Escuela de Ingeniería y Arquitectura Universidad Zaragoza

Programación 1

Examen práctico. 7 de febrero de 2019

Turno 1.º 15:15 a 16:45

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

Se ha de diseñar un programa cuya temática es la misma que la de los programas desarrollados en la práctica 6 y el trabajo obligatorio, cuyos enunciados están disponibles en la sección «Material docente» de la página web de la asignatura[1].

El programa pregunta al usuario por el nombre de un fichero de usos del sistema Bizi Zaragoza, que sigue la sintaxis establecida en el enunciado de la práctica 6 y que debe estar previamente ubicado en el directorio «`../../Datos/Bizi/`», relativo al directorio del proyecto en el que se desarrolle el programa. El usuario escribe simplemente su nombre, y es responsabilidad del programa realizar las acciones necesarias para acceder al mismo en la ubicación especificada.

A continuación, el programa solicita la introducción por el teclado de un identificador de usuario del sistema Bizi Zaragoza para, seguidamente, presentar un mensaje indicando el nombre de la estación más utilizada por el usuario cuyo identificador ha sido introducido y su número de usos, según el contenido del fichero de usos especificado por el usuario y el del fichero «`../../Datos/Bizi/estaciones.csv`».

Se muestran a continuación varios ejemplos de ejecución independientes del programa solicitado:

```
Escriba el nombre de un fichero de usos del sistema Bizi: usos-16.csv
Introduzca el identificador de un usuario: 84318

La estación más utilizada por el usuario 84318 es Plaza Mariano Arregui, con
118 usos.
```

```
Escriba el nombre de un fichero de usos del sistema Bizi: usos-17.csv
Introduzca el identificador de un usuario: 84318

La estación más utilizada por el usuario 84318 es Plaza Mariano Arregui, con
110 usos.
```

```
Escriba el nombre de un fichero de usos del sistema Bizi: usos-16.csv
Introduzca el identificador de un usuario: 90245

La estación más utilizada por el usuario 90245 es Pº de la Ribera - Puente de
Santiago, con 20 usos.
```

```
Escriba el nombre de un fichero de usos del sistema Bizi: usos-17.csv
Introduzca el identificador de un usuario: 90245

La estación más utilizada por el usuario 90245 es Pº de la Ribera - Puente de
Santiago, con 7 usos.
```

[1] http://webdiis.unizar.es/asignaturas/PROG1/?page_id=18

Cuando el usuario cuyo identificador se ha introducido no haya hecho uso del sistema Bizi, según el contenido del fichero de usos seleccionado, se indicará dicha circunstancia:

```
Escriba el nombre de un fichero de usos del sistema Bizi: usos-16.csv
Introduzca el identificador de un usuario: 12345

No hay usos registrados del usuario 12345.
```

Cuando se introduzca el nombre de un fichero de usos que no exista, el programa mostrará un mensaje de error en la pantalla:

```
Escriba el nombre de un fichero de usos del sistema Bizi: usos-19.csv
Introduzca el identificador de un usuario: 41867

No se ha podido leer del fichero "../../Datos/Bizi/usos-19.csv".
```

El desarrollo del programa se apoyará en los módulos desarrollados para la práctica 6 y el trabajo obligatorio en un módulo principal, denominado «`turno1.cpp`» que contenga la función `main` del programa solicitado. Se sugiere plantear el desarrollo del programa pedido partiendo del código del proyecto «`Estaciones`» de la práctica 6, reutilizando al máximo el código de sus funciones y haciendo en ellas las modificaciones que sean precisas. En cualquier caso, las funciones ubicadas inicialmente en otros módulos que vayan a ser modificadas para resolver este problema, deberán moverse al fichero «`turno1.cpp`». Las funciones adicionales que sean necesarias para resolver el problema también deberán ubicarse en el fichero «`turno1.cpp`». La ubicación relativa al fichero «`turno1.cpp`» de los ficheros de interfaz e implementación de los módulos reutilizados será la misma que la establecida para la práctica 6 y el trabajo obligatorio.

Presentación del trabajo y criterios de evaluación

Cada estudiante presentará, antes de hora que se indique al comienzo del examen, a través de la plataforma Moodle[2] el fichero con el código del módulo principal del programa, «`turno1.cpp`», junto con todos aquellos ficheros de interfaz e implementación que sean necesarios para su correcta compilación. El contenido de cada uno de los ficheros vendrá encabezado por un comentario con el nombre y apellidos del estudiante. Los ficheros deberán subirse a Moodle como ficheros individuales, sin comprimir.

En este examen práctico **se valorará esencialmente que el programa pueda ser compilado sin errores y que, al ser ejecutado, proporcione los resultados esperados.**

También se valorará la eficiencia de la solución presentada. En este sentido, debe tenerse en cuenta que, pese a que la ordenación completa de un conjunto de datos permite hallar fácilmente su máximo, esta no es la solución más eficiente a este problema.

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.

[2] https://moodle2.unizar.es/

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

Programación 1

Examen práctico. 7 de febrero de 2019

Turno 2.º 17:15 a 18:45

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

Se ha de diseñar un programa cuya temática es la misma que la de los programas desarrollados en la práctica 6 y el trabajo obligatorio, cuyos enunciados están disponibles en la sección «Material docente» de la página web de la asignatura[1].

El programa pregunta al usuario por el nombre de un fichero de usos del sistema Bizi Zaragoza, que sigue la sintaxis establecida en el enunciado de la práctica 6 y que debe estar previamente ubicado en el directorio «`../../Datos/Bizi/`», relativo al directorio del proyecto en el que se desarrolle el programa. El usuario escribe simplemente su nombre, y es responsabilidad del programa realizar las acciones necesarias para acceder al mismo en la ubicación especificada.

A continuación, el programa solicita la introducción por el teclado de un identificador de estación Bizi para, seguidamente, presentar un mensaje indicando el identificador y nombre de la estación introducida por el usuario y el identificador y nombre de la estación más utilizada en los trayectos que parten de o terminan en la estación cuyo identificador fue introducido, así como ese número de trayectos, según el contenido del fichero de usos especificado por el usuario y el del fichero «`../../Datos/Bizi/estaciones.csv`».

Se muestran a continuación varios ejemplos de ejecución independientes del programa solicitado:

```
Escriba el nombre de un fichero de usos del sistema Bizi: usos-16.csv
Introduzca el identificador de la estación: 67

La estación más usada conjuntamente con la estación 67 (Avda. G. Gómez de
Avellaneda - C/ Clara Campoamor) es la 12 (Pasarela del Voluntariado - Avda.
Almozara) con 2597 trayectos.
```

```
Escriba el nombre de un fichero de usos del sistema Bizi: usos-17.csv
Introduzca el identificador de la estación: 67

La estación más usada conjuntamente con la estación 67 (Avda. G. Gómez de
Avellaneda - C/ Clara Campoamor) es la 12 (Pasarela del Voluntariado - Avda.
Almozara) con 2620 trayectos.
```

```
Escriba el nombre de un fichero de usos del sistema Bizi: usos-17.csv
Introduzca el identificador de la estación: 1

La estación más usada conjuntamente con la estación 1 (Avda. Pablo Ruiz
Picasso - Torre del Agua) es la 1 (Avda. Pablo Ruiz Picasso - Torre del Agua)
con 434 trayectos.
```

Cuando el usuario introduzca el identificador de una estación inexistente, indicará dicha circunstancia escribiendo un mensaje de error en la pantalla:

[1] http://webdiis.unizar.es/asignaturas/PROG1/?page_id=18

```
Escriba el nombre de un fichero de usos del sistema Bizi: usos-17.csv
Introduzca el identificador de la estación: 131

No existe ninguna estación con identificador 131.
```

```
Escriba el nombre de un fichero de usos del sistema Bizi: usos-17.csv
Introduzca el identificador de la estación: 0

No existe ninguna estación con identificador 0.
```

Cuando se introduzca el nombre de un fichero de usos que no exista, el programa mostrará un mensaje de error en la pantalla:

```
Escriba el nombre de un fichero de usos del sistema Bizi: usos-19.csv
Introduzca el identificador de la estación: 67

No se ha podido leer del fichero "../../Datos/Bizi/usos-19.csv".
```

El desarrollo del programa se apoyará en los módulos desarrollados para la práctica 6 y el trabajo obligatorio y en un módulo principal, denominado «`turno2.cpp`» que contenga la función `main` del programa solicitado. Se sugiere plantear el desarrollo del programa pedido partiendo del código del proyecto «`Estaciones`» de la práctica 6, reutilizando al máximo el código de sus funciones y haciendo en ellas las modificaciones que sean precisas. En cualquier caso, las funciones ubicadas inicialmente en otros módulos que vayan a ser modificadas para resolver este problema, deberán moverse al fichero «`turno2.cpp`». Las funciones adicionales que sean necesarias para resolver el problema también deberán ubicarse en el fichero «`turno2.cpp`». La ubicación relativa al fichero «`turno2.cpp`» de los ficheros de interfaz e implementación de los módulos reutilizados será la misma que la establecida para la práctica 6 y el trabajo obligatorio.

Presentación del trabajo y criterios de evaluación

Cada estudiante presentará, antes de hora que se indique al comienzo del examen, a través de la plataforma Moodle[2] el fichero con el código del módulo principal del programa, «`turno2.cpp`», junto con todos aquellos ficheros de interfaz e implementación que sean necesarios para su correcta compilación. El contenido de cada uno de los ficheros vendrá encabezado por un comentario con el nombre y apellidos del estudiante. Los ficheros deberán subirse a Moodle como ficheros individuales, sin comprimir.

En este examen práctico **se valorará esencialmente que el programa pueda ser compilado sin errores y que, al ser ejecutado, proporcione los resultados esperados.**

También se valorará la eficiencia de la solución presentada. En este sentido, debe tenerse en cuenta que, pese a que la ordenación completa de un conjunto de datos permite hallar fácilmente su máximo, esta no es la solución más eficiente a este problema.

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.

[2] https://moodle2.unizar.es/

Soluciones

Se presentan a continuación soluciones parciales a cada uno de los problemas planteados en el turno 1.º y 2.º del examen práctico. Solo se incluyen en ellas sus respectivas funciones `main()` y las funciones que, previsiblemente, se han modificado para resolver los problemas solicitados. El código correspondiente a las funciones cuya implementación no aparece en estas soluciones debería estar en los módulos `bizi-usos`, `bizi-usuarios`, `bizi-estaciones` o cualquier otro módulo adicional que el estudiante haya decidido utilizar al resolver la práctica 6 y el trabajo obligatorio.

Posible solución al problema del turno 1.º

```
#include <iostream>
#include <fstream>
#include "../../Biblioteca/bizi-estacion.h"
#include "../../Biblioteca/bizi-usos.h"

using namespace std;

const int MAX_LONG_NOMBRE_FICHERO = 200;
const char NOMBRE_FICHERO_ESTACIONES[] = "../../Datos/Bizi/estaciones.csv";
const int MAX_LONG_LINEA = 2500;

/*
 * Pre:  El vector «estaciones» tiene «NUM_ESTACIONES» componentes.
 * Post: Ha devuelto el índice de la primera componente del vector
 *       «estaciones» cuyo número de usos es mayor o igual que el del resto
 *       de las componentes.
 */
int buscarMaximoUso(const Estacion estaciones[]) {
    int indMax = 0;
    for (int i = 1; i < NUM_ESTACIONES; i++) {
        if (estaciones[i].numUsos > estaciones[indMax].numUsos) {
            indMax = i;
        }
    }
    return indMax;
}
```

```
/*
 * Pre:  La cadena de caracteres «rutaFichero» representa la ruta de acceso y
 *       el nombre de un fichero de texto con el formato de usos del sistema
 *       Bizi establecido en el enunciado. El vector «estaciones» tiene
 *       «NUM_ESTACIONES» componentes donde se ha almacenado información
 *       sobre estaciones Bizi de forma tal que en cada componente «i» se
 *       almacena la información de la estación de identificador «i» + 1 (es
 *       decir, en estaciones[0] se encuentra almacenada la información de la
 *       estación con identificador 1, en estaciones[1] se encuentra
 *       almacenada la estación con identificador 2 y así sucesivamente). La
 *       información sobre el número de usos de cada estación es 0 en todas
 *       ellas.
 * Post: Si no ha habido problemas de lectura del fichero «rutaFichero», la
 *       función ha devuelto «true» y ha actualizado el número de usos de
 *       cada estación del vector «estaciones» por parte del usuario de
 *       identificador «idUsuario» de acuerdo con el contenido del fichero
 *       «rutaFichero», contabilizando como un uso tanto la retirada como la
 *       devolución de bicicletas. En caso contrario, se ha limitado a
 *       devolver «false».
 */
bool contarUsosEstacionesDeUsuario(const char rutaFichero[],
        const int idUsuario, Estacion estaciones[]) {
    ifstream f(rutaFichero);
    if (f.is_open()) {
        f.ignore(MAX_LONG_LINEA, '\n');
        UsoBizi uso;
        leerUso(f, uso);
        while (!f.eof()) {
            if (uso.idUsuario == idUsuario) {
                if (esIdEstacionCorrecto(uso.estacionRetirada)) {
                    estaciones[uso.estacionRetirada - 1].numUsos++;
                }
                if (esIdEstacionCorrecto(uso.estacionDevolucion)) {
                    estaciones[uso.estacionDevolucion - 1].numUsos++;
                }
            }
            leerUso(f, uso);
        }
        f.close();
        return true;
    }
    else {
        return false;
    }
```

```
}

/*
 * Pre:  Existe un fichero de nombre «NOMBRE_FICHERO_ESTACIONES», con la
 *       sintaxis del fichero de estaciones reflejada en el enunciado de la
 *       práctica 6, que contiene los datos de «NUM_ESTACIONES» estaciones
 *       con identificadores numéricos consecutivos entre el 1 y el
 *       «NUM_ESTACIONES».
 * Post: Ha solicitado al usuario el nombre de un fichero de usos del sistema
 *       Bizi, que responde a la sintaxis de los ficheros de usos establecida
 *       en enunciado de la práctica 6 y que debe estar previamente ubicado
 *       en el directorio «../../Datos/Bizi/»,  relativo al directorio donde
 *       se está ejecutando el programa. A continuación, ha solicitado la
 *       introducción por el teclado de un identificador de usuario del
 *       sistema Bizi Zaragoza para, seguidamente, presentar un mensaje
 *       indicando el nombre de la estación más utilizada por el usuario cuyo
 *       identificador ha sido introducido y su número de usos, según el
 *       contenido del fichero de usos especificado por el usuario y el del
 *       fichero «NOMBRE_FICHERO_ESTACIONES». Cuando el usuario cuyo
 *       identificador se ha introducido no haya hecho uso del sistema Bizi
 *       según el contenido del fichero de usos seleccionado, se ha
 *       indicado dicha circunstancia. Cuando se ha introducido el nombre de
 *       un fichero de usos que no existe, el programa ha mostrado un mensaje
 *       de error en la pantalla.
 */
int main() {
    Estacion estaciones[NUM_ESTACIONES];
    if (leerEstaciones(NOMBRE_FICHERO_ESTACIONES, estaciones)) {
        cout << "Escriba␣el␣nombre␣de␣un␣fichero␣de␣usos␣del␣sistema␣Bizi:␣"
             << flush;
        char nombreFichero[MAX_LONG_NOMBRE_FICHERO];
        cin.getline(nombreFichero, MAX_LONG_NOMBRE_FICHERO);
        char rutaRelativa[MAX_LONG_NOMBRE_FICHERO];
        nombreFicheroRelativo(nombreFichero, rutaRelativa);

        cout << "Introduzca␣el␣identificador␣de␣un␣usuario:␣" << flush;
        int idUsuario;
        cin >> idUsuario;

        if (contarUsosEstacionesDeUsuario(rutaRelativa, idUsuario,
                                          estaciones)) {
            int indMax = buscarMaximoUso(estaciones);
            cout << endl;
            if (estaciones[indMax].numUsos == 0) {
```

```
                cout << "No␣hay␣usos␣registrados␣del␣usuario␣" << idUsuario
                     << "." << endl;
            }
            else {
                cout << "La␣estación␣más␣utilizada␣por␣el␣usuario␣"
                     << idUsuario << "␣es␣" << estaciones[indMax].nombre
                     << ",␣con␣" << estaciones[indMax].numUsos << "␣usos."
                     << endl;
            }
        }
        else {
            cout << "No␣se␣ha␣podido␣leer␣del␣fichero␣\"" << rutaRelativa
                 << "\"." << endl;
        }
    }
    else {
        cout << "No␣se␣ha␣podido␣leer␣del␣fichero␣\""
             << NOMBRE_FICHERO_ESTACIONES << "\"." << endl;
    }
    return 0;
}
```

Una solución del problema propuesto en el turno 2.º

```
#include <iostream>
#include <fstream>
#include "../../Biblioteca/bizi-estacion.h"
#include "../../Biblioteca/bizi-usos.h"

using namespace std;

const int MAX_LONG_NOMBRE_FICHERO = 200;
const char NOMBRE_FICHERO_ESTACIONES[] = "../../Datos/Bizi/estaciones.csv";
const int MAX_LONG_LINEA = 2500;

/*
 * Pre:  El vector «estaciones» tiene «NUM_ESTACIONES» componentes.
 * Post: Ha devuelto el índice de la primera componente del vector
 *       «estaciones» cuyo número de usos es mayor o igual que el del resto
 *       de las componentes.
 */
int buscarMaximoUso(const Estacion estaciones[]) {
    int indMax = 0;
    for (int i = 1; i < NUM_ESTACIONES; i++) {
        if (estaciones[i].numUsos > estaciones[indMax].numUsos) {
            indMax = i;
        }
    }
    return indMax;
}
```

```
/*
 * Pre:  La cadena de caracteres «rutaFichero» representa la ruta de acceso y
 *       el nombre de un fichero de texto con el formato de usos del sistema
 *       Bizi establecido en el enunciado. El vector «estaciones» tiene
 *       «NUM_ESTACIONES» componentes donde se ha almacenado información
 *       sobre estaciones Bizi de forma tal que en cada componente «i» se
 *       almacena la información de la estación de identificador «i» + 1 (es
 *       decir, en estaciones[0] se encuentra almacenada la información de la
 *       estación con identificador 1, en estaciones[1] se encuentra
 *       almacenada la estación con identificador 2 y así sucesivamente). La
 *       información sobre el número de usos de cada estación es 0 en todas
 *       ellas. «idEstación» está comprendido entre 1 y «NUM_ESTACIONES»,
 *       ambos inclusive.
 * Post: Si no ha habido problemas de lectura del fichero «rutaFichero», la
 *       función ha devuelto «true» y ha actualizado el número de usos de
 *       cada estación del vector «estaciones» desde o hasta la estación de
 *       identificador «idEstacion» de acuerdo con el contenido del fichero
 *       «rutaFichero», contabilizando como un uso tanto la retirada como la
 *       devolución de bicicletas, siempre que el otro extremo del trayecto
 *       haya sido la estación de identificador «idEstacion». En caso
 *       contrario, se ha limitado a devolver «false».
 */
bool contarUsosEstacionesDesdeEstacion(const char rutaFichero[],
        int idEstacion, Estacion estaciones[]) {
    ifstream f(rutaFichero);
    if (f.is_open()) {
        f.ignore(MAX_LONG_LINEA, '\n');
        UsoBizi uso;
        leerUso(f, uso);
        while (!f.eof()) {
            if (uso.estacionDevolucion == idEstacion
                    && esIdEstacionCorrecto(uso.estacionRetirada)) {
                estaciones[uso.estacionRetirada - 1].numUsos++;
            }
            if (uso.estacionRetirada == idEstacion
                    && esIdEstacionCorrecto(uso.estacionDevolucion)) {
                estaciones[uso.estacionDevolucion - 1].numUsos++;
            }
            leerUso(f, uso);
        }
        f.close();
        return true;
    }
    else {
```

```
        return false;
    }
}

/*
 * Pre:  Existe un fichero de nombre «NOMBRE_FICHERO_ESTACIONES», con la
 *       sintaxis del fichero de estaciones reflejada en el enunciado de la
 *       práctica 6, que contiene los datos de «NUM_ESTACIONES» estaciones
 *       con identificadores numéricos consecutivos entre el 1 y el
 *       «NUM_ESTACIONES».
 * Post: Ha solicitado al usuario el nombre de un fichero de usos del sistema
 *       Bizi, que responde a la sintaxis de los ficheros de usos establecida
 *       en enunciado de la práctica 6 y que debe estar previamente ubicado
 *       en el directorio «../../Datos/Bizi/»,  relativo al directorio donde
 *       se está ejecutando el programa. A continuación, ha solicitado la
 *       introducción por el teclado de un identificador de estación Bizi
 *       para, seguidamente, presentar un mensaje indicando el identificador
 *       y nombre de la estación introducida por el usuario y el
 *       identificador y nombre de la estación másutilizada en los trayectos
 *       que parten de o terminan en la estación cuyo identificador fue
 *       introducido, así como ese número de trayectos, según el contenido
 *       del fichero de usos especificado por el usuario y el del fichero
 *       «NOMBRE_FICHERO_ESTACIONES». Cuando el usuario ha introducido el
 *       identificador de una estación inexistente, se ha indicado dicha
 *       circunstancia escribiendo un mensaje de error en la pantalla. Cuando
 *       se ha introducido el nombre de un fichero de usos que no existe, el
 *       programa ha mostrado un mensaje de error en la pantalla.
 */
int main() {
    Estacion estaciones[NUM_ESTACIONES];
    if (leerEstaciones(NOMBRE_FICHERO_ESTACIONES, estaciones)) {
        cout << "Escriba el nombre de un fichero de usos del sistema Bizi: "
             << flush;
        char nombreFichero[MAX_LONG_NOMBRE_FICHERO];
        cin >> nombreFichero;
        char rutaRelativa[MAX_LONG_NOMBRE_FICHERO];
        nombreFicheroRelativo(nombreFichero, rutaRelativa);

        cout << "Introduzca el identificador de la estación: " << flush;
        int idEstacion;
        cin >> idEstacion;
        cout << endl;

        if (esIdEstacionCorrecto(idEstacion)) {
```

```
            if (contarUsosEstacionesDesdeEstacion(rutaRelativa, idEstacion,
                                                  estaciones)) {
                int indMax = buscarMaximoUso(estaciones);
                cout << "La␣estación␣más␣usada␣conjuntamente␣con␣la␣"
                     << "estación␣" << idEstacion << "␣("
                     << estaciones[idEstacion - 1].nombre
                     << ")␣es␣la" << endl;
                cout << estaciones[indMax].id << "␣("
                     << estaciones[indMax].nombre << ")␣con␣"
                     << estaciones[indMax].numUsos << "␣trayectos." << endl;
            }
            else {
                cout << "No␣se␣ha␣podido␣leer␣del␣fichero␣\"" << rutaRelativa
                     << "\"." << endl;
            }
        }
        else {
            cerr << "No␣existe␣ninguna␣estación␣con␣identificador␣"
                 << idEstacion << "." << endl;
        }
    }
    else {
        cout << "No␣se␣ha␣podido␣leer␣del␣fichero␣\""
             << NOMBRE_FICHERO_ESTACIONES << "\"." << endl;
    }
    return 0;
}
```