

Prácticas de Programación 1

Grado en Ingeniería Informática



Escuela de
Ingeniería y Arquitectura
Universidad Zaragoza

**Miguel Ángel Latre, Ricardo J. Rodríguez, Rafael Tolosana y
Javier Martínez**

Área de Lenguajes y Sistemas Informáticos
Departamento de Informática e Ingeniería de Sistemas



Universidad
Zaragoza

Curso 2021-22



Distribuido bajo licencia CC BY-NC-SA. ©Profesorado del DIIS
<https://creativecommons.org/licenses/by-nc-sa/4.0/deed.es>

Práctica 6: Programas C++ con registros, vectores y ficheros de texto

6.1. Objetivos de la práctica

En esta práctica se va a trabajar con datos de tipo registro, vectores de registros y ficheros de texto.

Será necesario definir tipos de datos para representar la información almacenada en unos ficheros de texto concretos, eligiendo cómo representarlos adecuadamente. También será necesario programar una colección de funciones que faciliten el trabajo con los tipos definidos y, finalmente, diseñar un conjunto de programas C++ que hagan uso de los ficheros de texto, tanto analizando su contenido, como creando otros nuevos.

Al igual que en las restantes prácticas, a la sesión asociada a esta práctica **hay que acudir con el trabajo que se describe a continuación razonablemente avanzado** y aprovecharla con un doble objetivo:

- Consultar al profesor las dudas surgidas y las dificultades que hayan impedido resolver satisfactoriamente los problemas planteados y, con su ayuda, tratar de aclarar ideas y avanzar en la resolución de los problemas surgidos.
- Presentar al profesor el trabajo realizado para que este lo pueda revisar, advertir de posibles errores y defectos y dar indicaciones sobre cómo mejorarlo y, en su caso, corregirlo.

6.2. Tecnología y herramientas

6.2.1. El portal de datos abiertos del Ayuntamiento de Zaragoza

El Ayuntamiento de Zaragoza, a través de su portal de datos abiertos¹, pone a disposición del público en general (ciudadanía, empresas y otros organismos) muchos y muy variados conjuntos de datos con el objeto de fomentar la reutilización de los mismos, aumentar la transparencia de la administración, incrementar la participación ciudadana y posibilitar el crecimiento económico en distintos sectores. Entre los conjuntos de datos publicados figuran varios relacionados con el sistema Bizi Zaragoza relativos a estaciones, usuarios y usos del sistema². En esta práctica, vamos a trabajar exclusivamente con los datos de usos de Bizi Zaragoza.

Los dos ficheros de usos del sistema Bizi publicados son ficheros de texto y responden a la misma sintaxis:

¹<https://www.zaragoza.es/sede/portal/datos-abiertos/>

²Accesibles a través de la página web <https://www.zaragoza.es/sede/servicio/catalogo/70#CSV>



```

<fichero-usos> ::= <cabecera> { <uso> }
<cabecera> ::= <BOM> "IDUsuario;RetiroDT;RetiroEstacion;AnclajeDT;AnclajeEstacion"
               <fin-línea>
<uso> ::= <IDUsuario> <separador> <RetiroDT> <separador> <RetiroEstacion> <separador>
          <AnclajeDT> <separador> <AnclajeEstacion> fin-línea
<IDUsuario> ::= literal-entero
<RetiroDT> ::= literal-cadena
<RetiroEstacion> ::= literal-entero
<AnclajeDT> ::= literal-cadena
<AnclajeEstacion> ::= literal-entero
<separador> ::= ";"
<fin-línea> ::= "\n"

```

Cada fichero comienza con una línea de cabecera a la que le sigue la información sobre los usos del sistema Bizi en un determinado periodo de tiempo. La cabecera es una línea de texto fijo, precedida por los *bytes* que conforman la denominada *marca de orden de bytes* (en inglés, *byte order mark* o *BOM*), que se utiliza en ocasiones para indicar que el fichero sigue una codificación Unicode concreta (en este caso, UTF-8). En los programas solicitados en esta práctica tenemos que tener en cuenta que esta línea está presente en los ficheros, pero podemos ignorar su contenido concreto por completo.

A continuación, aparece una secuencia de usos o utilizaciones del sistema Bizi. Estos usos figuran cada uno en una línea distinta y contienen, en este orden, información sobre el usuario al que corresponde el uso (identificado a través de un código entero), el día y la hora en la que se retiró una bicicleta por parte del usuario, el código de la estación de la que se retiró, el día y la hora en la que se devolvió la bicicleta y el código de la estación en la que se devolvió. En los problemas que nos van a ocupar, no vamos a utilizar los datos relativos a los días y horas de retirada y devolución de las bicicletas, por lo que podemos considerar estos datos como meras cadenas de caracteres.

Los dos ficheros que ofrece actualmente el portal de datos abiertos del Ayuntamiento de Zaragoza son los siguientes:

- El fichero «Usos 0ct16-Mar17 1.csv»³, con información sobre las utilizaciones del sistema Bizi desde octubre de 2016 y hasta marzo de 2017. Se encuentra también disponible con un nombre simplificado («usos-16.csv») en la carpeta «datos» del repositorio <https://github.com/prog1-eina/practica6>, comprimido también en formato ZIP debido a su tamaño. **El contenido del fichero del repositorio de la práctica ha sido modificado para eliminar una línea en la que faltaba el identificador del usuario y los datos de recogida.**
- El fichero «UsosMar17_Ago17.csv»⁴, de usos comprendidos entre marzo de 2017 y agosto de 2017. Se encuentra también disponible con un nombre simplificado («usos-17.csv») en la carpeta «datos» del repositorio de la práctica, comprimido también en formato ZIP debido a su tamaño.

Estos ficheros son ofrecidos por el portal del Ayuntamiento con la extensión «.csv» con el objetivo de que sean abiertos fácilmente por la aplicación del sistema predefinida para la gestión del formato CSV (*comma-separated values*, o valores separados por comas). Esta aplicación es, normalmente, una aplicación de gestión de hojas de cálculo (como Excel de Microsoft Office o Calc de LibreOffice). En cualquier caso, como ficheros de texto que son, la estructura de su contenido se ve mejor si se abren con un editor de texto cualquiera (entre los que puede contarse el propio editor de Visual Studio Code).



³Se puede descargar comprimido en formato ZIP desde <http://www.zaragoza.es/contenidos/bici/Usos0ct16-Mar17csv.zip>

⁴Se puede descargar comprimido en formato ZIP desde http://www.zaragoza.es/contenidos/bici/UsosMar17_Ago17.zip

6.3. Trabajo a desarrollar en esta práctica

En esta sección se propone el desarrollo de tres programas de complejidad creciente que trabajan con ficheros de usos del sistema Bizi Zaragoza. Cada uno de ellos se desarrollará en un módulo principal diferente dentro de un directorio de nombre «practica6», creado en el directorio «programacion1».

En el repositorio <https://github.com/prog1-eina/practica6> tienes un área de trabajo para esta práctica, con la estructura de directorios y ficheros necesaria para que el fichero «Makefile» y las tareas de compilación, ejecución y depuración de Visual Studio Code funcionen adecuadamente. Sin embargo, a diferencia de otras prácticas, **en esta práctica la estructura de ficheros y el fichero «Makefile» no están completos. En la tarea 3 se comenta lo que hay que completar en el fichero «Makefile».**

Puedes descargarte el área de trabajo (botón   de la web del repositorio), y descomprimirla en tu directorio «programacion1» como «practica6» (borra el sufijo «-master» que añade GitHub al preparar el fichero comprimido para su descarga).

En el directorio del área de trabajo «practica6» hay una carpeta denominada «datos» donde se encuentran los ficheros comprimidos «usos-16.zip» y «usos-17.zip». Descomprímelos para dejar en su lugar los ficheros «usos-16.csv» y «usos-17.csv», que son los ficheros de texto que utilizaremos en esta práctica (es decir, necesitarás que estos ficheros ZIP estén descomprimidos para poder trabajar con los ficheros CSV).

6.3.1. Tarea 1. Proyecto «datos-pruebas»

Los ficheros «usos-16.csv» y «usos-17.csv» tienen, en ambos casos, más de un millón de líneas. Este volumen puede suponer un problema a la hora de realizar pruebas con los programas que se piden en las dos tareas siguientes, en los que sería recomendable realizar pruebas iniciales con ficheros más pequeños, de forma que sea más sencillo controlar que los resultados producidos son los esperados y con los que el tiempo de ejecución de los programas será sensiblemente menor.

En esta primera tarea, **desarrolla un primer programa que genere ficheros para hacer pruebas más pequeños que los originales.** En concreto, el programa solicitado realizará las siguientes tareas:

Tarea 1.1 Generar, en el directorio «datos», un fichero denominado «pruebas-10.csv» cuyo contenido sean las primeras 10 líneas, incluyendo la cabecera, del fichero «usos-16.csv».

Tarea 1.2 Generar, también en el directorio «datos», un fichero denominado «pruebas-2000.csv» cuyo contenido sean las primeras 2 000 líneas, incluyendo la cabecera, del fichero «usos-17.csv».

Diseña dicho programa en el fichero denominado «1-datos-pruebas.cpp».

El programa **no tiene que ser interactivo, bastando con que realice exactamente las dos tareas enunciadas en el listado anterior sin necesidad de solicitar información adicional al usuario.** En todo caso, se informará al usuario de si se han generado los ficheros o no, en el caso de que se haya producido un error creando o abriendo alguno de los ficheros.

El programa tampoco tiene necesidad de conocer la estructura de cada línea del fichero: basta con que copie de un fichero a otro el número de líneas establecido, sin necesidad de procesar su contenido de acuerdo a ninguna regla sintáctica.

Al escribir el código del programa solicitado en esta sección, hay que tener en cuenta cómo se resuelven los nombres de ficheros o rutas de acceso a un fichero cuando se utilizan desde un programa:

- Una *ruta absoluta* es la especificación completa de la ubicación de un determinado fichero, partiendo desde la raíz del sistema de ficheros e incluyendo los nombres de todos los directorios que hay que atravesar hasta llegar al fichero. Por ejemplo, en Windows la ruta absoluta «C:\Users\Usuario\Documentos\Trabajo\Programacion1\practica6\src\1-datos-pruebas.cpp» podría especificar la ubicación del fichero de código fuente en el que tienes que escribir el programa solicitado en esta tarea, mientras que en Linux podría ser «/home/usuario/trabajo/programacion1/practica6/src/1-datos-pruebas.cpp».
- Una *ruta relativa* es una especificación parcial de la ubicación de un determinado fichero, partiendo desde el directorio especificado por una segunda ruta. Por ejemplo, la ruta «src\1-datos-pruebas.cpp», relativa al directorio «C:\Users\Usuario\Documentos\Trabajo\Programacion1\practica6» representaría la ubicación del fichero de ejemplo de Windows del punto anterior. Del mismo modo, la ruta «src/1-datos-pruebas.cpp», relativa al directorio «/home/usuario/trabajo/programacion1/practica6», representaría la ubicación del ejemplo de Linux del punto anterior.

Todo programa en ejecución está asociado a un directorio del sistema de ficheros, denominado *directorio de trabajo* o *directorio actual*, cuyo valor inicial se establece cuando se inicia el programa. Cuando el nombre de un fichero no se especifica a través de una ruta absoluta, por defecto se considera que el nombre de fichero o ruta son relativos al directorio de trabajo actual del programa.

En el caso de Visual Studio Code, cuando se ha abierto un determinado directorio a través de la opción **File > Open Folder...**, el directorio de trabajo se establece en dicho directorio abierto. El fichero «Makefile» proporcionado para esta práctica y las tareas configuradas de Visual Studio Code asumen que es así, por lo que el directorio de trabajo de los programas que ejecutes a través de dichas tareas será el directorio «practica6».

Dado el árbol de directorios que se propone en estas prácticas, para que el programa «datos - pruebas» acceda a los ficheros de usos del sistema Bizi Zaragoza podría utilizarse una ruta de acceso absoluta (por ejemplo, «Z:\programacion1\practica6\datos\usos-16.csv») o una ruta de acceso relativa al directorio de trabajo del programa cuando se ejecute («datos/usos-16.csv»).

Solo se recomienda la segunda opción (la utilización de rutas relativas), tanto en el programa que se pide en esta tarea, como en los siguientes. Las rutas absolutas son, en muchos casos, específicas de la configuración de un determinado equipo y contraproducentes para la experiencia de usuario con los programas. Si los programas que entregas utilizan rutas absolutas, para ejecutarlos tendríamos que ubicar tu programa en la ruta concreta que tú hubieras especificado en el código fuente (*cosa que no haremos*).

6.3.2. Tarea 2. Proyecto «contar-usos»

Escribe un programa que solicite al usuario el nombre de un fichero de usos del sistema Bizi Zaragoza y, a continuación, muestre un resumen del contenido del fichero en el que **se indique el número de usos totales recogido en el fichero, distinguiendo cuántos de ellos se corresponden con traslados efectivos de la bicicleta de una estación a otra y cuántos usos son circulares** (es decir, tienen como origen y destino la misma estación).

Se muestran a continuación **dos ejemplos de ejecución** del programa solicitado:

Escriba el nombre de un fichero de usos del sistema Bizi: **usos-16.csv**

Número de usos como traslado: 1010992

Número de usos circulares: 13828

Número total de usos: 1024820

Escriba el nombre de un fichero de usos del sistema Bizi: **usos-17.csv**

Número de usos como traslado: 1001208

Número de usos circulares: 15726

Número total de usos: 1016934

El usuario solo tendrá que escribir el nombre del fichero, siendo el programa el responsable de obtener el acceso al mismo en su ubicación en la carpeta «datos» del área de trabajo utilizando una ruta relativa al mismo desde el directorio de ejecución del programa (mira los pasos de desarrollo más abajo).

Para desarrollar este programa, sigue los siguientes pasos:

Paso 1. Crea y completa el fichero de implementación «pedir-nombre-fichero.cpp» correspondiente al fichero de interfaz «pedir-nombre-fichero.hpp». El objetivo de la función pedirNombreFichero que se declara en el fichero de interfaz es facilitar la labor de solicitar al usuario el nombre de un fichero (como «usos-16.csv» o «pruebas-2000.csv») y convertirlo en una ruta de acceso relativa al directorio de ejecución del proyecto solicitado en esta tarea y en la siguiente.

Si utilizas Visual Studio Code, no deberías necesitar modificar el valor de la constante RUTA_RELATIVA que se declara en el fichero de interfaz. En otro caso, puedes modificarla si es preciso.

Paso 2. El fichero «uso-bizi.hpp» define la interfaz de un módulo para trabajar con registros que representan utilizaciones del sistema Bizi Zaragoza. Completa en él la definición del tipo registro UsoBizi para que represente los siguientes datos de un uso del sistema Bizi Zaragoza:

- el identificador del usuario que utiliza la bicicleta;
- el código de la estación de la que se retira la bicicleta; y
- el código de la estación en la que se devuelve.

Nota: La definición de este tipo registro no es estrictamente necesaria para la resolución de este problema, aunque facilita las operaciones de lectura de los ficheros en los programas solicitados en esta tarea y la siguiente.

Paso 3. Crea y completa el fichero de implementación «uso-bizi.cpp» correspondiente al fichero de interfaz «uso-bizi.hpp».

Paso 4. Escribe el módulo principal del proyecto con su función main() en el fichero «2-contar-usos.cpp».

Con el objeto de facilitar la realización de pruebas, se indica que el fichero «pruebas-10.csv», si se ha generado correctamente en la tarea anterior, contiene un total de 9 usos (8 de traslado y 1 circular). Análogamente, el fichero «pruebas-2000.csv» contiene 1989 usos de traslado y 10 usos circulares, totalizando 1999 usos.

6.3.3. Tarea 3. Proyecto «usos-por-usuario»

En esta tarea se pide escribir un programa que, al igual que el del anterior, **solicite al usuario el nombre de un fichero de usos del sistema Bizi Zaragoza y, a continuación, escriba en la pantalla el número de usuarios distintos que aparecen en el fichero y un listado con los 15 usuarios que más uso hayan hecho del sistema según el contenido del fichero dado.** Para cada uno de estos usuarios indicará el número de usos entre estaciones distintas, el número de usos entre la misma estación y el número de usos totales. Este listado de 15 usuarios deberá aparecer **ordenado de mayor a menor número de usos totales**.

Se muestran a continuación dos ejemplos de ejecución del programa solicitado:

Escriba el nombre de un fichero de usos del sistema Bizi: usos-16.csv

Número usuarios distintos: 20751

Usuario	Traslados	Circular	Total
=====	=====	=====	=====
84686	903	22	925
15381	804	61	865
64463	817	6	823
69481	736	13	749
22109	615	81	696
29275	642	10	652
57637	640	7	647
76506	627	3	630
83878	598	15	613
47889	591	6	597
83793	568	13	581
89792	568	12	580
24639	237	339	576
53548	564	6	570
90372	554	5	559

Escriba el nombre de un fichero de usos del sistema Bizi: usos-17.csv

Número usuarios distintos: 20817

Usuario	Traslados	Circular	Total
=====	=====	=====	=====
15381	736	16	752
57637	740	8	748
64463	709	10	719
76506	677	6	683
30605	653	16	669
66658	627	31	658
29275	632	6	638
44550	624	2	626
91141	593	10	603
69481	585	9	594
21110	507	86	593
24639	224	357	581
24637	576	4	580
32970	559	17	576
22109	504	68	572

Como en el programa solicitado en la tarea 2, **el usuario solo tendrá que escribir el nombre del fichero, siendo el programa el responsable de obtener el acceso al mismo** en su ubicación en la carpeta «datos» del área de trabajo utilizando una ruta relativa al mismo desde el directorio de ejecución del programa.

Para desarrollar este programa, sigue los siguientes pasos:

Paso 1. La infraestructura para resolver este problema está incompleta en el repositorio de GitHub que os hemos suministrado: faltan algunos ficheros de implementación y el fichero «Makefile» no está completo.

Crea en el directorio «src» los dos ficheros que faltan, denominados «3-usos-por-usuario.cpp» y «usuario-bizi.cpp». En ellos escribirás, en los siguientes pasos, el código necesario para resolver este problema.

Paso 2. Completa el fichero «Makefile» modificando el contenido de la regla usos-por-usuario para que compile un ejecutable a partir de los ficheros objeto resultantes de los distintos módulos de los que depende este programa:

- «3-usos-por-usuario»
- «uso-bizi»
- «usuario-bizi»
- «pedir-nombre-fichero»

Añade también las reglas que faltan para que los módulos «3-usos-por-usuario» y «usuario-bizi» se compilen individualmente.

Ya hay reglas para la compilación de los módulos que escribiste en la tarea 2 («uso-bizi» y «pedir-nombre-fichero»). Puedes basarte en ellas para escribir las reglas que faltan.

Paso 3. En el fichero de interfaz «usuario-bizi.hpp», define los campos del tipo registro UsuarioBizi para que represente los siguientes datos de un usuario del sistema Bizi: su identificador de usuario, número de usos entre estaciones distintas realizadas por ese usuario y número de usos entre la misma estación.

Paso 4. Escribe el código del fichero de implementación «usuario-bizi.cpp».

A diferencia del caso del tipo registro UsoBizi, para resolver el problema que se plantea en esta tarea, la definición del tipo UsuarioBizi sí que es necesaria, ya que para determinar el número de usuarios distintos y los usos realizados por estos, va a ser necesario trabajar con vectores de registros de tipo UsuarioBizi.

Paso 5. Escribe el módulo principal del proyecto con su función main en el fichero «3-usos-por-usuario.cpp».

Se recomienda la aplicación de la metodología de diseño descendente a la hora de escribir este programa y la minimización del esfuerzo de desarrollo, haciendo un uso adecuado de las funciones que los módulos «uso-bizi» y «usuario-bizi» implementan.

Va a ser necesario trabajar con un vector de registros de tipo UsuarioBizi en el que ir recogiendo, a nivel de usuario Bizi, la información que se vaya extrayendo del fichero de usos del sistema Bizi. Es razonable descomponer el problema que tiene que resolver el programa en las siguientes funciones:

- Una función que, **dados un vector de registros de tipo UsuarioBizi y el número de componentes utilizadas, busque a un usuario en concreto, determinado por su identificador**. La función debería devolver el índice que ocupa en el vector, si dicho usuario aparece en él, o un valor negativo (por ejemplo, -1) en caso contrario. Deberías aplicar el esquema de búsqueda lineal en un vector sin garantía de éxito.

- Una función similar a la anterior que, **dados un vector de registros de tipo UsuarioBizi y el número de componentes utilizadas, busque a un usuario en concreto, determinado por su identificador**. Si el usuario está en el vector, la función debe devolver su índice. **En caso contrario, debe añadirlo al mismo** en una componente no utilizada del mismo y devolver el índice de la componente en la que lo ha añadido.
- Una función denominada obtenerUsosPorUsuario que, **dados el nombre y ruta de acceso a un fichero de usos del sistema Bizi, almacene en las primeras componentes de un vector de registros de tipo UsuarioBizi un resumen del número de usos que cada usuario presente en el fichero ha realizado**.
- Una función que **ordene el contenido del vector de registros de tipo UsuarioBizi**. Puedes aplicar el esquema de ordenación por selección directa, adaptándolo al hecho de que no va a ser necesario tener ordenado todo el vector, sino solo los 15 usuarios con más usos del sistema.
- Una función que **escriba en la pantalla los resultados**.

La descomposición anterior es una mera sugerencia, pero en todo caso, es importante descomponer el problema que se plantea y resolverlo apoyándose en funciones auxiliares.

Para facilitar la comprobación de que los resultados obtenidos por el programa, a los ejemplos de ejecución suministrados al comienzo de esta sección, se añaden los correspondientes a los ficheros «pruebas-10.csv» y «pruebas-2000.csv», generados como resultado de la tarea 1:

Escriba el nombre de un fichero de usos del sistema Bizi: pruebas-10.csv

Número usuarios distintos: 9

Usuario	Traslados	Circular	Total
=====	=====	=====	=====
88412	1	0	1
79884	1	0	1
74089	0	1	1
88192	1	0	1
53535	1	0	1
68977	1	0	1
89736	1	0	1
82747	1	0	1
82664	1	0	1

Escriba el nombre de un fichero de usos del sistema Bizi: **pruebas-2000.csv**

Número usuarios distintos: 1857

Usuario	Traslados	Circular	Total
=====	=====	=====	=====
89629	4	0	4
47794	4	0	4
15381	4	0	4
2301	3	0	3
53082	3	0	3
89258	3	0	3
91318	3	0	3
24082	3	0	3
46340	3	0	3
6224	3	0	3
40354	3	0	3
39168	3	0	3
90978	3	0	3
91141	3	0	3
81121	3	0	3

6.4. Entrega de la práctica

Antes del **lunes 27 de diciembre a las 18:00**, se deben haber subido a Moodle los siguientes ficheros:

- «uso-bizi.hpp» y «uso-bizi.cpp»
- «usuario-bizi.hpp» y «usuario-bizi.cpp»
- «pedir-nombre-fichero.cpp»
- «1-datos-pruebas.cpp»
- «2-contar-usos.cpp»
- «3-usos-por-usuario.cpp»

En esta práctica se admitirán entregas no completas, pero, en todo caso, debe tenerse en cuenta que al evaluar la misma, el programa solicitado en la tarea 3 tendrá un peso superior al 50 % en la valoración de la práctica.