

***Trabajo Práctico Especial  
Programación Imperativa  
Julio 2020***

## ***1. Objetivo***

Diseñar e implementar un programa para el **procesamiento de datos de árboles en el espacio público de una ciudad**, basado en datos reales. Para ello se deberá realizar tanto el *front-end* como el *back-end*, este último **basado en la creación de al menos un TAD**.

Para este trabajo se busca poder procesar los datos de los árboles dos ciudades: Ciudad Autónoma de Buenos Aires, Argentina  y Vancouver, British Columbia, Canadá  Ambos datos son extraídos de los respectivos portales de gobierno en formato CSV.

## ***2. Descripción funcional***

El programa consiste en dos ejecutables distintos, uno para cada dataset.

Cada uno de estos ejecutables deberá leer dos archivos CSV: uno con la información de los árboles correspondientes a una ciudad y otro archivo con la información de los barrios de esa ciudad. En ambos el delimitador es un punto y coma (“;”).

A continuación se listan los dos ejemplos de uso que se busca para la aplicación: procesar los datos de árboles de la Ciudad Autónoma de Buenos Aires  y de Vancouver  . Sin embargo, es importante recordar que la mayor parte de la implementación no debe estar atada a la realidad de los ejemplos de uso. **Por ejemplo, los barrios serán los que la aplicación obtenga en ejecución a partir del archivo de barrios y no serán aceptadas implementaciones que tengan fijos estos datos.** En otras palabras, la implementación deberá funcionar también para procesar los datos de árboles de cualquier otra ciudad, manteniendo siempre la estructura de los archivos que se presenta a continuación. Si bien se detalla el origen de los archivos, los mismos fueron modificados para simplificar su procesamiento.

Los archivos tienen las siguientes características:

Datos de árboles de Buenos Aires , a partir de ahora **`árbolesBUE.csv`**

- ❖ **Origen:**  
<https://data.buenosaires.gob.ar/dataset/arbolado-publico-lineal/archivo/ecf38a47-563f-42c1-9bd4-7cedf35d536b>
- ❖ **Descarga: En Campus ITBA.**
- ❖ **Cantidad de líneas:** 370.089 (incluyendo el encabezado)
- ❖ **Campos Relevantes:**
  - **comuna:** Nombre del barrio donde se encuentra el árbol.
  - **calle\_nombre:** Nombre de la calle donde se encuentra el árbol
  - **nombre\_cientifico:** Nombre científico del árbol
  - **diametro\_altura\_pecho:** Diámetro del árbol a la altura del pecho

El archivo se compone de una primera línea de encabezado, con los títulos de cada campo. De la segunda línea en adelante, cada línea representa un árbol conteniendo los datos de cada uno de los campos, separados por “;”.

```

nro_registro;tipo_activ;comuna;manzana;calle_nombre;calle_altura;di-
rección_normalizada;nombre_científico;estado_plantera;ubicacion_plantera;
nivel_plantera;diametro_altura_pecho;altura_arbol
79838;Lineal;1; ;Eyle Petrona;0;EYLE, PETRONA 47;Platanus x
acerifolia;Ocupada;Fuera de línea;A nivel;1;17
...

```

Datos de barrios de Buenos Aires , a partir de ahora **barriosBUE.csv**

- ❖ **Descarga:** En Campus ITBA. El archivo es un subconjunto del dataset, conteniendo únicamente los campos relevantes.
- ❖ **Cantidad de líneas:** 16 (incluyendo el encabezado)
- ❖ **Campos:**
  - **nombre:** Nombre del barrio, para relacionarlo con el campo **comuna** de **arbolesBUE.csv**.
  - **habitantes:** Cantidad de habitantes del barrio.

El archivo se compone de una primera línea de encabezado. De la segunda línea en adelante, cada línea representa un barrio.

**nombre;habitantes**

2;149607

14;227003

15;182427

...

Datos de árboles de Vancouver , a partir de ahora **arbolesVAN.csv**

- ❖ **Origen:** <https://opendata.vancouver.ca/explore/dataset/street-trees/>
- ❖ **Descarga:** En Campus ITBA.
- ❖ **Cantidad de líneas:** 146.746 (incluyendo el encabezado)
- ❖ **Campos Relevantes:**
  - **NEIGHBOURHOOD\_NAME:** Nombre del barrio donde se encuentra el árbol.
  - **STD\_STREET:** Nombre de la calle donde se encuentra el árbol
  - **COMMON\_NAME:** Nombre científico del árbol
  - **DIAMETER:** Diámetro del árbol a la altura del pecho

El archivo se compone de una primera línea de encabezado, con los títulos de cada campo. De la segunda línea en adelante, cada línea representa un árbol conteniendo los datos de cada uno de los campos, separados por “,”.

```

TREE_ID;CIVIC_NUMBER;STD_STREET;GENUS_NAME;SPECIES_NAME;CULTIVAR_NA-
ME;COMMON_NAME;ASSIGNED;ROOT_BARRIER;PLANT_AREA;ON_STREET_BLOCK;ON_STREET-
;NEIGHBOURHOOD_NAME;STREET_SIDE_NAME;HEIGHT_RANGE_ID;DIAMETER;CURB;DATE_P-
LANTED;Geom

```

```
20666;2838;W 19TH AV;PRUNUS;CERASIFERA;ATROPURPUREUM;PISSARD
PLUM;N;N;10;2800;W 19TH AV;ARbutus-RIDGE;EVEN;3;20.0;Y;;"{""type"":""Point"""
, ""coordinates"": [-123.16858, 49.25546]}"
```

Datos de barrios de Vancouver , a partir de ahora **barriosVAN.csv**

- ❖ **Descarga:** En Campus ITBA. El archivo es un subconjunto del dataset, conteniendo únicamente los campos relevantes.
- ❖ **Cantidad de líneas:** 23 (incluyendo el encabezado)
- ❖ **Campos:**
  - **nombre:** Nombre del barrio, para relacionarlo con el campo **NEIGHBOURHOOD\_NAME** de arbolesVAN.csv.
  - **habitantes:** Cantidad de habitantes del barrio.

El archivo se compone de una primera línea de encabezado. De la segunda línea en adelante, cada línea representa un barrio.

```
nombre;habitantes
WEST POINT GREY;13065
HASTINGS-SUNRISE;34575
KERRISDALE;13975
...
```

*Se asume que el formato y contenido de los archivos es correcto.*

El programa deberá recibir por línea de comando (no por entrada estándar) el path del archivo de árboles y el path del archivo de barrios.

El ejecutable para procesar los datos de  debe llamarse **arbolesADTBUE**.

El ejecutable para procesar los datos de  debe llamarse **arbolesADTVAN**.

Por ejemplo, si se desea procesar los archivos CSV de  y se llaman **arboles.csv** y **barrios.csv** y están en el mismo directorio que el ejecutable **arbolesADTBUE**, el programa se puede invocar como:

```
$> ./arbolesADTBUE arboles.csv barrios.csv
```

Si ambos archivos CSV se llaman **arb.csv** y **bar.csv** y están en el directorio superior al ejecutable **arbolesADTBUE**, se invocará como

```
$> ./arbolesADTBUE ../arb.csv ../bar.csv
```

Una vez recibido correctamente los path de los archivos CSV, el programa deberá resolver las consultas (*queries*) que se listan a continuación, dejando los resultados de cada una en archivos .csv con el nombre pedido y localizados en el mismo directorio que el ejecutable.

Resolver todas las consultas con una única lectura de cada uno de los archivos y de una sola vez. No se aceptarán implementaciones que ofrezcan un menú de opciones o similar que permita al usuario decidir cuál de las consultas procesar.

Los ejemplos de salida que se listan a continuación tienen valores ficticios y **no coinciden con la salida esperada para los archivos CSV.**

En caso de rendir en la primera fecha de final y formando un grupo de tres o menos alumnos, deberán implementar las tres primeras queries.

En caso de rendir en la primera fecha de final y formando un grupo de cuatro alumnos, deberán implementar todas las queries.

En caso de rendir en la segunda fecha de final deberán formar grupos de tres o menos alumnos e implementar todas las queries. No se aceptarán grupos de cuatro integrantes para la segunda fecha.

### Query 1: Total de árboles por barrio

Donde cada línea de la salida contenga, separados por “;” el nombre del barrio y el total de árboles pertenecientes a ese barrio.

Sólo se deben listar los barrios presentes en el archivo CSV de barrios.

El orden de impresión es descendente por cantidad de árboles y luego alfabético por nombre de barrio.

Nombre del archivo: **query1.csv**

Salida de ejemplo para :

#### **BARRIO;ARBOLES**

11;35620

12;34118

9;33601

10;31012

4;29446

...

Salida de ejemplo para :

#### **BARRIO;ARBOLES**

KENSINGTON-CEDAR COTTAGE;10012

HASTINGS-SUNRISE;9128

RENFREW-COLLINGWOOD;9128

DUNBAR-SOUTHLANDS;8289

SUNSET;6223

...

## Query 2: Total de árboles por habitante

Donde cada línea de la salida contenga, separados por “;” el nombre del barrio y el total de árboles por habitante (que consiste en el cociente entre el total de árboles de ese barrio y el número de habitantes del mismo).

Sólo se deben listar los barrios presentes en el archivo CSV de barrios.

El orden de impresión es descendente por el total de árboles por habitante y luego alfabético por nombre de barrio.

El total de árboles por habitante debe imprimirse truncado con dos decimales.

Nombre del archivo: **query2.csv**

Salida de ejemplo para :

**BARRIO;ARBOLES\_POR\_HABITANTE**

9;0.24

10;0.20

11;0.20

15;0.16

12;0.11

...

Salida de ejemplo para :

**BARRIO;ARBOLES\_POR\_HABITANTE**

SHAUGHNESSY;0.92

KERRISDALE;0.53

SOUTH CAMBIE;0.51

DUNBAR-SOUTHLANDS;0.50

WEST POINT GREY;0.43

...

## Query 3: Diámetro promedio por especie de árbol

Donde cada línea de la salida contenga, separados por “;” el nombre de la especie del árbol y el promedio del diámetro del árbol a la altura del pecho de esa especie.

El orden de impresión es descendente por diámetro y luego alfabético por nombre de la especie.

El promedio del diámetro del árbol a la altura del pecho debe imprimirse truncado con dos decimales.

Nombre del archivo: **query3.csv**

Salida de ejemplo para :

```
NOMBRE_CIENTIFICO;PROMEDIO_DIAMETRO
Sterculia coccinea;75.20
Eucalyptus tereticornis;71.12
Tristania conferta;70.39
Salix alba;66.51
Caryota urens;65.42
...
```

- Salida de ejemplo para :

```
NOMBRE_CIENTIFICO;PROMEDIO_DIAMETRO
MANCHURIAN BIRCH;36.15
LEYLAND CYPRESS;35.53
AMERICAN CHESTNUT;32.11
GRAY POPLAR;32.10
JAPANESE WALNUT;29.19
...
```

#### Query 4: Especie de árbol más popular por barrio

Donde cada línea de la salida contenga, separados por “;” el nombre del barrio y el nombre de la especie del árbol más popular de ese barrio.

Sólo se deben listar los barrios presentes en el archivo CSV de barrios.

El orden de impresión es alfabético por nombre de barrio.

- Nombre del archivo: **query4.csv**

- Salida de ejemplo para :

```
BARRIO;NOMBRE_CIENTIFICO
1;Fraxinus pennsylvanica
10;Fraxinus pennsylvanica
11;Fraxinus pennsylvanica
12;Fraxinus pennsylvanica
13;Fraxinus pennsylvanica
...
```

- Salida de ejemplo para :

```
BARRIO;NOMBRE_CIENTIFICO
ARBUTUS-RIDGE;PISSARD PLUM
DOWNTOWN;RED MAPLE
DUNBAR-SOUTHLANDS;KWANZAN FLOWERING CHERRY
FAIRVIEW;RED MAPLE
GRANDVIEW-WOODLAND;KWANZAN FLOWERING CHERRY
...
```

## Query 5: Calle con más árboles por barrio

Donde cada línea de la salida contenga, separados por “;” el nombre del barrio, el nombre de la calle con más árboles de ese barrio y la cantidad de árboles que hay en esa calle de ese barrio.

Sólo se deben listar los barrios presentes en el archivo CSV de barrios.

El orden de impresión es alfabético por el nombre del barrio.

Nombre del archivo: **query5.csv**

Salida de ejemplo para :

```
BARRIO;CALLE_CON_MAS_ARBOLES;ARBOLES
1;Saenz Peña Roque,Pres. Diagonal Norte Av.;123
10;Lascano;436
11;Lamarca Emilio;712
12;Quesada;523
13;3 de Febrero;439
...
```

Salida de ejemplo para :

```
BARRIO;CALLE_CON_MAS_ARBOLES;ARBOLES
ARBUTUS-RIDGE;W KING EDWARD AV;276
DOWNTOWN;PACIFIC BOULEVARD;173
DUNBAR-SOUTHLANDS;W KING EDWARD AV;463
FAIRVIEW;W BROADWAY;399
GRANDVIEW-WOODLAND;VICTORIA DRIVE;295
...
```

*Para todas las queries, tener en cuenta que todos los archivos de salida deben contener la línea de encabezado correspondiente indicada en las salidas de ejemplo.*

### 3. Contenidos

Para la realización del presente trabajo se recomienda consultar el siguiente material:

- Para leer argumentos por línea de comandos:  
[https://www.tutorialspoint.com/cprogramming/c\\_command\\_line\\_arguments.htm](https://www.tutorialspoint.com/cprogramming/c_command_line_arguments.htm)
- `strtok` puede resultar útil para "parsear" las líneas de los archivos:  
[https://www.tutorialspoint.com/c\\_standard\\_library/c\\_function\\_strtok.htm](https://www.tutorialspoint.com/c_standard_library/c_function_strtok.htm)
- **Presentación de Archivos:** Contiene lo esencial para el manejo de archivos en C. Tener en cuenta que se procesarán únicamente archivos de texto en forma secuencial, por lo que las funciones a utilizar serán `fopen`, `fgets`, `fclose`, `fprintf` o similares. Se encuentra en Campus ITBA.

De todas formas, los alumnos pueden consultar dudas relacionadas a estos nuevos contenidos usando los foros de discusión de Campus ITBA. No se responderán consultas que no sean formuladas a través del Campus.

#### **4. Diseño e Implementación del Programa**

Se debe realizar un diseño donde se separe claramente la lectura y preparación de datos (*front-end*) del almacenamiento y procesamiento de los datos (*back-end*).

Recordar que ninguna función de back-end debe invocar a una función de front-end.

Para el almacenamiento de los datos se deberá desarrollar al menos un TAD.

En ningún caso se debe repetir código para resolver situaciones similares, sino que debe implementarse una correcta modularización y se deben reutilizar funciones parametrizadas.

Tanto la biblioteca como el *front-end* deben estar correctamente comentados y en el caso de la biblioteca se debe escribir el archivo de encabezado correspondiente.

El programa no debe presentar leaks de memoria.

El programa no debe abortar por ningún motivo y ante cualquier error se debe mostrar un mensaje adecuado.

Se tendrá en cuenta la eficiencia en el uso de recursos, tanto el tiempo de ejecución como la memoria utilizada.

#### **5. Uso de Git**

Es obligatorio el uso de un repositorio Git para la resolución de este final. Siguiendo los pasos indicados en la clase de Taller de Git deberán crear un repositorio en GitHub donde todos los integrantes del grupo colaboren con las modificaciones del código provisto. No se aceptarán entregas que utilicen un repositorio *git* con un único *commit* que consista en la totalidad del código a entregar.

Los distintos *commits* deben permitir ver la evolución del trabajo, tanto grupal como individual

**Muy importante:** los repositorios creados deben ser privados.

#### **6. Material a entregar**

Cada grupo deberá subir al Campus ITBA un archivo compactado conteniendo como mínimo los siguientes archivos:

- Archivos fuentes y de encabezado.
- Archivo de texto README con la explicación de cómo generar los ejecutables y de cómo ejecutarlos.
- `makefile`. Deberá generar los dos ejecutables pedidos. Consultar apunte en Campus ITBA.

Revisar que en el archivo comprimido también se encuentra el directorio oculto `.git/` donde se detallan todas las modificaciones realizadas.

#### **7. Armado de Grupos**

Los alumnos deben informar la conformación del grupo y la fecha de final elegida en el Foro de Discusión “**TPE Final**” de Campus ITBA antes del **13/07/2020 23:59 ART**.

Cada grupo deberá realizar la entrega mediante la actividad correspondiente en Campus ITBA:

- En caso de rendir en la primera fecha de final (17/07/2020 08:00 ART) deberán entregar antes del 14/07/2020 23:59 ART.
- En caso de rendir en la segunda fecha de final (24/07/2020 08:00 ART) deberán entregar antes del 21/07/2020 23:59 ART.

En ambos casos enviar sólo el material pedido. No incluir código compilado, archivos de prueba publicados, etc.

En ambos casos el grupo completo deberá asistir a la Sala Virtual de Campus ITBA en un día y hora previamente acordada (se asignarán turnos para todos los grupos que rinden el final ese día). En la Sala Virtual se comunicará la nota del final y los integrantes del grupo rendirán un coloquio, el cual podrá modificar la nota individual de los integrantes del grupo.

Para que el trabajo sea aceptado todos los integrantes del grupo deben estar inscriptos en fecha de final que acordaron en el armado del grupo.

En caso de inscribirse y no entregar en fecha, se calificará como ausente. Si algún miembro del equipo no se presenta en el horario previamente acordado el alumno será calificado como ausente, y deberá conformar otro grupo en otra fecha de final, no afectando la evaluación de los alumnos que se presenten.

## **8. Criterios de Evaluación y Calificación**

El programa debe poder compilarse y ejecutarse en Pampero usando **gcc** con los parámetros **-pedantic -std=c99 -Wall -fsanitize=address**. El último flag verifica que la memoria no sea usada en forma incorrecta (extenderse de la cantidad de bytes reservados en una zona, acceder a una zona de memoria liberada, no liberar memoria antes de finalizar la ejecución, etc.).

**Recomendamos utilizar todos los flags desde el principio del desarrollo para encontrar y resolver antes los posibles errores que surjan. Es útil agregar además el flag de debug -g para obtener más información sobre los errores encontrados por el -fsanitize.**

No se aceptará el uso de bibliotecas de terceros, a excepción de la biblioteca estándar de C. El código debe ser íntegramente de autoría propia. El uso de bibliotecas no autorizadas implicará la desaprobación.

Si un trabajo presenta errores de compilación, el mismo será reprobado. Se espera que no se presenten “warnings” evitables.

**Para la evaluación y calificación del trabajo especial se considerarán:**

- el cumplimiento del modo de ejecución y de los nombres de archivos pedidos
- el correcto funcionamiento del programa
- la modularización realizada
- el correcto diseño del TAD, que debería poder ser fácilmente adaptable para otras codificaciones, nombres o formatos de archivos
- la separación del *front-end* y el *back-end*
- la claridad del código
- el cumplimiento de las reglas de estilo de programación dadas en clase
- la eficiencia del procesamiento de los datos y el uso de la memoria.

## 9. Dudas sobre el TPE

Si bien el enunciado contempla la funcionalidad completa a desarrollar es normal que surjan dudas acerca de cómo interpretar ciertos casos. O que una consigna genere más de una posible solución, por lo que es importante que analicen bien el enunciado, y ante cualquier duda pregunten. Sólo se contestarán dudas sobre el enunciado. Las mismas deben volcarse al **Foro de Discusión “TPE Final” del Campus ITBA.**