NIF.
- **`fResultados`** define el nombre de un fichero binario que ha de ser creado y que almacenará finalmente una secuencia de datos de tipo **`Ciudadano`**, concretamente una copia de todos los datos del fichero **`nombreFBC`** que correspondan a ciudadanos cuya letra de NIF sea igual al valor de 'c' con la particularidad de que estos ciudadanos han de estar almacenados en este fichero según su fecha de nacimiento, comenzando por los más viejos y finalizando por los más jóvenes.
- La función, además de crear el fichero **`fResultados`** con el contenido descrito, informará por pantalla del número de ciudadanos leídos del fichero **`nombreFBC`** y del número de ciudadanos almacenados en el fichero **`fResultados`**. Por ejemplo:

  Leidos: 1247 ciudadanos y creado un fichero con los 72 cuya letra de NIF es la M

```
/*
 * Pre:  ...
 * Post:  ...
 */
void seleccionarPorLetra (const char nombreFBC[], char c, const char fResultados []);
```

Desarrollar la función en un nuevo proyecto denominado **ExamenFebrero2015** ubicado en el área de trabajo **practica5**. En el desarrollo de la función se debe hacer uso de los módulos de biblioteca **ciudadano**, **fecha** y **nif**, cuyos ficheros de interfaz y de especificación se encuentran ubicados en **practicasPROG1/definicionDatos/Tipos**.

Si se desea, puede hacerse uso de los recursos programados en el módulo **gestiones**, ubicado en el proyecto **Gestiones** en el área de trabajo **practica5** para facilitar la realización de pruebas para comprobar el buen funcionamiento de la función pedida. En cambio no debe hacerse uso de los recursos del módulo **gestiones** en el desarrollo de la función pedida ni de sus funciones auxiliares, ya que se pretende hacer un diseño no dependiente de dicho módulo.

Se debe entregar, a través de la plataforma *Moodle2*, un fichero que incluya:

- Un comentario inicial con el nombre y apellidos del autor del trabajo.
- Las especificaciones y el código de la funciones auxiliares que se hayan desarrollado para facilitar el diseño de la función **`seleccionarPorLetra`**`(nombreFBC, c, fResultados)`.
- La especificación y el código de la función **`seleccionarPorLetra`**`(nombreFBC, c, fResultados)`

Una solución del problema propuesto en el turno 1º

```
/*
 * Pre: ---
 * Post: Devuelve 0 si uno y otro tiene el mismo número de DNI, devuelve un
 *       valor negativo si el número de DNI de uno es inferior al de otro y
 *       devuelve un valor positivo si el número de DNI de uno es superior
 *       al de otro
 */
int comparar (Ciudadano uno, Ciudadano otro) {
    return dni(nif(uno)) - dni(nif(otro));
}

/*
 * Pre: uno = X y otro = Y
 * Post: uno = Y y otro = X
 */
void permutar (Ciudadano& uno, Ciudadano& otro) {
    Ciudadano c = uno;
    uno = otro;   otro = c;
}

/*
 * Pre: Todos los elementos de T[0..n-1] almacenan información de n ciudadanos con n>0
 * Post: T[0..n-1] es una permutación de los datos iniciales de T[0..n-1] y todos
 *       ellos están ordenados de forma que el número de DNI de cada ciudadano
 *       de T[0..n-2] es menor que el del que le sigue.
 */
void ordenar (Ciudadano T[], int n) {
    /*
     * Ordenación de una tabla por el método de selección
     */
     for (int i=0; i<n-1; ++i) {
        /* Los ciudadanos de T[0..i-1] tienen los DNI más bajos y ya están ordenados */
         /* Selecciona el ciudadano con DNI más bajo de T[i..n-1] */
         int iMenor = i;
         for (int j=i+1; j<n; ++j) {
             /* T[iMenor] tiene el DNI más bajo de T[i..j-1] */
             if (comparar(T[j], T[iMenor])<0) {
                 iMenor = j;
             }
             /* T[iMenor] tiene el DNI más bajo de T[i..j] */
         }
         /* T[iMenor] tiene el DNI más bajo de T[i..n-1]. Permuta T[i] y T[iMenor] */
         permutar(T[i], T[iMenor]);
         /* Los ciudadanos de T[0..i] tienen los DNI más bajos y ya están ordenados */
     }
}

/*
 * Pre: 'nombreFBC' es el nombre de un fichero binario que almacena una secuencia
 *      de datos de tipo Ciudadano y m>=1 y m<=12
 * Post: ' ficheroResultados ' es el nombre de un fichero binario en el que se ha escrito
 *       una copia de todos los datos del fichero 'nombreFBC' que corresponden a ciudadanos
 *       nacidos en el mes 'm' y los datos escritos están ordenados según números de DNI
 *       crecientes . También informa por pantalla del número de ciudadanos leídos y del
 *       número de ciudadanos nacidos en el mes 'm' del siguiente modo:
 *         Leidos: 1247 ciudadanos. Creado un fichero con los 97 nacidos en el mes 11
 */
void seleccionarPorMes (const char nombreFBC[], int m, const char ficheroResultados []) {
    const int MAX = 1000;
```

```
    Ciudadano T[MAX];
    ifstream fDatos;
    fDatos.open(nombreFBC, ios::binary );
    ofstream fResultados;
    fResultados.open( ficheroResultados , ios :: binary );
    /*
     * Lee los datos almacenados en el fichero binario asociado a fDatos y,
     * los que correspondan a ciudadanos nacidos en el mes 'm' los almacena
     * en T[0,cuentaResultados-1]
     */
    int cuentaDatos = 0, cuentaResultados = 0;
    while (!fDatos.eof()) {
        fDatos.read( reinterpret_cast<char *>(&T[cuentaResultados]), sizeof(Ciudadano));
        if (!fDatos.eof()) {
            Fecha f = nacimiento(T[cuentaResultados]);
            cuentaDatos = cuentaDatos + 1;
            if (mes(f)==m) {
                cuentaResultados = cuentaResultados + 1;
            }
        }
    }
    fResultados.close();
    /*
     * Ordena T[0,cuentaResultados-1] según valores de sus DNI crecientes
     */
    ordenar(T, cuentaResultados);
    /*
     * Escribe los datos T[0,cuentaResultados-1] en el fichero binario asociado
     * a fResultados
     */
    for (int i = 0; i < cuentaResultados; i++) {
        fResultados.write( reinterpret_cast<char *>(&T[i]), sizeof(Ciudadano));
        cout << fechaCompacta(nacimiento(T[i])) << "    ";
        cout << dni(nif(T[i])) << "-" << letra(nif(T[i])) << endl;
    }
    fDatos.close();
    /*
     * Informa mediante un mensaje por pantalla del número de datos de ciudadanos
     * leidos y del número de ellos que han nacido en el mes 'm'
     */
    cout << "Leidos " << cuentaDatos << " ciudadanos. Creado un fichero con los "
         << cuentaResultados << " nacidos en el mes " << m << endl;
}
```

Una solución del problema propuesto en el turno 2º

```
/*
 * Pre: ---
 * Post: Devuelve 0 si uno y otro han nacido en la misma fecha, devuelve
 *       un valor negativo si uno ha nacido antes que otro y devuelve
 *       un valor positivo si uno ha nacido después que otro
 */
int comparar (Ciudadano uno, Ciudadano otro) {
    return fechaCompacta(nacimiento(uno)) - fechaCompacta(nacimiento(otro));
}

/*
 * Pre: uno = X y otro = Y
 * Post: uno = Y y otro = X
 */
void permutar (Ciudadano& uno, Ciudadano& otro) {
    Ciudadano c = uno;
    uno = otro;   otro = c;
}

/*
 * Pre: Todos los elementos de T[0..n-1] almacenan información de n ciudadanos con n>0
 * Post: T[0..n-1] es una permutación de los datos iniciales de T[0..n-1] y todos
 *       ellos están ordenados de forma que el número de DNI de cada ciudadano
 *       de T[0..n-2] es menor que el del que le sigue.
 */
void ordenar2 (Ciudadano T[], int n) {
    /*
     * Ordenación de una tabla por el método de selección
     */
     for (int i=0; i<n-1; ++i) {
        /* Los ciudadanos de T[0..i-1] tienen los DNI más bajos y ya están ordenados */
         /* Selecciona el ciudadano con DNI más bajo de T[i..n-1] */
         int iMenor = i;
         for (int j=i+1; j<n; ++j) {
             /* T[iMenor] tiene el DNI más bajo de T[i..j-1] */
             if (comparar(T[j], T[iMenor])<0) {
                 iMenor = j;
             }
             /* T[iMenor] tiene el DNI más bajo de T[i..j] */
         }
         /* T[iMenor] tiene el DNI más bajo de T[i..n-1]. Permuta T[i] y T[iMenor] */
         permutar(T[i], T[iMenor]);
         /* Los ciudadanos de T[0..i] tienen los DNI más bajos y ya están ordenados */
     }
}

/*
 * Pre: 'nombreFBC' es el nombre de un fichero binario que almacena una secuencia de
 *      datos de tipo Ciudadano y c es una de las letras mayúsculas asociadas a los NIF
 * Post: ' ficheroResultados ' es el nombre de un fichero binario en el que se ha escrito
 *       una copia de todos los datos del fichero 'nombreFBC' que corresponden a ciudadanos
 *       cuya letra de su NIP es 'c' y los datos escritos están ordenados según la edad
 *       de los ciudadanos, comenzando por los más viejos y acabando por los más jóvenes.
 *       También informa por pantalla del número de ciudadanos leídos y del número
 *       de ciudadanos cuya letra de su NIP es 'c' del siguiente modo:
 *         Leidos: 1247 ciudadanos. Creado un fichero con los 30 cuya letra de NIP es la H
 */
void seleccionarPorLetraXXX (const char nombreFBC[], char c, const char ficheroResultados []) {
    const int MAX = 1000;
```

```
    Ciudadano T[MAX];
    ifstream fDatos;
    fDatos.open(nombreFBC, ios::binary );
    ofstream fResultados;
    fResultados.open( ficheroResultados , ios :: binary );
    /*
     * Lee los datos almacenados en el fichero binario asociado a fDatos y,
     * los que correspondan a ciudadanos cuya letra de NIF sea 'c' los almacena
     * en T[0,cuentaResultados-1]
     */
    int cuentaDatos = 0, cuentaResultados = 0;
    while (!fDatos.eof()) {
        fDatos.read( reinterpret_cast<char *>(&T[cuentaResultados]), sizeof(Ciudadano));
        if (!fDatos.eof()) {
            = cuentaDatos + 1;
            if ( letra ( nif(T[cuentaResultados]))==c) {
                cuentaResultados = cuentaResultados + 1;
            }
        }
    }
    fResultados.close();
    /*
     * Ordena T[0,cuentaResultados-1] según su fecha de nacimiento, comenzando con
     * los más viejos y acabando con los más jóvenes
     */
    ordenar(T, cuentaResultados );
    /*
     * Escribe los datos T[0,cuentaResultados-1] en el fichero binario asociado
     * a fResultados
     */
    for (int i = 0; i < cuentaResultados; i++) {
        fResultados.write( reinterpret_cast<char *>(&T[i]), sizeof(Ciudadano));
        cout << fechaCompacta(nacimiento(T[i])) << "    ";
        cout << dni(nif(T[i])) << "-" << letra(nif(T[i])) << endl;
    }
    fDatos.close();
    /*
     * Informa mediante un mensaje por pantalla del número de datos de ciudadanos
     * leidos y del número de ellos cuya letra de NIP es la 'c'
     */
    cout << "Leidos " << cuentaDatos << " ciudadanos. Creado un fichero con los "
         << cuentaResultados << " cuya letra de NIF es la " << c << endl;
}
```