Examen Práctico de Programación 1 - 31/enero/2018

- 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 ha de diseñar un programa cuyo comportamiento es análogo al del programa desarrollado en el trabajo obligatorio.

El programa pregunta al operador por los nombres de los ficheros que almacenan la información de los alumnos de un grupo (un fichero de texto) y sus calificaciones (un fichero binario que almacena las calificaciones de todos los alumno del grupo). Ambos ficheros deben haber sido ubicados previamente en el directorio **`../../datos/`**.

A continuación, el programa presenta un listado de los alumnos del grupo cuyo contenido y formato es análogo al presentado por el programa desarrollado en el trabajo obligatorio con la salvedad que ahora los alumnos están ordenados según calificaciones decrecientes y, en caso de igualdad en su mejor calificación, están ordenados alfabéticamente por apellidos y, en caso de coincidencia de éstos, se considera el orden alfabético de sus nombres. Conviene observar que las peores calificaciones son las que corresponden a alumnos no presentados en ninguna de las dos convocatorias.

```
Fichero de alumnos del grupo: ../../datos/grupo.txt
Fichero binario de notas del grupo: ../../datos/notasGrupo.bin

            LISTADO DE CALIFICACIONES DE PROGRAMACION 1
            ===========================================

CODIGO  NOMBRE         APELLIDOS                   FEBRERO   SEPTIEMBRE
======  ============== =========================== =======   ==========
52686777 PAULA         FONSECA CARPINTERO          SB  9.4
52211139 ROSA          MUNDO CASTELLANO            SB  9.1
72129653 CONCEPCION    GARCES GARRAPIZ             NO  7.2
52593434 ROLDAN        CARRO BADENES               NO  7.1
52447916 ANA ISABEL    ZARAGOZA ZABALZA            SU  3.0   NO 7.1
52341323 CLARA         DOMINGUEZ GARRAPIZ          AP  6.5
52219175 ODON          OLIVENZA ACIN               AP  6.4
 5264028 JARA          ZAMORA HORNILLOS            SU  3.0   AP 6.1
52646586 RAUL          LABRADOR LISO               AP  5.8
72514465 FERNANDO      ZABALZA SANTIAGO            NP        AP 5.1
   ...     ...              ...                        ...
52373123 ALICIA        ESPINOSA OLIVITO            NP        SU 3.6
62257732 GUILLERMO     MELUS MELUS                 SU  3.3   NP
52242985 JUAN CARLOS   HERNANDO FONSECA            SU  2.6   SU  2.2
52051729 DAVID         DOMINGO BELMEZ              NP        NP
```

El desarrollo del programa se apoyará en el módulo **grupo** del trabajo obligatorio y almacenado en el área de trabajo **`biblioteca`**. El programa constará de un único fichero con el código del módulo principal del programa.

Se sugiere plantear el desarrollo del programa pedido partiendo del código del programa **listarNotas** diseñado en el trabajo obligatorio, reutilizando al máximo el código de sus funciones, haciendo en ellas las modificaciones que sean precisas, y diseñando las funciones adicionales que sean necesarias.

Presentación del trabajo y criterios de evalución

Cada alumno presentará el fichero con el código del módulo principal del programa a través de la plataforma **Moodle2** (https://moodle2.unizar.es/) antes de las 16:50. Este fichero vendrá encabezado por un comentario con el nombre y apellidos del alumno.

En este examen práctico **se valorará esencialmente que el programa pueda ser ejecutado y proporcione los resultados esperados**, es decir, un listado de los alumnos del grupo ordenados según calificaciones decrecientes, tal como se ha detallado anteriormente.

También se valorará el diseño del programa, la adecuada especificación de cada una de las funciones que integran el módulo principal y la legibilidad del código atendiendo a los criterios de la **"*Guía de estilo para programar en C++*"** publicada en la web de la asignatura.

Examen Práctico de Programación 1 - 31/enero/2018

- 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 ha de diseñar un programa cuyo comportamiento que, en lo esencial, es semejante al del programa desarrollado en el trabajo obligatorio.

El programa pregunta al operador por los nombres de los ficheros que almacenan la información de los alumnos de un grupo (un fichero de texto) y sus calificaciones (un fichero binario que almacena las calificaciones de todos los alumno del grupo). Ambos ficheros deben haber sido ubicados previamente en el directorio **`../../datos/`**.

A continuación, el programa presenta un listado de los alumnos del grupo cuyo contenido difiere al presentado por el programa desarrollado en el trabajo obligatorio ya que ahora los alumnos se agrupan según su calificación (los calificados con matrícula de honor, con sobresaliente, con notable, con aprobado, con suspenso y con no presentado). Los alumnos de cada uno de estos subgrupos se presentan ordenado alfabéticamente según sus apellidos (en caso de coincidencia de los apellidos, según su nombre). Si en un grupo de calificaciones no hay ningún alumno (en el ejemplo que se muestra a continuación no hay alumnos calificados con matrícula de honor) se debe omitir el listado del subgrupo. Conviene observar que se presentan primero los subgrupos de alumnos con mejores calificaciones y se acaba con los subgrupos con peores calificaciones.

```
 Fichero de alumnos del grupo: ../../datos/grupo.txt
 Fichero binario de notas del grupo: ../../datos/notasGrupo.bin

ALUMNOS CALIFICADOS CON SOBRESALIENTE
  1.  FONSECA CARPINTERO, PAULA
  2.  MUNDO CASTELLANO, ROSA

ALUMNOS CALIFICADOS CON NOTABLE
  1.  CARRO BADENES, ROLDAN
  2.  GARCES GARRAPIZ, CONCEPCION
  3.  ZARAGOZA ZABALZA, ANA ISABEL

ALUMNOS CALIFICADOS CON APROBADO
  1.  DOMINGUEZ GARRAPIZ, CLARA
  2.  LABRADOR LISO, RAUL
  3.  OLIVENZA ACIN, ODON
  4.  ZABALZA SANTIAGO, FERNANDO
  5.  ZAMORA HORNILLOS, JARA

ALUMNOS CALIFICADOS CON SUSPENSO
  1.  ESPINOSA OLIVITO, ALICIA
  2.  HERNANDO FONSECA, JUAN CARLOS
  3.  MELUS MELUS, GUILLERMO

ALUMNOS CALIFICADOS CON NO PRESENTADO
  1.  DOMINGO BELMEZ, DAVID
```

El desarrollo del programa se apoyará en el módulo **grupo** del trabajo obligatorio y almacenado en el área de trabajo **`biblioteca`**. El programa constará de un único fichero con el código del módulo principal del programa.

Se sugiere plantear el desarrollo del programa pedido partiendo del código del programa **listarNotas** diseñado en el trabajo obligatorio, reutilizando al máximo el código de sus funciones, haciendo en ellas las modificaciones que sean precisas, y diseñando las funciones adicionales que sean necesarias.

Presentación del trabajo y criterios de evalución

Cada alumno presentará el fichero con el código del módulo principal del programa a través de la plataforma **Moodle2** (https://moodle2.unizar.es/) antes de las 18:50. Este fichero vendrá encabezado por un comentario con el nombre y apellidos del alumno.

En este examen práctico **se valorará esencialmente que el programa pueda ser ejecutado y proporcione los resultados esperados**, es decir, un listado de los alumnos agrupados según su calificación, tal como se ha detallado anteriormente.

También se valorará el diseño del programa, la adecuada especificación de cada una de las funciones que integran el módulo principal y la legibilidad del código atendiendo a los criterios de la **"*Guía de estilo para programar en C++*"** publicada en la web de la asignatura.

Una solución del problema propuesto en el turno 1º

```
/*
 * Autor del trabajo (escriba su nombre y apellidos):
 */

/*
 * Fichero turno1.cc que almacena un programa que constituye una solución al trabajo
 * planteado en el turno 1 del examen práctico de PROG1 celebrado el 31 de enero de 2018.
 * Este programa, a partir de la información de un fichero de texto con la información
 * básica de los alumnos de un grupo y un fichero binario con sus calificaciones, cuyos
 * nombres son facilitados interactivamente por el operador, presenta por el dispositivo
 * estándar de salida un listado con las calificaciones de todos los alumnos del grupo,
 * ordenados según calificaciones decrecientes (comenzando por los alumnos con mejores
 * calificaciones y concluyendo con los que presentan peores calificaciones).
 */

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

using namespace std;

// En el módulo de biblioteca <grupo> se han definido los siguientes elementos:
//   - Constantes LIMITE_NOMBRE, LIMITE_APELLIDOS y LIMITE_GRUPO
//   - Constantes MH, SB, NO, AP, SU y NP
//   - Tipos de datos Alumno, Grupo y Notas
//   - Colecciones de funciones para trabajar los datos de los tres tipos anteriores
#include "../../../biblioteca/GestionGrupos/grupo.h"

/*
 * Pre: El valor de <codigoNumerico> representa el código numérico de un alumno
 *      universitario. El vector lasNotas[0,n-1] almacena las calificaciones de <n>
 *      alumnos en una asignatura
 * Post: Si constan en el vector lasNotas[0,n-1] las calificaciones del alumno
 *       <codigoNumerico> entonces devuelve la mejor de sus notas cuantitativas expresada
 *       en décimas de punto (i.e.: un valor entre 0 y 100) o un valor negativo igual a
 *       NO_PRESENTADO si no se ha presentado a ninguna de las dos convocatorias.
 *       Si no constan en el vector lasNotas[0,n-1] las calificaciones del alumno
 *       <codigoNumerico> entonces devuelve un valor negativo igual a NO_ENCONTRADO
 */
int suMejorNota(const int codigoNumerico, const Notas lasNotas[], const int n) {
    // Búsqueda secuencial de las notas del alumno <codigoNumerico> en lasNotas[0,n-1]
    int i = 0;
    bool encontrado = false;
    while (!encontrado && i < n) {
        if (codigoAlumno(lasNotas[i]) == codigoNumerico) {
            encontrado = true;
        }
        else {
            i = i + 1;
        }
    }
    if (encontrado) {
        // Se han localizado las notas del alumno <codigoNumerico>
        // Determina la calificación en la primera convocatoria
        int cual, cuan;
        convocatoria1(lasNotas[i], cual, cuan);
        // Asigna a <mejorNota> la nota cuantitativa en la primera convocatoria
        int mejorNota;
```

```
        if (cual != NP) {
            mejorNota = cuan;
        }
        else {
            const int NO_PRESENTADO = -1;
            mejorNota = NO_PRESENTADO;
        }
        // ¿Cabe esperar una nota en la segunda convocatoria?
        if ((cual == SU) || (cual == NP)) {
            // Determina la calificación en la segunda convocatoria
            convocatoria2(lasNotas[i], cual, cuan);
            if (cual != NP) {
                // ¿Supera la nota de la segunda convocatoria a la nota de la primera?
                if (cuan > mejorNota) {
                    mejorNota = cuan;
                }
            }
            // Ha asignado a <mejorNota> la mejor nota cuantitiva de ambas convocatorias
        }
        // Devuelve la mejor nota cuantitiva de ambas convocatorias
        return mejorNota;
    }
    else {
        // Devuelve <NO_ENCONTRADO> al no haber localizado en lasNotas[0,n-1]
        // las notas del alumno <codigoNumerico>
        const int NO_ENCONTRADO = -999;
        return NO_ENCONTRADO;
    }
}

/*
 * Pre: --
 * Post: Ha devuelto <true> si y solo si <uno> precede alfabéticamente
 *       a <otro> dando prioridad, en primer lugar, a los apellidos, y,
 *       para apellidos iguales, al nombre
 */
bool esMenor (const Alumno uno, const Alumno otro) {
    char nombre1[LIMITE_NOMBRE], nombre2[LIMITE_NOMBRE];
    char apellidos1 [LIMITE_APELLIDOS], apellidos2[LIMITE_APELLIDOS];
    nombreCompleto(uno, nombre1, apellidos1 );
    nombreCompleto(otro, nombre2, apellidos2 );
    if (strcmp( apellidos1 , apellidos2 ) < 0) {
        // <apellidos1> precede alfabéticamente a <apellidos2>
        return true;
    }
    else if (strcmp( apellidos1 , apellidos2 ) == 0) {
        // <apellidos1> coincide con <apellidos2>
        // Los nombres son los que determinan, en este caso, el orden alfabético
        return strcmp(nombre1, nombre2) < 0;
    }
    else {
        // <apellidos1> es posterior alfabéticamente a <apellidos2>
        return false ;
    }
}

/*
 * Pre: uno = A y otro = B
 * Post: uno = B y otro = A
 */
void permutar (Alumno &uno, Alumno &otro) {
```

```
    Alumno aux = uno;
    uno = otro;
    otro = aux;
}

/*
 * Pre: n > 0, <T[0..n-1]> almacena los <n> alumnos de un grupo y <lasNotas[0..numNotas-1]>
 *      almacena sus calificaciones en una asignatura. Las notas <lasNotas[i]> pueden
 *      no corresponder al alumno <T[i]>
 * 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 cada uno corresponde a un alumno cuya
 *       nota en la asignatura es mayor o igual que la del siguiente en la tabla <T>
 */
void ordenar (Alumno T[], const int n, const Notas lasNotas [], const int numNotas) {
    // Ordenación de T[0..n-1] por el método de selección
    for (int i = 0; i < n - 1; ++i) {
        // Selecciona en <iMayor> el índice del alumno de T[i,n-1] con la mejor nota
        int iMayor = i;
        int nota_iMayor = suMejorNota(codigo(T[i]), lasNotas, numNotas);
        for (int j = i + 1; j < n; ++j) {
            int nota_j = suMejorNota(codigo(T[j]), lasNotas, numNotas);
            if (nota_j > nota_iMayor) {
                iMayor = j;
                nota_iMayor = nota_j;
            }
            else if (nota_j == nota_iMayor) {
                if (esMenor(T[j], T[iMayor])) {
                    iMayor = j;
                    nota_iMayor = nota_j;
                }
            }
        }
        // Permuta T[i] y T[iMayor]. Tras ello, en T[0..i] estarán ubicados y ordenados
        // los (i + 1) alumnos con mejores notas
        permutar(T[i], T[iMayor]);
    }
}

/*
 * Pre: <notaCualitativa> presenta uno de los siguientes valores: MH, SB, NO, AP, SU o NP
 *      que representan cualificaciones cualitativas de una asignatura
 * Post: Ha escrito dos caracteres que describen la calificación <notaCualitativa>:
 *       "MH", SB", NO", AP", SU" o "NP", respectivamente
 */
void mostrarCalificacion (const int notaCualitativa ) {
    if (notaCualitativa == MH) { cout << "MH"; }
    else if (notaCualitativa == SB) { cout << "SB"; }
    else if (notaCualitativa == NO) { cout << "NO"; }
    else if (notaCualitativa == AP) { cout << "AP"; }
    else if (notaCualitativa == SU) { cout << "SU"; }
    else if (notaCualitativa == NP) { cout << "NP"; }
}

/*
 * Pre: <codigoNumerico> define el código numérico de un alumno
 *      <lasNotas[0..n-1]> almacena las calificaciones de un grupo de alumnos
 * Post: Si el valor de <codigoNumerico> corresponde con uno de los alumnos del
 *       grupo entonces ha escrito por pantalla la calificación del alumno en
 *       la primera convocatoria y, si no ha logrado aprobar en ella la asignatura,
 *       la calificación en segunda convocatoria y devuelve <true>. Ejemplos
 *       de lo que puede escribir por pantalla: "NP AP 6.5" y "SB 9.2"
```

```
 *        Si el valor de <codigoNumerico> no se corresponde con ninguno de los alumnos
 *        del grupo entonces se ha limitado a devolver <false>
 */
bool mostrarNotas (const int codigoNumerico, const Notas lasNotas [], const int n) {
    int i = 0;
    bool encontrado = false;
    while (!encontrado && i < n) {
        if (codigoAlumno(lasNotas[i]) == codigoNumerico) {
            encontrado = true;
        }
        else {
            i = i + 1;
        }
    }
    if (encontrado) {
        // Se han localizado las calificaciones del alumno <codigoNumerico> en lasNotas[i]
        // Presenta sus calificaciones en la primera convocatoria
        int cual, cuan;
        convocatoria1 (lasNotas[i], cual, cuan);
        mostrarCalificacion (cual);
        if (cual != NP) {
            cout << fixed << right << setprecision(1) << setw(5) << cuan / 10.0 << '␣';
        }
        else {
            cout << setw(6) << '␣';
        }
        if ((cual == SU) || (cual == NP)) {
            // Presenta sus calificaciones en la segunda convocatoria
            convocatoria2 (lasNotas[i], cual, cuan);
            cout << setw(3) << '␣';
            mostrarCalificacion (cual);
            if (cual != NP) {
                cout << fixed << right << setprecision(1) << setw(5) << cuan / 10.0;
            }
        }
        // Devuelve <true> por haber localizado las notas del alumno <codigoNumerico>
        return true;
    }
    else {
        // Devuelve <false> al no haber localizado las notas del alumno <codigoNumerico>
        return false;
    }
}

/*
 * Pre: <g> gestiona los datos básicos de un grupo de alumnos y <nombreFicheroNotas> es
 *      un fichero binario que almacena las calificaciones de los alumnos del grupo
 * Post: Si no ha podido acceder al fichero <nombreFicheroNotas> entonces se ha limitado
 *       a devolver el valor <false>
 *       Si ha podido acceder al fichero <nombreFicheroNotas> entonces presenta por el
 *       dipositivo estándar de salida un listado ordenado alfabéticamente por apellidos
 *       (y, en caso de coinciencia de los apellidos, por nombres) con las calificaciones
 *       obtenidas por los alumnos del grupo a razón de un alumno por línea y ha devuelto
 *       el valor <true>. Ilustración del formato del listado presentado:
 *
 *                    LISTADO DE CALIFICACIONES DE PROGRAMACION 1
 *                    ==========================================
 *
 *        CODIGO NOMBRE      APELLIDOS                 FEBRERO SEPTIEMBRE
 *       ======== =========== ==================== ======= ==========
 *       19223455 MARIA       ABADIA DE LAIGLESIA   NP      AP 5.4
```

```
 *          5602288  JOSE MANUEL BERMUDEZ LISBOA        NO  8.4
 *                                  .   .   .
 *         75099800  PEDRO          TARRAGONA PEREZ      SU  2.2     SU 3.0
 */
bool listadoNotas (const Grupo& g, const char nombreFicheroNotas[]) {
    Notas lasNotas[LIMITE_GRUPO];
    int n;
    if (leerNotas(nombreFicheroNotas, lasNotas, n)) {
        Alumno T[LIMITE_GRUPO];
        // Almacena en T[0..numeroAlumnos-1] los <numeroAlumnos> alumnos del grupo
        int numeroAlumnos = numAlumnos(g);
        for (int i = 1; i <= numeroAlumnos; ++i) {
            T[i-1] = alumno(g, i);
        }
        // Ordena los alumnos de T[0..numeroAlumnos-1] según calificaciones decrecientes
        ordenar(T, numeroAlumnos, lasNotas, n);
        // Prsenta un listado de los alumnos de T[0..numeroAlumnos-1]
        cout << endl;
        cout << setw(12) << ' ' << "LISTADO DE CALIFICACIONES DE PROGRAMACION 1"
             << endl;
        cout << setw(12) << ' ' << "=========================================="
             << endl << endl;
        cout << left << " CODIGO " << ' ' << left << setw(LIMITE_NOMBRE)
             << "NOMBRE" << setw(LIMITE_APELLIDOS) << "APELLIDOS"
             << setw(8) << "FEBRERO    " << setw(10) << "SEPTIEMBRE" << endl;
        cout << internal << setw(8) << "========"<< ' ';
        for (int i = 1; i < LIMITE_NOMBRE - 1; ++i) {
            cout << '=';
        }
        cout << "  ";
        for (int i = 1; i < LIMITE_APELLIDOS- 1; ++i) {
            cout << '=';
        }
        cout << "  ";
        for (int i = 1; i < 8; ++i) {
            cout << '=';
        }
        cout << "    ";
        for (int i = 1; i < 10; ++i) {
            cout << '=';
        }
        cout << endl;
        for (int i = 0; i < numeroAlumnos; ++i) {
            char nombre[LIMITE_NOMBRE];
            char apellidos [LIMITE_APELLIDOS];
            nombreCompleto(T[i], nombre, apellidos );
            cout << right << setw(8) << codigo(T[i]) << ' ' << left << setw(LIMITE_NOMBRE)
                 << nombre << setw(LIMITE_APELLIDOS) << apellidos;
            mostrarNotas(codigo(T[i ]), lasNotas, n);
            cout << endl;
        }
        // Devuelve <true> tras haber presentado el listado
        return true;
    }
    else {
        // Devuelve <false> por no haber podido presentar el listado
        return false ;
    }
}

/*
```

```
 * Pre: <nombre> almacena una cadena de caracteres
 * Post: Devuelve <true> si y solo si exite un fichero de nombre <nombre> accesible
 *       para la lectura de sus datos
 */
bool esAccesible (const char nombre[]) {
    ifstream f(nombre);
    if (f.is_open ()) {
        f.close ();
        return true;
    }
    else {
        return false;
    }
}

/*
 * Post: Presenta por el dispositivo estándar de salida un listado con las calificaciones
 *       de los alumnos de un grupo a partir de la información almacenada previamente de
 *       dos ficheros almacenados en la carpeta o directorio ../../ datos cuyos nombres
 *       han de ser proporcionados interactivamente por el operador.
 *         1. Fichero de texto con la información básica de los alumnos del grupo
 *         2. Fichero binario de calificaciones de todos los alumnos del grupo
 *       El listado se presenta con los alumnos ordenados según calificaciones decrecientes,
 *       y en caso de coincidir la calificación, alfabéticamente. El formato del listado
 *       se ilustra a continuación:
 *
 *                  LISTADO DE CALIFICACIONES DE PROGRAMACION 1
 *                  ==========================================
 *
 *        CODIGO NOMBRE      APELLIDOS                FEBRERO SEPTIEMBRE
 *       ======== =========== ==================== ======= ==========
 *       19223455 MARIA       ABADIA DE LAIGLESIA  NP      AP 5.4
 *        5602288 JOSE MANUEL BERMUDEZ LISBOA      NO  8.4
 *                           . . .
 *       75099800 PEDRO       TARRAGONA PEREZ      SU  2.2  SU 3.0
 */
int main () {
    // Almacenará el nombre de un fichero de texto que almacena los datos básicos de
    // los alumnos de un grupo (sus código numéricos, sus apellidos y sus nombres)
    char ficheroAlumnos[64] = "../../datos/";
    // Almacenará el nombre de un fichero binario que almacena las calificaciones de
    // los alumnos de ese mismo grupo
    char ficheroNotas [64] = "../../datos/";
    // Definición del nombre completo del fichero de alumnos
    char nombreFichero[64];
    cout << "Fichero de alumnos del grupo: " << ficheroAlumnos;
    cin >> nombreFichero;
    strcat (ficheroAlumnos, nombreFichero);
    char linea [12];
    cin.getline (linea, 12);
    if (esAccesible(ficheroAlumnos)) {
        // Definición del nombre completo del fichero de calificaciones
        cout << "Fichero de notas del grupo: " << ficheroNotas;
        cin >> nombreFichero;
        strcat (ficheroNotas, nombreFichero);
        // Define el grupo de alumnos <g> a partir de los datos almacenados en el fichero
        // de alumnos <ficheroAlumnos>
        Grupo g;
        leerGrupo(ficheroAlumnos, g);
        // Intenta presentar un listado con las calificaciones de los alumnos del grupo <g>
        // que están almacenadas en el fichero binario de calificaciones <ficheroNotas>
```

```
        if (! listadoNotas (g, ficheroNotas )) {
            // Informa de que el fichero de calificaciones no es accesible
            cout << "No se ha podido acceder al fichero " << ficheroNotas << endl;
        }
    }
    else {
        cout << "No se ha podido acceder al fichero " << ficheroAlumnos << endl;
    }
    // El programa concluye normalmente
    return 0;
}
```

Una solución del problema propuesto en el turno 2º

```
/*
 * Autor del trabajo (escriba su nombre y apellidos):
 */

/*
 * Fichero turno2.cc que almacena un programa que constituye una solución al trabajo
 * planteado en el turno 2 del examen práctico de PROG1 celebrado el 31 de enero de 2018.
 * Este programa, a partir de la información de un fichero de texto con la información
 * básica de los alumnos de un grupo y un fichero binario con sus calificaciones, cuyos
 * nombres son facilitados interactivamente por el operador, presenta por el dispositivo
 * estándar de salida un listado con las calificaciones de todos los alumnos del grupo,
 * agrupados según su calificación cualitativa.
 */

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

using namespace std;

// En el módulo de biblioteca <grupo> se han definido los siguientes elementos:
//   - Constantes LIMITE_NOMBRE, LIMITE_APELLIDOS y LIMITE_GRUPO
//   - Constantes MH, SB, NO, AP, SU y NP
//   - Tipos de datos Alumno, Grupo y Notas
//   - Colecciones de funciones para trabajar los datos de los tres tipos anteriores
#include "../../../biblioteca/GestionGrupos/grupo.h"

/*
 * Pre: El valor de <codigoNumerico> representa el código numérico de un alumno
 *      universitario. El vector lasNotas[0,n-1] almacena las calificaciones de <n>
 *      alumnos en una asignatura
 * Post: Si constan en el vector lasNotas[0,n-1] las calificaciones del alumno
 *       <codigoNumerico> entonces devuelve la mejor de sus notas cuantitativas expresada
 *       en décimas de punto (i.e.: un valor entre 0 y 100) o un valor negativo igual a
 *       NO_PRESENTADO si no se ha presentado a ninguna de las dos convocatorias.
 *       Si no constan en el vector lasNotas[0,n-1] las calificaciones del alumno
 *       <codigoNumerico> entonces devuelve un valor negativo igual a NO_ENCONTRADA
 */
void suMejorNota(const int codigoNumerico, const Notas lasNotas[], const int n,
                 int& cual, int& cuan) {
    // Búsqueda secuencial de las notas del alumno <codigoNumerico> en lasNotas[0,n-1]
    int i = 0;
    bool encontrado = false;
    while (!encontrado && i < n) {
        if (codigoAlumno(lasNotas[i]) == codigoNumerico) {
            encontrado = true;
        }
        else {
            i = i + 1;
        }
    }
    if (encontrado) {
        // Se han localizado las notas del alumno <codigoNumerico>
        // Determina la calificación en la primera convocatoria
        int nota_cual, nota_cuan;
        convocatoria1(lasNotas[i], nota_cual, nota_cuan);
        // Asigna a <mejorNota> la nota cuantitativa en la primera convocatoria
        cual = nota_cual;
```

```
        if ( nota_cual != NP) {
            cuan = nota_cuan;
        }
        else {
            const int NO_PRESENTADO = -1;
            cuan = NO_PRESENTADO;
        }
        // ¿Cabe esperar una nota en la segunda convocatoria?
        if (( nota_cual == SU) || ( nota_cual == NP)) {
            // Determina la calificación en la segunda convocatoria
            convocatoria2 ( lasNotas [ i ], nota_cual , nota_cuan );
            if ( nota_cual != NP) {
                cual = nota_cual ;
                // ¿Supera la nota de la segunda convocatoria a la nota de la primera?
                if (nota_cuan > cuan) {
                    cuan = nota_cuan;
                }
            }
            // Ha asignado a <mejorNota> la mejor nota cuantitiva de ambas convocatorias
        }
    }
    else {
        // Devuelve <NO_ENCONTRADA> al no haber localizado en lasNotas[0,n-1]
        // las notas del alumno <codigoNumerico>
        const int NO_ENCONTRADA = -999;
        cual = NO_ENCONTRADA;
    }
}

/*
 * Pre: --
 * Post: Ha devuelto <true> si y solo si <uno> precede alfabéticamente
 *       a <otro> dando pripioridad, en primer lugar, a los apellidos , y,
 *       para apellidos iguales , al nombre
 */
bool esMenor (const Alumno uno, const Alumno otro) {
    char nombre1[LIMITE_NOMBRE], nombre2[LIMITE_NOMBRE];
    char apellidos1 [LIMITE_APELLIDOS], apellidos2[LIMITE_APELLIDOS];
    nombreCompleto(uno, nombre1, apellidos1 );
    nombreCompleto(otro, nombre2, apellidos2 );
    if (strcmp( apellidos1 , apellidos2 ) < 0) {
        // <apellidos1> precede alfabéticamente a <apellidos2>
        return true;
    }
    else if (strcmp( apellidos1 , apellidos2 ) == 0) {
        // <apellidos1> coincide con <apellidos2>
        // Los nombres son los que determinan, en este caso, el orden alfabético
        return strcmp(nombre1, nombre2) < 0;
    }
    else {
        // <apellidos1> es posterior alfabéticamente a <apellidos2>
        return false ;
    }
}

/*
 * Pre: uno = A y otro = B
 * Post: uno = B y otro = A
 */
void permutar (Alumno &uno, Alumno &otro) {
    Alumno aux = uno;
```

```
    uno = otro;
    otro = aux;
}

/*
 * Pre: 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 cada uno precede alfabéticamente (apellidos
 *       y, en su caso, nombres) o coincide con siguiente en la tabla <T>
 */
void ordenar (Alumno T[], const int n) {
    // Ordenación de T[0..n-1] por el método de selección
    for (int i = 0; i < n - 1; ++i) {
        // Selecciona en <iMenor> el índice del menor de los elementos de T[i,n-1]
        int iMenor = i;
        for (int j = i + 1; j < n; ++j) {
            if (esMenor(T[j], T[iMenor])) {
                iMenor = j;
            }
        }
        // Permuta T[i] y T[iMenor]. Tras ello, en T[0..i] estarán ubicados y ordenados
        // los (i + 1) menores elementos que inicialmente había en la tabla
        permutar(T[i], T[iMenor]);
    }
}

/*
 * Pre: <notaCualitativa> presenta uno de los siguientes valores: MH, SB, NO, AP, SU o NP
 *      que representan cualificaciones cualitativas de una asignatura
 * Post: Ha escrito dos caracteres que describen la calificación <notaCualitativa>:
 *       "MH", SB", NO", AP", SU" o "NP", respectivamente
 */
void mostrarCalificacion (const int notaCualitativa ) {
    if ( notaCualitativa == MH) { cout << "MH"; }
    else if ( notaCualitativa == SB) { cout << "SB"; }
    else if ( notaCualitativa == NO) { cout << "NO"; }
    else if ( notaCualitativa == AP) { cout << "AP"; }
    else if ( notaCualitativa == SU) { cout << "SU"; }
    else if ( notaCualitativa == NP) { cout << "NP"; }
}

/*
 * Pre: <codigoNumerico> definie el código numérico de un alumno
 *      <lasNotas[0..n-1]> almacena las calificaciones de un grupo de alumnos
 * Post: Si el valor de <codigoNumerico> corresponde con uno de los alumnos del
 *       grupo entonces ha escrito por pantalla la calificación del alumno en
 *       la primera convocatoria y, no si ha logrado aprobar en ella la asignatura,
 *       la calificación en segunda convocatoria y devuelve <true>. Ejemplos
 *       de lo que puede escribir por pantalla: "NP AP 6.5" y "SB 9.2"
 *       Si el valor de <codigoNumerico> no se corresponde con ninguno de los alumnos
 *       del grupo entonces se ha limitado a devolver <false>
 */
bool mostrarNotas (const int codigoNumerico, const Notas lasNotas [], const int n) {
    int i = 0;
    bool encontrado = false;
    while (!encontrado && i < n) {
        if (codigoAlumno(lasNotas[i]) == codigoNumerico) {
            encontrado = true;
        }
        else {
            i = i + 1;
```

```
        }
    }
    if (encontrado) {
        // Se han localizado las calificaciones del alumno <codigoNumerico> en lasNotas[i]
        // Presenta sus calificaciones en la primera convocatoria
        int cual, cuan;
        convocatoria1(lasNotas[i], cual, cuan);
        mostrarCalificacion(cual);
        if (cual != NP) {
            cout << fixed << right << setprecision(1) << setw(5) << cuan / 10.0 << ' ';
        }
        else {
            cout << setw(6) << ' ';
        }
        if ((cual == SU) || (cual == NP)) {
            // Presenta sus calificaciones en la segunda convocatoria
            convocatoria2(lasNotas[i], cual, cuan);
            cout << setw(3) << ' ';
            mostrarCalificacion(cual);
            if (cual != NP) {
                cout << fixed << right << setprecision(1) << setw(5) << cuan / 10.0;
            }
        }
        // Devuelve <true> por haber localizado las notas del alumno <codigoNumerico>
        return true;
    }
    else {
        // Devuelve <false> al no haber localizado las notas del alumno <codigoNumerico>
        return false;
    }
}

/*
 * Pre: n > 0 y T[0..n-1] almacena los datos de <n> alumnos cuya calificación es <nota>
 *      y están ordenados alfabéticamente.
 * Post: Ha presentado por el dipositivo estándar de salida un listado con los nombres
 *       y apellidos, ordenados alfabéticamente, de los alumnos almacenados en T[0..n-1],
 *       informando previamente de su calificación cualitativa común. Ejemplo:
 *
 *      ALUMNOS CALIFICADOS CON APROBADO
 *        1. DOMINGUEZ GARRAPIZ, CLARA
 *                 . . .
 *        5. ZAMORA HORNILLOS, JARA
 */
void mostrarListado (const Alumno T[], const int n, const char nota[]) {
    cout << "Alumnos calificados con " << nota << endl;
    for (int i = 0; i < n; ++i) {
        char nombre[LIMITE_NOMBRE];
        char apellidos[LIMITE_APELLIDOS];
        nombreCompleto(T[i], nombre, apellidos);
        cout << setw(3) << i + 1 << ". " << apellidos << ", " << nombre << endl;
    }
    cout << endl;
}

/*
 * Pre: <g> gestiona los datos básicos de un grupo de alumnos y <nombreFicheroNotas> es
 *      un fichero binario que almacena las calificaciones de los alumnos del grupo
 * Post: Si no ha podido acceder al fichero <nombreFicheroNotas> entonces se ha limitado
 *       a devolver el valor <false>
 *       Si ha podido acceder al fichero <nombreFicheroNotas> entonces presenta por el
```

```
 *          dipositivo estándar de salida un listado de calificaciones de alumnos agrupados
 *          en bloques según su calificación cualitativa y, dentro de cada bloque, ordenados
 *          alfabéticamente por apellidos y, en caso de igualdad, por nombre. Ejemplo:
 *
 *        ALUMNOS CALIFICADOS CON SOBRESALIENTE
 *          1. FONSECA CARPINTERO, PAULA
 *          2. MUNDO CASTELLANO, ROSA
 *        ALUMNOS CALIFICADOS CON NOTABLE
 *          1. CARRO BADENES, ROLDAN
 *          2. GARCES GARRAPIZ, CONCEPCION
 *          3. ZARAGOZA ZABALZA, ANA ISABEL
 *        ALUMNOS CALIFICADOS CON APROBADO
 *          1. DOMINGUEZ GARRAPIZ, CLARA
 *                . . .
 *          5. ZAMORA HORNILLOS, JARA
 *        ALUMNOS CALIFICADOS CON SUSPENSO
 *          1. ESPINOSA OLIVITO, ALICIA
 *                . . .
 *          3. MELUS MELUS, GUILLERMO
 *        ALUMNOS CALIFICADOS CON NO PRESENTADO
 *          1. DOMINGO BELMEZ, DAVID
 */
bool listadoNotas (const Grupo& g, const char nombreFicheroNotas[]) {
    const int NUM_NOTAS = 6;
    const int NOTAS[] = { MH, SB, NO, AP, SU, NP };
    const char NOMBRE_NOTAS[][3] = { "MH", "SB", "NO", "AP", "SU", "NP" };
    int numeroAlumnos = numAlumnos(g);
    Notas lasNotas[LIMITE_GRUPO];
    int numNotas;
    if (leerNotas(nombreFicheroNotas, lasNotas, numNotas)) {
        for (int nota = 0; nota < NUM_NOTAS; ++nota) {
            Alumno T[LIMITE_GRUPO];
            // Almacena en T[0..numeroAlumnos-1] los alumnos del grupo
            int cuenta = 0;
            for (int i = 1; i <= numeroAlumnos; ++i) {
                int cual, cuan;
                Alumno nuevo = alumno(g, i);
                suMejorNota(codigo(nuevo), lasNotas, numNotas, cual, cuan);
                if (cual == NOTAS[nota]) {
                    T[cuenta] = nuevo;
                    cuenta = cuenta + 1;
                }
            }
            // cuenta = numeroAlumnos
            if (cuenta > 0) {
                // Ordena alfabéticamente de los alumnos de T[0..cuenta-1]
                ordenar(T, cuenta);
                // Presenta por el dispositivo estándar de salida un listado
                // de los alumnos de T[0..cuenta-1]
                mostrarListado(T, cuenta, NOMBRE_NOTAS[nota]);
            }
        }
        return true;
    }
    else {
        return false;
    }
}

/*
 * Pre: <nombre> almacena una cadena de caracteres
```

```
 * Post: Devuelve <true> si y solo si exite un fichero de nombre <nombre> accesible
 *       para la lectura de sus datos
 */
bool esAccesible (const char nombre[]) {
    ifstream f(nombre);
    if (f.is_open ()) {
        f.close ();
        return true;
    }
    else {
        return false;
    }
}

/*
 * Post: Presenta por el dispositivo estándar de salida un listado con las calificaciones
 *       de los alumnos de un grupo a partir de la información almacenada previamente de
 *       dos ficheros almacenados en la carpeta o directorio ../../ datos cuyos nombres
 *       han de ser proporcionados interactivamente por el operador.
 *         1. Fichero de texto con la información básica de los alumnos del grupo
 *         2. Fichero binario de calificaciones de todos los alumnos del grupo
 *       El listado se presenta por bloques. En cada bloque figuran ordenados
 *       alfabéticamente los alumno con una misma calificación cualitatva . Los bloques
 *       se ordenan de mayor calificación (MH, SB, ..) a menor calificación (.., AP, SU,
 *       NP). Un bloque sin ningún alumno se omite en el listado : Ejemplo de listado :
 *
 *      ALUMNOS CALIFICADOS CON SOBRESALIENTE
 *        1. FONSECA CARPINTERO, PAULA
 *        2. MUNDO CASTELLANO, ROSA
 *      ALUMNOS CALIFICADOS CON NOTABLE
 *        1. CARRO BADENES, ROLDAN
 *        2. GARCES GARRAPIZ, CONCEPCION
 *        3. ZARAGOZA ZABALZA, ANA ISABEL
 *      ALUMNOS CALIFICADOS CON APROBADO
 *        1. DOMINGUEZ GARRAPIZ, CLARA
 *                . . .
 *        5. ZAMORA HORNILLOS, JARA
 *      ALUMNOS CALIFICADOS CON SUSPENSO
 *        1. ESPINOSA OLIVITO, ALICIA
 *                . . .
 *        3. MELUS MELUS, GUILLERMO
 *      ALUMNOS CALIFICADOS CON NO PRESENTADO
 *        1. DOMINGO BELMEZ, DAVID
 */
int main () {
    // Almacenará el nombre de un fichero de texto que almacena los datos básicos de
    // los alumnos de un grupo (sus código numéricos, sus apellidos y sus nombres)
    char ficheroAlumnos[64] = " ../../ datos/";

    // Almacenará el nombre de un fichero binario que almacena las calificaciones de
    // los alumnos de ese mismo grupo
    char ficheroNotas [64] = " ../../ datos/";

    // Definición del nombre completo del fichero de alumnos
    char nombreFichero[64];
    cout << "Fichero de alumnos del grupo: " << ficheroAlumnos;
    cin >> nombreFichero;
    strcat (ficheroAlumnos, nombreFichero);
    char linea [12];
    cin.getline ( linea , 12);
    if (esAccesible(ficheroAlumnos)) {
```

```
        // Definición del nombre completo del fichero de calificaciones
        cout << "Fichero de notas del grupo: " << ficheroNotas;
        cin >> nombreFichero;
        strcat ( ficheroNotas , nombreFichero);
        cout << endl;

        // Define el grupo de alumnos <g> a partir de los datos almacenados en el fichero
        // de alumnos <ficheroAlumnos>
        Grupo g;
        leerGrupo(ficheroAlumnos, g);

        // Intenta presentar un listado con las calificaciones de los alumnos del grupo <g>
        // que están almacenadas en el fichero binario de calificaciones <ficheroNotas>
        if (! listadoNotas (g, ficheroNotas )) {
            // Informa de que el fichero de calificaciones no es accesible
            cout << "No se ha podido acceder al fichero " << ficheroNotas << endl;
        }
    }
    else {
        cout << "No se ha podido acceder al fichero " << ficheroAlumnos << endl;
    }
    // El programa concluye normalmente
    return 0;
}
```