Examen de prácticas de Programación 1

Escuela de Ingeniería y Arquitectura
Departamento de Informática e Ingeniería de Sistemas

2 de septiembre de 2015

- Tiempo para realizar el trabajo de programación propuesto: **90 minutos**.
- Entrega del trabajo a través de la plataforma Moodle, a la que deberá subirse el fichero correspondiente al código fuente C++ con la función `main` que se solicita y **todos aquellos ficheros** que sean necesarios para poder **compilar** el mismo.

Especificación del trabajo que debe realizarse

Se pide un programa solicite al operador el nombre de un fichero de texto que contenga fechas a razón de una por línea en formato *aaaammdd*, donde *aaaa* es el año, *mm* es el número del mes y *dd* el día del mes. El programa deberá escribir las mismas fechas en la pantalla, pero en formato *dd.mm.aaaa*, utilizando números romanos y ordenadas cronológicamente. El programa también informará de cuántas fechas ha escrito en la pantalla.

Por ejemplo, supongamos que el contenido de un fichero denominado `fechas.txt` fuese el siguiente:

```
20140619
19781206
20140921
20080614
14921012
20150902
20040311
20080914
18080502
```

En ese caso, la ejecución del programa solicitado cuando se introdujese el nombre del fichero anterior presentaría una interacción con el operador como la que sigue:

```
Escriba el nombre del fichero con las fechas: fechas.txt

XII.X.MCDXCII
II.V.MDCCCVIII
VI.XII.MCMLXXVIII
XI.III.MMIV
XIV.VI.MMVIII
XIV.IX.MMVIII
XIX.VI.MMXIV
XXI.IX.MMXIV
II.IX.MMXV

9 fechas en total, comprendidas en el intervalo [ 12.10.1492 - 02.09.2015 ].
```

En esta prueba se valorarán los siguientes aspectos y con este orden de importancia:

1. Comportamiento del programa según las especificaciones de este enunciado.
2. Legibilidad del código.
3. Diseño algorítmico del código.
4. Reutilización al máximo del módulo `romanos` de la práctica 6, cuyos ficheros de interfaz y de implementación deben entregarse junto con el fichero que contenga el programa solicitado.

Un programa que tenga errores de compilación o que, al ser ejecutado, no proporcione ningún resultado, será calificado con un cero. Se sugiere resolver el problema incrementalmente, añadiendo código que vaya cumpliendo con los requisitos gradualmente.

Solución

```
#include <iostream>
#include <fstream>
#include <iomanip>
#include "../../practica6/Romanos/romanos.h"
using namespace std;

// Estimación del número máximo de fechas que habrá en el fichero
const int MAX_FECHAS = 200;

// Símbolo para separar elementos en una fecha
const char SEPARADOR_FECHA = '.';

/*
 * Pre:  n >= 0
 * Post: fechas[0..n-1] es una permutación de los datos iniciales de
 *       fechas[0..n-1] y todos ellos están ordenados de forma que cada uno
 *       de ellos indexado por un índice i en el intervalo [0, n - 2] es
 *       anterior o igual cronológicamente al indexado por i + 1.
 */
void ordenar(int fechas[], int n) {
    // Ordenación de una tabla por el método de selección:
    // El orden cronológico de fechas coincide con el orden natural de enteros
    // con la codificación de fechas seleccionada
    for (int i = 0; i < n - 1; i++) {

        // Selección de la menor fecha de fechas[i..n-1]
        int iMenor = i;
        for (int j = i + 1 ; j < n; j++) {
            // fechas[iMenor] es la menor de fechas[i..j-1]
            if (fechas[j] < fechas[iMenor]) {
                iMenor = j;
            }
            // fechas[iMenor] la menor de fechas[i..j]
        }
        // fechas[iMenor] es la menor de fechas[i..n-1]:
        // permutación fechas[i] y fechas[iMenor].
        int aux = fechas[i];
        fechas[i] = fechas[iMenor];
        fechas[iMenor] = aux;

        // Las fechas de fechas[0..i-1] son las menores de la tabla
```

```
        // y ya están ordenadas.
    }
}

/*
 * Pre:  «fecha» representa una fecha válida en formato «aaaammdd», donde
 *       «aaaa» es el año, «mm» es el número del mes y «dd» el día del mes.
 * Post: «dia» es el día del mes de la fecha representada por «fecha»,
 *       «mes» es el número del mes de la fecha representada por «fecha» y
 *       «agno» es el año de la fecha representada por «fecha».
 */
void descomponerFecha(int fecha, int& dia, int& mes, int& agno) {
    dia = fecha % 100;
    mes = fecha / 100 % 100;
    agno = fecha / 10000;
}

/*
 * Pre:  «fecha» representa una fecha válida en formato «aaaammdd», donde
 *       «aaaa» es el año, «mm» es el número del mes y «dd» el día del mes.
 *       El año correspondiente a «fecha» está comprendido entre 1 y 3999,
 *       ambos inclusive.
 * Post: Ha escrito en pantalla la fecha representada por el valor de «fecha»
 *       en formato dd.mm.aaaa y utilizando números romanos.
 */
void escribirFechaRomana(int fecha) {
    int dd, mm, aaaa;
    descomponerFecha(fecha, dd, mm, aaaa);

    /* En esta función de hace uso de otra función denominada
     * «convertirARomano», definida en el módulo «romanos» y que asigna a su
     * segundo argumento la secuencia de caracteres que definen un número
     * romano equivalente al valor del primer argumento. */

    char dia[MAX_LONG_ROMANO];
    convertirARomano(dd, dia);

    char mes[MAX_LONG_ROMANO];
    convertirARomano(mm, mes);

    char agno[MAX_LONG_ROMANO];
    convertirARomano(aaaa, agno);

    cout << dia << SEPARADOR_FECHA << mes << SEPARADOR_FECHA << agno << endl;
```

```
}


/*
 * Pre:  «fecha» representa una fecha válida en formato «aaaammdd», donde
 *       «aaaa» es el año, «mm» es el número del mes y «dd» el día del mes.
 * Post: Ha escrito en pantalla la fecha representada por el valor de «fecha»
 *       en formato dd.mm.aaaa, utilizando números arábigos y completando con
 *       ceros a la izquierda cuando es necesario.
 */
void escribirFechaDecimal(int fecha) {
    int dia, mes, agno;
    descomponerFecha(fecha, dia, mes, agno);
    cout << setfill('0')
         << setw(2) << dia << SEPARADOR_FECHA
         << setw(2) << mes << SEPARADOR_FECHA
         << setw(4) << agno;
}

/*
 * Pre:  ---
 * Post: Ha solicitado al operador el nombre de un fichero de texto que
 *       contuviera fechas a razón de una por línea en formato «aaaammdd»,
 *       donde «aaaa» es el año, «mm» es el número del mes y «dd» el día del
 *       mes. A continuación, ha escrito las mismas fechas en la pantalla,
 *       pero en formato \emph{dd.mm.aaaa}, utilizando números romanos y
 *       ordenadas cronológicamente. El programa también informará de cuántas
 *       fechas ha escrito en la pantalla y del intervalo al que pertenecían.
 */
int main() {
    // Petición y lectura del nombre del fichero de texto de fechas.
    cout << "Escriba␣el␣nombre␣del␣fichero␣con␣las␣fechas:␣" << flush;
    char nombreFichero[256];
    cin >> nombreFichero;
    cout << endl;

    // Apertura del fichero para su lectura y comprobación.
    ifstream f(nombreFichero);
    if (f.is_open()) {
        // Declaración de una tabla para almacenar las fechas leídas.
        int fechas[MAX_FECHAS];

        // Intento de lectura de una primera fecha del fichero a la tabla.
        int cuenta = 0;
```

```
        f >> fechas[cuenta];

        while (!f.eof()) {
            // Intento de lectura de una siguiente fecha.
            cuenta++;
            f >> fechas[cuenta];
        }
        // Fin de la lectura de datos. Cierre del fichero.
        f.close();

        // Ordenación cronológica de las fechas leídas.
        ordenar(fechas, cuenta);

        // Escritura en orden cronológico de las fechas leídas.
        for (int i = 0; i < cuenta; i++) {
            escribirFechaRomana(fechas[i]);
        }

        // Escritura del número total de fechas leídas y el intervalo al que
        // pertenecen.
        cout << endl;
        cout << cuenta << "␣fechas␣en␣total,␣comprendidas␣en␣el␣intervalo␣[␣";
        escribirFechaDecimal(fechas[0]);
        cout << "␣-␣";
        escribirFechaDecimal(fechas[cuenta-1]);
        cout << "␣]." << endl;

        // Finalización correcta del programa.
        return 0;
    }
    else {
        // Finalización errónea del programa (no se pudo leer del fichero cuyo
        // nombre introdujo el operador).
        return -1;
    }
}
```