

Práctica 6 y trabajo obligatorio.

Bizi Zaragoza

6.1. Objetivos de la práctica y el trabajo

El proyecto a desarrollar como parte de esta práctica y trabajo obligatorio va a ser un programa destinado a analizar los datos que ofrece el Ayuntamiento de Zaragoza sobre el sistema Bizi Zaragoza. El programa es un intérprete de órdenes o comandos que, reiteradamente, solicitará una orden al usuario y la ejecutará, ofreciendo estadísticas o informes sobre las utilizaciones, usuarios o estaciones de Bizi Zaragoza, hasta que el usuario decida dar por concluida la ejecución del programa.

Este programa, al que denominaremos *bizi*, tendrá que trabajar con ficheros de texto, datos de tipo registro y vectores y matrices de registros. Será necesario definir tipos de datos para representar la información almacenada en unos ficheros de texto concretos, eligiendo cómo representarlos adecuadamente. El programa diseñado estará **integrado por varios módulos, con distintas funciones especializadas** en la resolución de problemas de pequeña entidad. Cada una de ellas estará convenientemente especificada, de modo que la lectura de estas especificaciones debe ser suficiente para comprender su comportamiento sin necesidad de consultar el código. El código de cada función debe constar de unas pocas líneas. El programa deberá ser fácilmente legible y estará adecuadamente documentado y presentado. **El diseño de este programa exige poner en práctica una buena parte de los conocimientos aprendidos a lo largo de toda la asignatura.**

En el repositorio de GitHub <https://github.com/progl-eina/trabajo-2022-23> se encuentra el material de partida para la realización de este trabajo (estructura de directorios, fichero «Makefile», ficheros de datos con los que trabajar y ficheros con otros recursos).

El trabajo será desarrollado en equipos de dos estudiantes, tal y como se indica en la sección 6.6.

En la siguiente sección se describen el origen, sintaxis y contenido de los ficheros de texto con los que se va a trabajar. En la sección 6.4 se establecen los requisitos que debe cumplir el programa solicitado y en la sección 6.5 se dan algunas pautas y guías para el diseño e implementación de la solución. Finalmente, se indica en que van a consistir los entregables del trabajo y el mecanismo de entrega.

Dado que el trabajo obligatorio se va a combinar con la práctica 6, habrá sesiones de prácticas en las que se podrá desarrollar el trabajo. Por lo tanto, y al igual que en las restantes prácticas, a la sesión asociada a esta práctica y trabajo obligatorio **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.

Antes de comenzar a realizar este trabajo, se recomienda la lectura completa del enunciado, con objeto de obtener una idea completa del conjunto de las tareas que la componen y de la carga de trabajo que puede suponer.

6.2. 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 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.

6.2.1. Ficheros de usos del sistema Bizi Zaragoza

Entre los datos publicados hay dos ficheros que reflejan las utilizaciones del sistema Bizi realizadas por sus distintos usuarios. Se trata de ficheros de texto que cumplen con la siguiente sintaxis:

```
<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 este trabajo tenemos que tener en cuenta que esta línea está presente en los ficheros, pero podemos simplemente 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

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>

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_Oct16-Mar17_1.csv»³, con información sobre las utilidades 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/trabajo-2022-23>, comprimido en formato ZIP debido a su tamaño. **El contenido del fichero del repositorio del trabajo 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 del trabajo, 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).

6.2.2. Fichero de usuarios de Bizi Zaragoza

Los dos ficheros de usos del sistema Bizi Zaragoza vienen acompañados de dos ficheros de usuarios correspondientes a esos mismos dos periodos⁵. Una combinación del contenido de ambos se encuentra disponible también en la carpeta «datos» del repositorio <https://github.com/prog1-eina/trabajo-2022-23> como «usuarios.csv».

Los dos ficheros de usuarios ofrecidos por el portal de datos abiertos del Ayuntamiento de Zaragoza y el fichero combinado «usuarios.csv» con el que trabajaremos en este trabajo responden a la siguiente sintaxis:

```
<fichero-usuarios> ::= <BOM> <cabecera> { <usuario> }
<cabecera> ::= "IDUsuario;Genero;Edad" <fin-línea>
<usuario> ::= <id-usuario> <separador> [<género>] <separador> <rango-edad> <fin-línea>
<id-usuario> ::= literal-entero
<separador> ::= ";"
<fin-línea> ::= "\n"
<género> ::= "M" | "F"
<rango-edad> ::= "<=25" | "26-35" | "36-50" | "51-65" | ">65"
```

Cada fichero comienza con una línea de cabecera a la que le sigue la información sobre los usuarios del sistema Bizi en un determinado periodo de tiempo. Estos usuarios figuran cada uno en una línea

³Disponible comprimido en formato ZIP en <http://www.zaragoza.es/contenidos/bici/UsosOct16-Mar17csv.zip>

⁴Disponible comprimido en formato ZIP en <http://www.zaragoza.es/contenidos/bici/UsosMar17-Ago17.zip>

⁵«UsuariosOct_16-Mar17.csv» y «UsuariosMar17_Ag17.csv», que pueden obtenerse de <https://www.zaragoza.es/sede/portal/datos-abiertos/servicio/catalogo/70#CSV>

distinta y contienen información sobre su identificador, su género (masculino –‘M’– o femenino –‘F’–) y su rango de edad.

Lamentablemente, en algunos casos no hay información sobre el género del usuario. Por ello, en la definición de la regla <usuario>, la aparición de <género> se ha marcado como opcional.

Los datos del fichero «usuarios.csv» **están ordenados por identificadores de usuario crecientes**.

6.2.3. Fichero de estaciones del sistema Bizi Zaragoza

A través del portal de datos abiertos del Ayuntamiento de Zaragoza se puede obtener también un fichero con información sobre las estaciones del sistema Bizi⁶. Se trata de un fichero de texto que responde a la siguiente sintaxis:

```
<fichero-estaciones> ::= <cabecera> { <estación> }
<cabecera> ::= <BOM> "id;about;title;estado;estadoEstacion;address;tipoEquipamiento;"
               "bicisDisponibles;anclajesDisponibles;geometry;lastUpdated;"
               "description;descripcion;icon" <fin-línea>
<estación> ::= <url-id> <separador> <url-acerca-de> <separador> <nombre>
               <separador> <estado> <separador> <descr-estado> <separador>
               <tp-equipamiento> <separador> <bicis-disponibles> <separador>
               <dirección> <separador> <anclajes-disponibles> <separador>
               <coordenadas> <separador> <última-actualización> <separador>
               <descripción> <separador> <descripción-estado> <separador> <url-icone>
               <fin-línea>
<fin-línea> ::= "\n"
<separador> ::= ";"
<url-id> ::= ""https://www.zaragoza.es/sede/servicio/urbanismo-infraestructuras/"
            "estacion-bicicleta/" <id-estación> ""
<id-estación> ::= literal-entero
<url-acerca-de> ::= "" literal-cadena ""
<nombre> ::= "" literal-cadena ""
<estado> ::= "" literal-cadena ""
<descr-estado> ::= "" literal-cadena ""
<dirección> ::= "" literal-cadena ""
<tp-equipamiento> ::= "" literal-cadena ""
<bicis-disponibles> ::= "" literal-entero ""
<anclajes-disponibles> ::= "" literal-entero ""
<coordenadas> ::= "" literal-cadena ""
<última-actualización> ::= literal-cadena
<descripción-estado> ::= "" literal-cadena ""
<descripción> ::= "" literal-cadena ""
<url-icone> ::= "" literal-cadena ""
```

Cada fichero comienza con una línea de cabecera a la que le sigue la información sobre las estaciones del sistema Bizi, a razón de una estación por línea. De toda la información que figura sobre cada estación en el fichero, solo vamos a utilizar su identificador numérico (que figura al final de la primera URL de

⁶El fichero es accesible a través del siguiente enlace: <https://www.zaragoza.es/sede/servicio/urbanismo-infraestructuras/estacion-bicicleta.csv?rows=500>.

cada línea) y el nombre de la estación (el tercer dato de cada línea del fichero). Nótese como tanto la URL donde figura el identificador numérico como el nombre de la estación están rodeados por comillas.

El sistema Bizi tiene en estos momentos 130 estaciones con identificadores numéricos consecutivos entre el 1 y el 130. Se puede comprobar fácilmente que **las estaciones no están ordenadas en el fichero** de acuerdo con ningún criterio.

Una copia del fichero de estaciones, ligeramente modificada para adaptarse a los objetivos de este trabajo, se encuentra disponible con el nombre «estaciones.csv» en el directorio «datos» del repositorio <https://github.com/progl-eina/trabajo-2022-23>.

Advertencia: En los ficheros **de usos** proporcionados por el portal de datos abiertos del Ayuntamiento de Zaragoza, en ocasiones aparecen códigos de estaciones que no se corresponden con estaciones reales del sistema (es decir, estaciones con códigos entre 1 y 130, ambos inclusive). El comportamiento del programa ante estos códigos de estación debe ser el de simplemente ignorarlos, pero sin provocar fallos en la ejecución o la obtención de resultados erróneos.

6.2.4. Sobre la disponibilidad de datos adicionales de periodos posteriores

En el momento de escribir este enunciado, estos son los dos únicos periodos para los que se han publicado datos, pero no se descarta que puedan publicarse otros posteriormente. El programa solicitado en este trabajo debería ser capaz de incluirlos sin realizar ningún cambio en su código. En todo caso, habría que modificar únicamente el contenido del fichero «res/opciones.txt» (ver sección 6.5).

6.3. Tarea previa. Creación de datos de 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 el programa que se pide en este trabajo. Sería recomendable realizar pruebas iniciales con ficheros más pequeños, de manera que sea más sencillo controlar que los resultados producidos son los esperados y con los que el tiempo de ejecución del programa será sensiblemente menor.

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

Tarea 0.1 Generar, en el directorio «datos», un fichero denominado «usos-t1.csv» cuyo contenido sean las primeras 10 líneas, incluyendo la cabecera, del fichero «usos-16.csv».

Tarea 0.2 Generar, también en el directorio «datos», un fichero denominado «usos-t2.csv» cuyo contenido sean las primeras 2 000 líneas, incluyendo la cabecera, del fichero «usos-17.csv».

Diseña dicho programa en el fichero denominado «0-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 correctamente los ficheros o no, en el caso de que se haya producido un error creando o abriendo alguno de los ficheros.

El programa solicitado en esta tarea previa 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 con 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\Programacion1\trabajo2022-23\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/programacion1/trabajo2022-23/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\Programacion1\trabajo2022-23» 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/programacion1/trabajo2022-23», 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 este trabajo y las tareas configuradas de Visual Studio Code asumen que es así, por lo que el directorio de trabajo de los programas que ejecute a través de dichas tareas será el directorio «trabajo2022-23».

Dado el árbol de directorios que se propone en este trabajo, 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\trabajo2022-23\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 el programa general correspondiente al trabajo. 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 entregáis utilizan rutas absolutas, para ejecutarlos tendríamos que ubicar vuestro programa en la ruta concreta que hubierais especificado en vuestro código fuente (*cosa que no haremos*).

6.4. Requisitos del programa bizi

6.4.1. Inicio del programa

Cuando el programa bizi comienza su ejecución, solicita al usuario que indique el conjunto de datos **de usos** con el que va a trabajar inicialmente. Como se dispone de un total de cuatro ficheros (los dos

obtenidos del Ayuntamiento de Zaragoza, «usos-16.csv» y «usos-17.csv», y los dos creados para hacer pruebas, «usos-t1.csv» y «usos-t2.csv»), bastará con que el usuario del programa elija entre las opciones «16», «17», «t1» o «t2». No obstante, el programa indicará cuales son estas opciones, de acuerdo con el siguiente modelo de interacción:

Elección de ficheros de usos y usuarios. Opciones disponibles:

16: octubre 2016 a marzo 2017

17: marzo 2017 a agosto 2017

t1: datos para pruebas (10 líneas)

t2: datos para pruebas (2000 líneas)

Introduzca una opción: **16**

El fichero "datos/usos-16.csv" existe y ha sido seleccionado.

El programa ofrecerá las opciones disponibles y solicitará una de ellas. Cuando el usuario introduzca su respuesta, el programa comprobará si existe el fichero de usos correspondiente y, si existe, informará de ha sido seleccionado.

Si el usuario introduce una opción para la que no hay ficheros, informará de que el fichero correspondiente no existe, y volverá a informar sobre las opciones disponibles y a solicitar una nueva hasta que el usuario elija una opción para la que exista un fichero de usos. Se muestra un ejemplo de esta interacción reiterada a continuación:

Elección de ficheros de usos y usuarios. Opciones disponibles:

16: octubre 2016 a marzo 2017

17: marzo 2017 a agosto 2017

t1: datos para pruebas (10 líneas)

t2: datos para pruebas (200 líneas)

Introduzca una opción: **18**

No se ha podido abrir el fichero "datos/usos-18.csv".

Elección de ficheros de usos y usuarios. Opciones disponibles:

16: octubre 2016 a marzo 2017

17: marzo 2017 a agosto 2017

t1: datos para pruebas (10 líneas)

t2: datos para pruebas (200 líneas)

Introduzca una opción: **15**

No se ha podido abrir el fichero "datos/usos-15.csv".

Elección de ficheros de usos y usuarios. Opciones disponibles:

16: octubre 2016 a marzo 2017

17: marzo 2017 a agosto 2017

t1: datos para pruebas (10 líneas)

t2: datos para pruebas (200 líneas)

Introduzca una opción: **17**

El fichero "datos/usos-17.csv" existe y ha sido seleccionado.

6.4.2. Segundo paso del programa bizi

A continuación, el programa informará de cuáles son las órdenes que acepta, mostrando la siguiente información:

ÓRDENES DISPONIBLES

=====

| | |
|-----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------|
| AYUDA: | Muestra esta pantalla de ayuda. |
| FICHERO: | Permite modificar la selección del fichero de usos a utilizar. |
| USOS: | Escribe en la pantalla el número de usos de traslado y circulares. |
| ESTADISTICAS: | Informa la distribución por edades y géneros de los usuarios. |
| USUARIO <id-usuario>: | Informa acerca del número de usos realizados por el usuario especificado. |
| MAYORES <n>: | Escribe en la pantalla el número de usuarios distintos y los <n> usuarios que más usos han hecho del sistema. |
| INFORME <nombre-fichero>: | Escribe en el fichero especificado un informe con el número de usos de las estaciones, según el fichero seleccionado. |
| DESTINOS: <nombre-fichero>: | Escribe en el fichero especificado un informe en el que, para cada estación dada, se indica la estación a la que más se ha viajado desde ella. |
| FIN: | Termina la ejecución de este programa. |

6.4.3. Órdenes disponibles

A continuación, el programa solicitará interactivamente una de las órdenes disponibles. El usuario deberá escribir una de ellas, junto con el argumento adecuado en el caso de se requiera (caso de las órdenes «USUARIO», «MAYORES», «INFORME» y «DESTINOS»), la ejecutará y volverá a pedir una nueva orden hasta que el usuario escriba la orden «FIN», en cuyo caso el programa terminará.

Será indiferente que el usuario escriba la orden en mayúsculas, minúsculas o cualquier combinación de ambas.

Se describen a continuación con más detalles en qué consisten las distintas órdenes.

Orden «AYUDA»

Cuando el usuario escribe la orden «AYUDA», se muestran en la pantalla las órdenes disponibles, tal y como se ha especificado en la sección 6.4.2:

Orden: ayuda

ÓRDENES DISPONIBLES

=====

AYUDA: Muestra esta pantalla de ayuda.
FICHERO: Permite modificar la selección del fichero de usos a utilizar.
USOS: Escribe en la pantalla el número de usos de traslado y circulares.
ESTADISTICAS: Informa la distribución por edades y géneros de los usuarios.
USUARIO <id-usuario>: Informa acerca del número de usos realizados por el usuario especificado.
MAYORES <n>: Escribe en la pantalla el número de usuarios distintos y los <n> usuarios que más usos han hecho del sistema.
INFORME <nombre-fichero>: Escribe en el fichero especificado un informe con el número de usos de las estaciones, según el fichero seleccionado.
DESTINOS: <nombre-fichero>: Escribe en el fichero especificado un informe en el que, para cada estación dada, se indica la estación a la que más se ha viajado desde ella.
FIN: Termina la ejecución de este programa.

Orden:

Orden «FICHERO»

La orden «FICHERO» permite modificar la selección del fichero de usos («usos - 16.csv», «usos - 17.csv», «usos - t1.csv», «usos - t2.csv» u otros que pudiera haber en el futuro) que el usuario seleccionó inicialmente. El programa se comportará como se especificó en la sección 6.4.1. La orden no se completará hasta que el usuario seleccione una opción para la que exista un fichero de usos.

Orden: fichero

Elección de ficheros de usos y usuarios. Opciones disponibles:

16: octubre 2016 a marzo 2017

17: marzo 2017 a agosto 2017

t1: datos para pruebas (10 líneas)

t2: datos para pruebas (2000 líneas)

Introduzca una opción: t1

El fichero "datos/usos-t1.csv" existe y ha sido seleccionado.

Orden:

Orden «USOS»

Cuando el usuario escribe esta opción, el programa muestra un resumen del contenido del contenido del último fichero de usos seleccionado hasta el momento (a través de la última ejecución de la orden «FICHERO» o, si no se ha ejecutado en ninguna ocasión, el que se solicitó al comenzar el programa). En el resumen **debe indicarse 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).

Por ejemplo, si el fichero de usos seleccionado actualmente fuese «usos-16.csv», el programa debería mostrar el siguiente resumen:

```
Orden: usos  
Fichero de usos seleccionado actualmente: datos/usos-16.csv  
Número de usos como traslado: 1010992  
Número de usos circulares: 13828  
Número total de usos: 1024820  
  
Orden:
```

Si el fichero de usos seleccionado fuese «usos-17.csv», debería mostrar:

```
Orden: usos  
Fichero de usos seleccionado actualmente: datos/usos-17.csv  
Número de usos como traslado: 1001208  
Número de usos circulares: 15726  
Número total de usos: 1016934  
  
Orden:
```

Si el fichero de usos seleccionado fuese «usos-t1.csv», debería mostrar:

```
Orden: usos  
Fichero de usos seleccionado actualmente: datos/usos-t1.csv  
Número de usos como traslado: 8  
Número de usos circulares: 1  
Número total de usos: 9  
  
Orden:
```

Y si el fichero de usos seleccionado fuese «usos-t2.csv», debería mostrar:

```
Orden: usos  
Fichero de usos seleccionado actualmente: datos/usos-t2.csv  
Número de usos como traslado: 1989  
Número de usos circulares: 10  
Número total de usos: 1999  
  
Orden:
```

Orden «ESTADÍSTICAS»

Cuando el usuario escribe esta opción, el programa muestra en la pantalla una tabla en la que se muestra la distribución de los usuarios del sistema para cada franja de edad y para cada género.

Al ejecutar esta orden, el programa deberá obtener del fichero «usuarios.csv» y mostrar por la pantalla la siguiente información:

```

Orden: estadísticas
Distribución de los usuarios
      |      M      F
-----+-----
<=25 |    1567    1306
26-35 |    2200    2214
36-50 |    4685    3857
51-65 |    3692    2326
>65   |     848     192
Orden:

```

Orden «USUARIO»

Cuando el usuario escribe esta opción, debe proporcionar además como argumento el identificador de un usuario del sistema Bizi. Si se encuentra en el fichero de usuarios, el programa informará acerca del género y el rango de edad al que pertenece el usuario con dicho identificador:

Por ejemplo, si el fichero de usos seleccionado cuando se escribe esta opción fuese «usos-16.csv», el programa mostraría por la pantalla la siguiente información (obsérvese que se ajusta el texto mostrado al género del usuario, tanto si este es conocido como si no):

```

Orden: usuario 55381
La usuaria 55381 está en el rango de edad "26-35".

Orden: usuario 16391
El usuario 16391 está en el rango de edad ">65".

Orden: usuario 91455
El/la usuario/a 91455 está en el rango de edad "<=25".

Orden: usuario 10201
El/la usuario/a 10201 no aparece en el fichero "datos/usuarios.csv".

Orden: usuario 93385
La usuaria 93385 está en el rango de edad "26-35".

Orden:

```

Orden «MAYORES»

Cuando el usuario escribe esta opción, el programa escribe en la pantalla el número de usuarios distintos que aparecen en el fichero de usos seleccionado actualmente y un listado con los n usuarios Bizi que más uso hayan hecho del sistema según el contenido del fichero de usos, siendo n un argumento que el usuario del programa proporciona junto con la orden. 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 n usuarios deberá aparecer **ordenado de mayor a menor número de usos totales**.

Se muestra a continuación un ejemplo de ejecución de la orden si el fichero de usos seleccionado actualmente fuese «usos-16.csv»:

Orden: **mayores 15**

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 |

Y, a continuación, otro ejemplo suponiendo que el fichero de usos seleccionado actualmente fuera «usos-17.csv»:

Orden: **mayores 10**

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 |

Orden «INFORME»

Cuando el usuario escribe esta opción, debe proporcionar como argumento el nombre del fichero en el que se va a escribir la información acerca de las estaciones Bizi más usadas de acuerdo con el fichero de usos que se encuentre seleccionado en ese momento, ordenadas de mayor a menor número de usos y con el formato que se muestra a continuación en los ejemplos.

Por ejemplo, si el fichero seleccionado cuando se escribe esta opción fuese «usos-16.csv», el programa debería interactuar del siguiente modo:

Orden: **informe estaciones16.txt**

Informe "estaciones16.txt" generado correctamente.

Orden:

El programa, además, habrá generado, en este caso en el directorio en el que se está ejecutando el programa, un fichero denominado «estaciones16.txt» en el que figurarán todas las estaciones Bizi ordenadas de mayor a menor número de usos según el contenido del fichero actualmente seleccionado («usos-16.csv» en el ejemplo mostrado):

| Puesto | Usos | Id | Nombre |
|--------|-------|-----|---------------------------------------------------|
| 1 | 47064 | 16 | Plaza España |
| 2 | 42306 | 67 | Avda. G. Gómez de Avellaneda - C/ Clara Campoamor |
| 3 | 40251 | 47 | Plaza San Francisco - Universidad |
| 4 | 39173 | 34 | Plaza Magdalena |
| 5 | 36264 | 10 | Pº Echegaray y Caballero - Puente de Santiago |
| ... | ... | ... | ... |
| 129 | 3625 | 82 | C/ Los Leñadores - Avda. Academia Gral. Militar |
| 130 | 3009 | 83 | Avda. Majas de Goya - Eroski |

No es responsabilidad del programa ubicar el fichero en un directorio concreto. Si el usuario escribe solo el nombre del fichero, este se creará en el directorio de ejecución del programa. Si, por el contrario, el usuario deseara que el informe se generase en el directorio «datos/», es el propio usuario el que debe indicarlo añadiendo la ruta relativa deseada al nombre del fichero.

Orden «DESTINOS»

Cuando el usuario escribe esta opción, el programa calculará, para cada estación del sistema, la estación que mayor número de veces es destino de los viajes que parten de la primera. Para ello, utilizará los datos de usos del fichero de usos seleccionado actualmente.

A continuación, el programa solicitará al usuario de forma interactiva el nombre de un fichero en el que guardar la información resultante, con el formato que se muestra posteriormente en los ejemplos de ejecución.

Si el usuario escribe el nombre de un fichero, la información se escribe en el fichero cuyo nombre ha especificado el usuario. El programa informa de si lo ha podido escribir correctamente o no.

No es responsabilidad del programa ubicar el fichero en un directorio concreto. Si el usuario escribe solo el nombre del fichero, este se creará en el directorio de ejecución del programa. Si, por el contrario, el usuario deseara que el informe se generase en el directorio «datos/», es el propio usuario el que debe indicarlo añadiendo la ruta relativa deseada al nombre del fichero.

Si el usuario opta por no escribir ningún nombre de fichero y simplemente pulsa la tecla «Entrar» para continuar, el programa escribirá el informe directamente en la pantalla, sin necesidad de crear ningún fichero.

Por ejemplo, si el fichero seleccionado cuando se escribe esta opción fuese «usos-16.csv» y el usuario del programa indicara que desea escribir el informe en el fichero «datos/destinos.txt», el programa mostraría por la pantalla la siguiente información:

Orden: **destinos**

Escriba el nombre del fichero del informe

(presione solo ENTRAR para escribirlo en la pantalla): **datos/destinos.txt**

Informe "datos/destinos.txt" creado correctamente.

Y el contenido del fichero resultante «datos/destinos.txt» sería el siguiente:

```


Viajes  Origen --> Destino
-----
210    1-Avda. Pablo Ruiz Picasso - Torre del Agua --> 88-Avda. Navarra - C/ Alagón
380    2-Avda. Ranillas - Avda. Pablo Ruiz Picasso --> 12-Pasarela del Voluntariado - Avda. Almozara
479    3-Pasarela del Voluntariado - Avda. Ranillas --> 86-Avda. Almozara - CDM Almozara
388    4-Avda. Ranillas - Puente de La Almozara --> 10-Pº Echegaray y Caballero - Puente de Santiago
...
439    129-C/ Graus - C/ Unceta --> 50-Avda. San Juan Bosco - C/ Menéndez Pelayo
361    130-C/ Corona de Aragón - C/ Lorente --> 52-Avda. Duquesa Villahermosa - C/ Franco y López

```

Si el fichero seleccionado cuando se escribe esta opción fuese «usos-17.csv» y el usuario optase por escribir el informe en la pantalla, el programa mostraría por la pantalla la siguiente información:

Orden: **destinos**

Escriba el nombre del fichero del informe

(presione solo ENTRAR para escribirlo en la pantalla): 

```

Viajes  Origen --> Destino
-----
217    1-Avda. Pablo Ruiz Picasso - Torre del Agua --> 1-Avda. Pablo Ruiz Picasso - Torre del Agua
369    2-Avda. Ranillas - Avda. Pablo Ruiz Picasso --> 12-Pasarela del Voluntariado - Avda. Almozara
436    3-Pasarela del Voluntariado - Avda. Ranillas --> 86-Avda. Almozara - CDM Almozara
391    4-Avda. Ranillas - Puente de La Almozara --> 10-Pº Echegaray y Caballero - Puente de Santiago
...
309    129-C/ Graus - C/ Unceta --> 50-Avda. San Juan Bosco - C/ Menéndez Pelayo
311    130-C/ Corona de Aragón - C/ Lorente --> 92-C/ San Juan de la Cruz - C/ Manuel Lasala

```

Por motivos de claridad, en los dos listados anteriores se han omitido los datos de las estaciones de origen de identificadores entre el 5 y el 128. El programa, cuando escriba estos listados, lo hará para todas las estaciones.

Tanto si se escribe en un fichero como si se escribe en la pantalla, el contenido del informe está ordenado por identificadores crecientes de estación origen y cada línea tiene el siguiente formato:

- número de usos entre la estación de origen y su estación de destino más utilizada;
- identificador de cada estación de origen, utilizando 3 caracteres;
- un guion de separación;
- nombre de la estación de origen;
- la cadena de 7 caracteres " **111** --> **111** ";
- identificador de la estación más utilizada como destino de la estación de origen;
- un guion de separación; y
- nombre de la estación.

Orden «FIN»

Cuando el usuario selecciona esta opción, el programa termina (y, por lo tanto, no vuelve a solicitar más órdenes):

Orden: fin

6.4.4. Órdenes no válidas

Cuando el usuario seleccione una orden no disponible, el programa deberá informar sobre el error y solicitar una nueva orden:

Orden: acabar
Orden "ACABAR" desconocida.


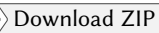
Orden:

6.4.5. Traza de ejecución

En Moodle se encuentra disponible una traza completa de una posible ejecución del programa, junto con los ficheros correspondientes a los informes que se han solicitado.

6.5. Diseño e implementación del programa

En el repositorio <https://github.com/progl-eina/trabajo-2021-22> se encuentra el material de partida para la realización de este trabajo. El repositorio cuenta 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.

El área de trabajo de puede descargar de GitHub (botón   de la web del repositorio), debiéndose descomprimir su contenido una vez concluida la descarga. Recordad borrar el sufijo «-main» que añade GitHub al preparar el fichero comprimido para su descarga.

En el repositorio hay un directorio denominado «datos» donde se encuentran los ficheros «usos-16.zip», «usos-17.zip», «usuarios.csv» y «estaciones.csv». Descomprimid los ficheros «usos-16.zip», «usos-17.zip» para obtener los ficheros «usos-16.csv» y «usos-17.csv», que tienen que quedar finalmente en el directorio «datos».

En el directorio «src» se encuentran los ficheros correspondientes a los diferentes módulos en los que estará repartido el código de este trabajo. En la sección 6.5 se da más información acerca de estos módulos.

Implementación incremental

Tal y como se ha indicado previamente, la función main debe apoyarse en un conjunto de funciones auxiliares que permitan resolver el problema de forma modular.

El programa puede comenzar invocando en primer lugar a una función que permita seleccionar el fichero de usos con el que trabajar. Esta misma función será la que se utilice en el caso de que, posteriormente, se seleccione la orden «FICHERO». El programa, a continuación, invocará a una función que muestra las órdenes disponibles, que será la misma a la que se invoque cuando se escriba la orden «AYUDA». El programa, a continuación, deberá entrar en un bucle en el que se irá solicitando una orden y se ejecutará, hasta que se escriba la orden «FIN».

De hecho, se sugiere realizar un desarrollo incremental en el que, inicialmente, el programa se limite a entrar en un bucle en el que se solicita al usuario la introducción de una orden. En una primera versión, el programa puede reconocer únicamente la orden «FIN» y considerar cualesquiera otras como órdenes incorrectas. Una vez que se haya escrito el código de esta versión inicial del programa, se haya compilado y se haya ejecutado para comprobar que este funcionamiento mínimo es correcto, se sugiere ir incorporando una a una las distintas opciones. La siguiente orden a considerar sería la orden «AYUDA», la posterior, la orden «FICHERO» y así sucesivamente. En el listado de ayuda y en este mismo enunciado, se han ordenado las distintas órdenes por lo que consideramos es complejidad creciente de la implementación de cada una de ellas.

A continuación se dan algunas pautas relativas a la implementación que consideramos más adecuada de cada una de las órdenes. Estas pautas son mucho más detalladas en lo que respecta a las órdenes que hubieran formado parte de una práctica 6 independiente («USOS» y «MAYORES»).

Orden «AYUDA». El fichero de texto «ayuda.txt» del directorio «res»⁷ contiene el texto del menú de ayuda con las órdenes disponibles. En lugar de escribir el código de la función que implemente esta orden como un conjunto *ad hoc* de instrucciones de escritura en pantalla de literales de tipo cadena que en conjunto terminan escribiendo la ayuda en la pantalla, lo más adecuado sería escribir el contenido de este fichero en la pantalla. De esta forma, un cambio en las explicaciones de estas órdenes no requeriría modificar el código fuente, sino simplemente modificar el contenido de este fichero.

El módulo «nombres-ficheros» define una constante con el nombre del fichero en el que se encuentra el texto de la ayuda.

Orden «FICHERO». La función que permita seleccionar el fichero de usos con el que trabajar inicialmente y la que ejecute la orden «FICHERO» tiene que leer del teclado una opción escrita en el teclado por el usuario y generar el nombre del fichero con el que se va a trabajar. El objetivo de la función `construirNombreFicheroUsos` que se declara en el fichero de interfaz del módulo «nombres-ficheros» es facilitar la labor de convertir dicha opción en una ruta de acceso a un fichero de usos relativa al directorio de ejecución del programa.

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

El nombre del fichero de usos tendrá la forma «datos/usos-xx.csv», donde *xx* será la opción elegida por el usuario. La lectura de esta opción como una cadena de caracteres facilitará la generación del nombre del fichero involucrado.

Esta orden no debe limitarse a restringir como opciones válidas aquellas que, con los datos publicados actualmente por el Ayuntamiento de Zaragoza y los derivados de la tarea previa de este trabajo, existen actualmente («16», «17», «t1» o «t2»), sino que debe comprobar que, efectivamente, el fichero especificado por la opción del usuario exista o no en el directorio «datos».

Puede ser tentador definir una variable global para almacenar la opción elegida por el usuario o el nombre del fichero de usos correspondiente con dicha opción, de forma que las funciones

⁷ «res» es una abreviatura de *resources*, y es una práctica habitual que los programas ubiquen en un directorio con este nombre los *recursos* estáticos que vayan a utilizar en forma de ficheros, como imágenes, iconos y ficheros de configuración.

que implementen las órdenes «USOS», «USUARIO», «MAYORES», «INFORME» y «DESTINOS» la consulten directamente. Sin embargo, un buen diseño de la solución exige que cada una de las funciones que implementen dichas órdenes (y aquellas otras funciones auxiliares en las que puedan apoyarse) reciban el nombre del fichero de usos con el que tienen que trabajar a través de un parámetro.

El módulo «nombres-ficheros» define una constante con el nombre del fichero en el que se encuentra el texto correspondiente a las opciones disponibles actualmente.

Orden «USOS». Para implementar la función que ejecute esta orden, se sugiere seguir los siguientes pasos:

Paso 1. El fichero «uso.hpp» define la interfaz de un módulo para trabajar con un tipo registro que representa una utilización concreta del sistema Bizi Zaragoza por parte de un usuario. Completa en él la definición del tipo registro `UsoBizi` para que represente los siguientes datos de un uso del sistema Bizi Zaragoza:

- el identificador entero del usuario que utiliza la bicicleta;
- el código entero de la estación de la que se retira la bicicleta; y
- el código entero 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 de usos al implementar esta orden y la orden «MAYORES».

Paso 2. Escribe el código de las funciones declaradas en el fichero de interfaz «uso.hpp» en su correspondiente fichero de implementación «uso.cpp».

Paso 3. Escribe el código necesario en el módulo principal para que se muestren en la pantalla los resultados obtenidos por el procedimiento `contarUsos`.

Orden «ESTADISTICAS». Los datos para escribir estas estadísticas pueden obtenerse del fichero «usuarios.csv». Los usuarios cuyo género no está registrado en el fichero no deben ser contabilizados en ninguna de las categorías. En el módulo «usuarios» hay declaraciones de constantes (`NUM_EDADES`, `NUM_GENEROS` y `RANGO_EDADES`) y funciones (`obtenerEstadisticas`, `indiceRangoEdad` e `indiceGenero`) que guían la implementación de esta orden.

Orden «USUARIO». Los datos sobre el género y edad del usuario deben obtenerse del fichero «usuarios.csv». En el módulo «usuarios» se declara la función `buscarUsuario`. En su implementación podéis aprovecharos del hecho de que el fichero de usuarios está ordenado por identificadores de usuario crecientes, por lo que en el momento en el que se lee del fichero un identificador mayor que el del usuario que se está buscando, se puede concluir inmediatamente que el usuario no se encuentra en el fichero.

Orden «MAYORES». Para implementar la función que ejecute esta orden, se sugiere seguir los siguientes pasos:

Paso 1. En el fichero de interfaz «uso.hpp», deberíais haber definido los campos del tipo registro `UsoBizi` tal y como se explica en los detalles de implementación de la orden «USOS».

Paso 2. A diferencia del caso del tipo registro `UsoBizi`, para resolver el problema que se plantea en esta tarea, la definición del tipo `UsosUsuario` de la «usos-usuario» 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 `UsosUsuario`.

Paso 3. Para la implementación de esta orden, 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» y «usos-usuario» implementan.

Va a ser necesario trabajar con un vector de registros de tipo `UsosUsuario` en el que ir recogiendo, a nivel de cada 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 los siguientes procedimientos y funciones, que pueden ubicarse en el módulo «usos-usuario»:

- Una función que, **dados un vector de registros de tipo `UsosUsuario` 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 en caso contrario (por ejemplo, -1). 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 `UsosUsuario` 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 la primera 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 `UsosUsuario` un resumen del número de usos que cada usuario presente en el fichero ha realizado**.
- Una función `ordenar` que **ordene el contenido del vector de registros de tipo `UsosUsuario`**. 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 n usuarios con más usos del sistema, siendo n el argumento que el usuario del programa haya establecido al solicitar la ejecución de la orden «MAYORES».
- Un procedimiento en el módulo principal **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.

Orden «INFORME». Para implementar esta opción, pueden utilizarse recursos definidos en el módulo «estaciones»:

- Definid los campos necesarios del tipo registro `Estacion` para representar los datos que nos van a resultar de interés de cada estación Bizi: el identificador de la estación, su nombre y el número de usos de la misma.
- Escribid el código de la función `leerEstaciones`, que se encargará de rellenar, a partir de los datos del fichero «estaciones.csv», los campos que representan el identificador de la estación y su nombre de cada registro de tipo `Estacion` de un vector.
- Escribid el código de la función `contarUsosEstaciones`, que se encargará de rellenar, a partir de los datos del fichero de usos que haya sido seleccionado, la información relativa al número de usos de cada estación.
- Por último, escribid el código del procedimiento `ordenarPorUso`, la función `escribirInformeEstaciones` y utilizad esta última en el módulo principal del programa.

Leed detenidamente las especificaciones de las funciones propuestas, puesto que dan información adicional sobre la forma de implementar esta orden.

Orden «DESTINOS». Los procedimientos y funciones `contarViajesOrigenDestino`, `calcularDestinosMasFrecuentes` y `escribirInformeDestinos` del módulo «estacion» os guiarán en su implementación. Para ello, leed atentamente sus especificaciones.

El procedimiento `escribirInformeDestinos` está pensado para escribir el informe tanto en `cout` como en un flujo de la clase `ofstream` que haya sido declarado y asociado a un fichero externo con anterioridad.

Aunque existen resultados de la ejecución de una orden que podrían ser de utilidad para simplificar la ejecución de otra posterior, para simplificar el problema, vamos a considerar que cada una de las órdenes se ejecuta de forma independiente de las otras y, por lo tanto, cada una, para producir los resultados que se espera de ella, procesará los datos que necesite de los ficheros adecuados sin importar que la ejecución de una orden anterior haya podido realizar procesamientos iguales o similares con los mismos datos. La única excepción es la de la orden «FICHERO», que modifica el estado del programa en el sentido en que fija el fichero de usos con el que deben trabajar las órdenes «USOS», «MAYORES», «INFORME» y «DESTINOS».

Desarrollo modular

Se sugiere repartir el código en distintos módulos de forma que en el módulo principal («bizi-main») se encuentre la función `main()` y todas aquellas funciones auxiliares que interactúen con el usuario del programa (leyendo datos del teclado o escribiendo resultados en la pantalla). En el resto de módulos, deberían ir aquellas funciones que no interactúan con el usuario, pero que facilitan la implementación de las órdenes de mayor complejidad.

- «nombres-ficheros». Este módulo contiene constantes que definen los nombres de los ficheros de usuarios, estaciones y recursos, junto con una función que, a partir de una opción elegida por el usuario al inicio del programa o cuando se ejecute la orden «FICHERO», genera el nombre de un fichero de usos.
- «usos». En este módulo se debe definir un tipo registro que representa usos individuales del sistema Bizi Zaragoza y una función `leerUso` que puede utilizarse en la implementación de las órdenes que necesitan procesar el contenido de ficheros de usos: «USOS», «MAYORES», «INFORME» y «DESTINOS». Esta función se encarga de leer, cada vez que es invocada, un uso del sistema Bizi almacenado en una línea de los ficheros de usos. Como devuelve un dato de tipo booleano que indica si la lectura ha podido realizarse, puede utilizarse como condición de iteración en los bucles de las funciones que tengan que recorrer los ficheros de usos.

También se incluye la cabecera del procedimiento `contarUsos` que se utilizará en la implementación de la orden «USOS».

A este módulo se le pueden añadir todas aquellas funciones relacionadas con la gestión de datos de tipo `UsoBizi` que se identifiquen de forma adicional durante el desarrollo del programa.

- «usuarios». Contiene declaraciones de constantes y funciones para la implementación de las órdenes «ESTADISTICAS» y «USUARIO».
- «usos-usuario». Se sugiere crear este módulo en el que declarar un tipo registro denominado `UsosUsuario` en el que representar la información sobre utilizaciones del sistema Bizi por parte de sus usuarios Bizi con la que se va a trabajar en la orden «MAYORES». En dicho módulo se pueden añadir todas aquellas funciones adicionales que gestionen datos de tipo `UsosUsuario` que se identifiquen durante el desarrollo del programa.
- «estacion». Este módulo contiene declaraciones de tipos y funciones pensadas para implementar las órdenes «INFORME» y «DESTINO». Se le pueden añadir todas aquellas funciones adicionales que estiméis convenientes y que estén relacionadas con estas órdenes.

- Módulo principal, con la función `main()`. Se sugiere aplicar un diseño modular en el que cada orden ofrecida por el programa sea implementada por una función específica. Este módulo puede hacer uso de las funciones de los otros módulos que han de desarrollarse y pueden definirse en él todas aquellas funciones adicionales que se estimen convenientes.

6.5.1. Recomendaciones y restricciones de diseño

Modularidad del diseño. El programa debe diseñarse de forma que esté integrado por un conjunto de funciones repartidas en los módulos que se suministran en el código de partida. Estas funciones deberían resolver problemas acotados y concretos, siempre que sea posible. Se recuerda que copiar y pegar código no es una forma adecuada de reutilizar código: ante situaciones en las que la solución a parte de un problema es similar a parte de otro, debe definirse una función, adecuadamente parametrizada, que permita resolver ambos problemas haciendo las invocaciones adecuadas a la función creada.

El número de líneas de código de cada función debe ser reducido. Conviene facilitar la comprensión del código de aquellas funciones que revistan cierta complejidad incorporando, con medida, comentarios que aclaren su comportamiento y eviten tener que hacer un seguimiento detallado del código para comprender su comportamiento. En cualquier caso, se recuerda que la mejor documentación del código consiste en utilizar identificadores adecuados y significativos y utilizar funciones que resuelvan problemas concretos siempre que sea posible.

Documentación. Las primeras líneas de los ficheros que forman el programa estarán dedicadas a un comentario en el que figuren los nombres y apellidos de los componentes del equipo.

Así mismo, cada función debe contar con su propia especificación mediante:

1. Una precondition que describa con precisión las condiciones que, en su caso, deben satisfacer sus datos de entrada.
2. Una postcondición que describa con precisión las condiciones que deben satisfacer sus datos de salida o resultados.

Legibilidad. La edición del programa fuente debe facilitar su lectura y comprensión. Para ello se deberán tener en cuenta, entre otros, los siguientes aspectos:

1. Elección de identificadores significativos.
2. Alineación y sangrado de las declaraciones y del código de cada algoritmo que faciliten la comprensión de su estructura.
3. Evitar líneas de longitud excesiva que dificulten su lectura en pantalla o su impresión. No conviene que la longitud de ninguna línea sobrepase los 120 caracteres.

Modificabilidad. El diseño del programa debe facilitar los cambios y modificaciones que se realicen durante la explotación del programa.

6.6. Equipos de trabajo

La formación de los equipos de dos estudiantes para la realización de este trabajo será libre y se realizará a través de la tarea de Moodle «Formación de grupos para el trabajo obligatorio»⁸. Los equipos podrán estar integrados por cualquier pareja de estudiantes matriculados en la asignatura, pudiendo ser de distintos grupos de teoría o prácticas. Es obligatorio pertenecer a uno de los grupos de realización del trabajo para poder realizar la entrega a través de Moodle.

⁸<https://moodle.unizar.es/add/mod/choicegroup/view.php?id=3807651>

Quien no encuentre pareja para apuntarse en la tarea, podrá inscribirse en un equipo de forma individual y esperar a que alguien en la misma situación se inscriba en su grupo, o bien inscribirse en un equipo en el que ya haya un estudiante, para completarlo. Cuando os inscribáis en un equipo, debéis ser conscientes de que adquiriréis un compromiso personal y académico con el otro miembro del equipo en cuanto a la realización, finalización y entrega del trabajo.

En caso de desavenencias entre los miembros de un equipo, se ofrece la posibilidad de modificar la composición de los mismos en cualquier momento (dentro de la disponibilidad de huecos que pueda haber), **con la fecha límite del 8 de enero de 2023**. A partir de ese momento, la composición de los grupos quedará fijada de forma definitiva.

El programa que se pide realizar tiene una carga de trabajo suficiente como para que sea objeto de una cuidadosa planificación por parte de los miembros de cada equipo, que va más allá de repartirse distintas funcionalidades para ser desarrolladas independientemente y luego integradas en una entrega final. **Lo que se espera es que realmente se trabaje en equipo, con comunicación frecuente y reuniones para la coordinación del trabajo y la implementación del mismo.**

La forma de colaboración que se sugiere es similar a la que en metodologías ágiles se denomina *programación por pares*⁹ (*pair programming* en inglés). Aplicando esta técnica, dos programadores trabajan simultáneamente en la escritura del código de un programa de forma conjunta, a la vez que planean y reflexionan sobre el trabajo que realizan, aclaran sus ideas y evalúan soluciones alternativas, dando como resultado *software* de mejor calidad¹⁰.

Habitualmente, se definen dos papeles:

1. *Piloto* o *controlador* (*driver*, en inglés), que es la persona que utiliza el teclado en un momento dado y, por lo tanto, escribe el código, concentrada en las líneas de código que está escribiendo, mientras comenta al otro programador lo que está haciendo.
2. *Navegador* u *observador* (*navigator*, en inglés), que es la persona que se mantiene en una posición observadora mientras la otra está tecleando. Se encarga de revisar el código que escribe su compañero mientras lo escribe, le guía y comparte ideas. Se mantiene alerta ante problemas o errores sintácticos o de ejecución que puedan estar introduciéndose en el código y va pensando en los siguientes pasos a realizar.

Se recomienda alternar estos dos roles con regularidad.

La técnica de programación por pares puede aplicarse de forma presencial, compartiendo un único equipo, o de forma remota. La extensión Live Share¹¹ de Visual Studio Code es una posibilidad. Requiere de una cuenta en GitHub¹² o de Microsoft para trabajar.

También es posible aplicarla de forma remota reuniéndose en una sala de Google Meet. El piloto escribe el código mientras comparte su pantalla y el navegador observa, guía y comparte sus ideas. El cambio de roles implica intercambiar primero el código (a través del correo electrónico, una carpeta compartida en Google Drive o incluso desde la tarea de entrega del trabajo en Moodle) y, después, que el nuevo piloto comparta su pantalla.

⁹<https://martinfowler.com/bliki/PairProgramming.html>

¹⁰<https://martinfowler.com/articles/on-pair-programming.html>

¹¹<https://docs.microsoft.com/en-us/visualstudio/liveshare/>

¹²<https://github.com/>

6.7. Entrega y evaluación del trabajo

Antes del **martes 10 de enero de 2023 a las 23:59**, se deben haber subido a Moodle los siguientes 11 ficheros:

- «0-datos-pruebas.cpp»
- «bizi-main.cpp»
- «estacion.hpp»
- «estacion.cpp»
- «nombres-ficheros.cpp»
- «uso.hpp»
- «uso.cpp»
- «usos-usuario.hpp»
- «usos-usuario.cpp»
- «usuarios.hpp»
- «usuarios.cpp»

Cada uno de los trabajos realizados por cada equipo será evaluado y calificado sobre 10 puntos. La calificación final de la primera convocatoria tiene en cuenta la nota de este trabajo en un 15 %.

Para posibilitar una compilación automática, deben entregarse todos los ficheros indicados en el apartado anterior, incluso si en la descomposición modular realizada, alguno de ellos no hubiera sido utilizado. En este caso, pueden entregarse ficheros sin contenido, pero deben ser subidos a Moodle igualmente. El código fuente entregado debe poderse compilar utilizando el fichero «Makefile» original suministrado en el repositorio del material de partida (es decir, **no** se debe modificar el «Makefile» proporcionado).

Cada uno de los ficheros entregados estará encabezado por un comentario en el que consten el nombre y apellidos de los autores del trabajo.

No se admitirán trabajos presentados mediante otros medios ni presentados fuera de plazo.

El trabajo que se propone debe ser realizado por cada equipo con total independencia de los demás equipos. Debe evitarse que miembros de otros equipos tengan acceso a los documentos de trabajo y programas desarrollados por el equipo. **Se hará un análisis de plagios entre trabajos.** La copia o coincidencia manifiesta de una parte o de la totalidad de un programa con el presentado por otro equipo conllevará una calificación de un cero en la nota del trabajo a todos los miembros de los equipos implicados.

Al corregir, se compilará el código de los ficheros entregados. La existencia de errores sintácticos en los mismos supondrá un cero en la calificación del trabajo.

Para calificar el trabajo se tendrán en cuenta los siguientes aspectos:

- **Fidelidad del comportamiento del programa a lo especificado y descrito en este documento** (50 % de la calificación del trabajo).

Se admitirán entregas de programas que no satisfagan la totalidad de los requisitos planteados en este enunciado, siempre que el programa pueda ser compilado correctamente y que la ejecución del programa se produzca sin errores. En todo caso, los distintos requisitos planteados en este enunciado tendrán pesos proporcionales a la dificultad de los mismos a la hora de establecer la calificación, y la no implementación de algunos requisitos también repercutirá en las calificaciones de los siguientes aspectos.

- **Calidad del diseño algorítmico del código (datos y funciones)** (35 % de la calificación del

trabajo). Se espera que el código entregado se adhiera a los diferentes criterios de calidad explicados en la asignatura (diseño descendente, eficiencia, reusabilidad del código, etc.).

- **Presentación y legibilidad del código** (15 % de la calificación del trabajo), teniendo muy presentes las reglas publicadas en la *Guía de estilo para programar en C++*, accesible desde Moodle¹³.

El código C++ del programa desarrollado por el equipo debe ser conservado por cada estudiante durante todo el curso.

¹³<https://moodle.unizar.es/add/mod/resource/view.php?id=3807888>

6.8. Traza de ejecución

```

Elección de ficheros de usos y usuarios. Opciones disponibles:
16: octubre 2016 a marzo 2017
17: marzo 2017 a agosto 2017
t1: datos para pruebas (10 líneas)
t2: datos para pruebas (2000 líneas)
Introduzca una opción: 16
El fichero "datos/usos-16.csv" existe y ha sido seleccionado.

ÓRDENES DISPONIBLES
=====
AYUDA:           Muestra esta pantalla de ayuda.
FICHERO:         Permite modificar la selección del fichero de usos a utilizar.
USOS:           Escribe en la pantalla el número de usos de traslado y circulares.
ESTADISTICAS:   Informa la distribución por edades y géneros de los usuarios.
USUARIO <id-usuario>: Informa acerca del número de usos realizados por el
                  usuario especificado.
MAYORES <n>:     Escribe en la pantalla el número de usuarios distintos y los
                  <n> usuarios que más usos han hecho del sistema.
INFORME <nombre-fichero>: Escribe en el fichero especificado un informe con el
                  número de usos de las estaciones, según el fichero seleccionado.
DESTINOS:       Escribe en un fichero de texto o en la pantalla un informe en el
                  que, para cada estación dada, se indica la estación a la que más
                  se ha viajado desde ella.
FIN:            Termina la ejecución de este programa.

Orden: usos
Fichero de usos seleccionado actualmente: "datos/usos-16.csv".
Número de usos como traslado: 1010992
Número de usos circulares:    13828
Número total de usos:         1024820

Orden: ESTADISTICAS
Distribución de los usuarios
      |   M   F
-----+-----
<=25 | 1567 1306
26-35 | 2200 2214
36-50 | 4686 3857
51-65 | 3692 2326
>65   |  848  192

Orden: Ayuda

ÓRDENES DISPONIBLES
=====
AYUDA:           Muestra esta pantalla de ayuda.
FICHERO:         Permite modificar la selección del fichero de usos a utilizar.
USOS:           Escribe en la pantalla el número de usos de traslado y circulares.
ESTADISTICAS:   Informa la distribución por edades y géneros de los usuarios.
USUARIO <id-usuario>: Informa acerca del número de usos realizados por el
                  usuario especificado.
MAYORES <n>:     Escribe en la pantalla el número de usuarios distintos y los
                  <n> usuarios que más usos han hecho del sistema.
INFORME <nombre-fichero>: Escribe en el fichero especificado un informe con el
                  número de usos de las estaciones, según el fichero seleccionado.
DESTINOS:       Escribe en un fichero de texto o en la pantalla un informe en el
                  que, para cada estación dada, se indica la estación a la que más
                  se ha viajado desde ella.
FIN:            Termina la ejecución de este programa.

...

```

...

Orden: usuario 55381

La usuaria 55381 está en el rango de edad "26-35".

Orden: USUARIO 16391

El usuario 16391 está en el rango de edad ">65".

Orden: Usuario 91455

El/la usuario/a 91455 está en el rango de edad "<=25".

Orden: usuario 10201

El/la usuario/a 10201 no aparece en el fichero "datos/usuarios.csv".

Orden: uSuArIo 93385

La usuaria 93385 está en el rango de edad "26-35".

Orden: mayores 15

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 |

Orden: informe estaciones16.txt

Informe "estaciones16.txt" generado correctamente.

Orden: destinos

Escriba el nombre del fichero del informe

(presione solo ENTRAR para escribirlo en la pantalla): destinos16.txt

Informe "destinos16.txt" creado correctamente.

Orden: cambiar

Orden "CAMBIAR" desconocida

Orden: fichero

Elección de ficheros de usos y usuarios. Opciones disponibles:

16: octubre 2016 a marzo 2017

17: marzo 2017 a agosto 2017

t1: datos para pruebas (10 líneas)

t2: datos para pruebas (2000 líneas)

Introduzca una opción: 17

El fichero "datos/usuarios-17.csv" existe y ha sido seleccionado.

Orden: informe estaciones17.txt

Informe "estaciones17.txt" generado correctamente.

...

...

Orden: **usos**

Fichero de usos seleccionado actualmente: "datos/usos-17.csv".

Número de usos como traslado: 1001208

Número de usos circulares: 15726

Número total de usos: 1016934

Orden: **mayores 10**

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 |

Orden: **destinos**

Escriba el nombre del fichero del informe

(presione solo ENTRAR para escribirlo en la pantalla):

Viajes Origen --> Destino

```

217  1-Avda. Pablo Ruiz Picasso - Torre del Agua --> 1-Avda. Pablo Ruiz Picasso - Torre del Agua
369  2-Avda. Ranillas - Avda. Pablo Ruiz Picasso --> 12-Pasarela del Voluntariado - Avda. Almozara
436  3-Pasarela del Voluntariado - Avda. Ranillas --> 86-Avda. Almozara - CDM Almozara
391  4-Avda. Ranillas - Puente de La Almozara --> 10-Pº Echegaray y Caballero - Puente de Santiago
652  5-Pº de la Ribera - Puente de Santiago --> 75-Avda. Salvador Allende - C/ Rubio de Francia
745  6-Pº de la Ribera - Puente del Pilar --> 34-Plaza Magdalena
285  7-Pº Echegaray y Caballero - Pasarela del Huerva --> 10-Pº Echegaray y Caballero - Puente ...
606  8-Pº Echegaray y Caballero - Puente del Pilar --> 101-C/ Doctor Iranzo - C/ Escultor ...
740  9-Pº Echegaray y Caballero - Puente de Piedra --> 11-Pº Echegaray y Caballero - Puente de ...
1196 10-Pº Echegaray y Caballero - Puente de Santiago --> 67-Avda. G. Gómez de Avellaneda - C/ ...
881  11-Pº Echegaray y Caballero - Puente de La Almozara --> 9-Pº Echegaray y Caballero - Puente ...
1492 12-Pasarela del Voluntariado - Avda. Almozara --> 67-Avda. G. Gómez de Avellaneda - C/ ...
270  13-C/ Francia - Pabellón Puente --> 2-Avda. Ranillas - Avda. Pablo Ruiz Picasso
198  14-Estación Intermodal Delicias (Salidas) --> 2-Avda. Ranillas - Avda. Pablo Ruiz Picasso
412  15-Estación de Cercanías El Portillo --> 46-Avda. Goya - C/ Baltasar Gracián
996  16-Plaza España --> 34-Plaza Magdalena
421  17-Pº Constitución - Plaza Aragón --> 40-Avda. Cesáreo Alierta - C/ Félix Burriel
544  18-C/ Asalto - Edificio Trovador --> 39-Cno. de las Torres - Avda. Cesáreo Alierta
555  19-C/ Asalto - C/ Miguel Servet --> 37-C/ Florentino Ballesteros - C/ Matadero
372  20-Jose Camón Aznar --> 44-Pº Sagasta - Cno. de las Torres
611  21-Avda. César Augusto - Avda. Conde Aranda --> 31-Avda. Pablo Gargallo - C/ Ignacio Menaya
201  22-Estación Intermodal Delicias (Llegadas) --> 13-C/ Francia - Pabellón Puente
512  23-Pº de La Ribera - Pasarela Azud --> 9-Pº Echegaray y Caballero - Puente de Piedra
368  24-Plaza San Pedro Nolasco --> 37-C/ Florentino Ballesteros - C/ Matadero
300  25-Plaza de los Sitios --> 18-C/ Asalto - Edificio Trovador

```

...

```
...
Orden: fichero

Elección de ficheros de usos y usuarios. Opciones disponibles:
16: octubre 2016 a marzo 2017
17: marzo 2017 a agosto 2017
t1: datos para pruebas (10 líneas)
t2: datos para pruebas (2000 líneas)
Introduzca una opción: 22
No se ha podido abrir el fichero "datos/usos-22.csv".
```

```
Elección de ficheros de usos y usuarios. Opciones disponibles:
16: octubre 2016 a marzo 2017
17: marzo 2017 a agosto 2017
t1: datos para pruebas (10 líneas)
t2: datos para pruebas (2000 líneas)
Introduzca una opción: a
No se ha podido abrir el fichero "datos/usos-a.csv".
```

```
Elección de ficheros de usos y usuarios. Opciones disponibles:
16: octubre 2016 a marzo 2017
17: marzo 2017 a agosto 2017
t1: datos para pruebas (10 líneas)
t2: datos para pruebas (2000 líneas)
Introduzca una opción: t1
El fichero "datos/usos-t1.csv" existe y ha sido seleccionado.
```

```
Orden: usos
Fichero de usos seleccionado actualmente: "datos/usos-t1.csv".
Número de usos como traslado:      8
Número de usos circulares:         1
Número total de usos:              9
```

```
Orden: mayores 20
```

```
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 |

```
Orden: informe estaciones-t1.txt
Informe "estaciones-t1.txt" generado correctamente.
```

```
Orden: destinos
Escriba el nombre del fichero del informe
(presione solo ENTRAR para escribirlo en la pantalla):
Viajes Origen --> Destino
```

```
-----
0  1-Avda. Pablo Ruiz Picasso - Torre del Agua --> 1-Avda. Pablo Ruiz Picasso - Torre del Agua
0  2-Avda. Ranillas - Avda. Pablo Ruiz Picasso --> 1-Avda. Pablo Ruiz Picasso - Torre del Agua
0  3-Pasarela del Voluntariado - Avda. Ranillas --> 1-Avda. Pablo Ruiz Picasso - Torre del Agua
0  4-Avda. Ranillas - Puente de La Almozara --> 1-Avda. Pablo Ruiz Picasso - Torre del Agua
0  5-Pº de la Ribera - Puente de Santiago --> 1-Avda. Pablo Ruiz Picasso - Torre del Agua
1  6-Pº de la Ribera - Puente del Pilar --> 55-Avda. Marqués de la Cadena - C/ Miguel Asso
...
```

...

Orden: **fichero**

Elección de ficheros de usos y usuarios. Opciones disponibles:

16: octubre 2016 a marzo 2017

17: marzo 2017 a agosto 2017

t1: datos para pruebas (10 líneas)

t2: datos para pruebas (2000 líneas)

Introduzca una opción: **t2**

El fichero "datos/usos-t2.csv" existe y ha sido seleccionado.

Orden: **usos**

Fichero de usos seleccionado actualmente: "datos/usos-t2.csv".

Número de usos como traslado: 1989

Número de usos circulares: 10

Número total de usos: 1999

Orden: **mayores 5**

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 |

Orden: **informe estaciones-t2.txt**

Informe "estaciones-t2.txt" generado correctamente.

Orden: **destinos**

Escriba el nombre del fichero del informe

(presione solo ENTRAR para escribirlo en la pantalla): datos/destinos-t2.txt

Informe "datos/destinos-t2.txt" creado correctamente.

Orden: **acabar**

Orden "ACABAR" desconocida

Orden: **fin**