PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

PROGRAMACIÓN 2
2da práctica (tipo b)
(Primer Semestre 2025)

Indicaciones Generales:
Duración: **110 minutos.**

- No se permite el uso de apuntes de clase, fotocopias ni material impreso.
- No se pueden emplear variables globales, ni objetos (con excepción de los elementos de `iostream`, `iomanip` y `fstream`). No se puede utilizar la clase `string`. Tampoco se podrán emplear las funciones de C que gesten memoria como `malloc`, `realloc`, `memset`, `strdup`, `strtok` o similares, igualmente no se puede emplear cualquier función contenida en las bibliotecas `stdio.h`, `cstdio` o similares y que puedan estar también definidas en otras bibliotecas. No podrá emplear plantillas en este laboratorio.
- Deberá modular correctamente el proyecto en archivos independientes. Las soluciones deberán desarrollarse bajo un estricto diseño descendente. Cada función no debe sobrepasar las 20 líneas de código aproximadamente. El archivo `main.cpp` solo podrá contener la función `main` de cada proyecto y el código contenido en él solo podrá estar conformado por tareas implementadas como funciones. En el archivo `main.cpp` deberá incluir un comentario en el que coloque claramente su nombre y código, de no hacerlo se le descontará 0.5 puntos en la nota final.
- El código comentado no se calificará. De igual manera no se calificará el código de una función si esta función no es llamada en ninguna parte del proyecto o su llamado está comentado.
- Los programas que presenten errores de sintaxis o de concepto se calificarán en base al 40 % de puntaje de la pregunta. Los que no muestren resultados o que estos no sean coherentes en base al 60 %.
- Se tomará en cuenta en la calificación el uso de comentarios relevantes.
- Se les recuerda que, de acuerdo al reglamento disciplinario de nuestra institución, constituye una falta grave copiar del trabajo realizado por otra persona o cometer plagio.
- No se harán excepciones ante cualquier trasgresión de las indicaciones dadas en la prueba.

Puntaje total: 20 puntos

- La unidad de trabajo será `t:\` (Si lo coloca en otra unidad, no se calificará su laboratorio y se le asignará como nota cero). En la unidad de trabajo `t:\` colocará el(los) proyecto(s) solicitado(s).
- Cree allí una carpeta con el nombre "Lab02_2025_1_CO_PA_PN" donde: **CO** indica: Código del alumno, **PA** indica: Primer Apellido del alumno y **PN** indica: primer nombre. De no colocar este requerimiento se le descontará 3 puntos de la nota final.

Cuestionario

- La finalidad principal de este laboratorio es la de reforzar los conceptos contenidos en el capítulo 2 del curso: "Arreglos y punteros". En este laboratorio se trabajará con **memoria dinámica** y los métodos de **asignación de memoria exacta** y **asignación de memoria por incrementos**.
- Al finalizar la práctica, comprima la carpeta dada en las indicaciones iniciales empleando el programa `Zip` que viene por defecto en el Windows, no se aceptarán los trabajos compactados con otros programas como `RAR`, `WinRAR`, `7zip` o similares.

Para el diseño de su solución, considere que:

- No podrá emplear arreglos estáticos de más de una dimensión.
- No puede manipular un puntero con más de un índice.
- No puede emplear arreglos auxiliares, estáticos o dinámicos, para guardar los datos de los archivos.
- Los archivos solo se pueden leer una vez.

Descripción del caso

Se requiere implementar un proyecto en C++ que permita gestionar los siguientes archivos de texto: `conductores.csv` (DNI y nombre), `faltas.csv` (DNI del conductor infractor, placa, fecha, código de la infracción) e `infracciones.csv` (código de la infracción, descripción, tipo,valor de la multa).

`conductores.csv`

```
63736112;ZAMORA ZAVALETA RONAL MANUEL
45043076;VEGA VILCARA CARMELA TERESA
53853385;SOLIS NARVAEZ MARIA MADELEINE
...
```

`faltas.csv`

```
81485316;J5T-691;24/12/2020;128
25518120;E1J-798;14/07/2020;118
58992942;Q6B-243;28/02/2020;102
...
```

`infracciones.csv`

```
101;Adelantar o sobrepasar en forma indebida a otro vehículo.;Grave;316.00
102;No hacer señales ni tomar las precauciones para girar voltear en U.;Grave;316.00
103;Detener el vehículo bruscamente sin motivo.;Grave;316.00
...
```

Tanto el archivo `conductores.csv` como el archivo `infracciones.csv` deberán ser cargados usando las estructuras `Conductores` e `Infracciones` que se presentan a continuación usando memoria dinámica y el método de **asignación de memoria exacta**.

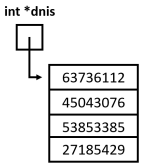
1: `conductores.hpp`

```
struct Conductores {
    int *dnis;
    char **nombres;
    int cantidad;
};
```

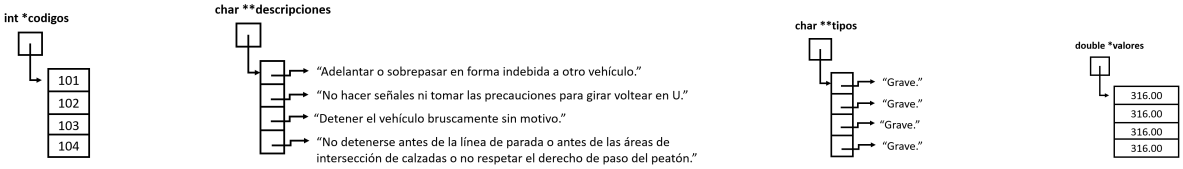
2: `infracciones.hpp`

```
struct Infracciones {
    int *codigos;
    char **descripciones;
    char **tipos;
    double *valores;
    int cantidad;
};
```

Estas estructuras utilizan arreglos paralelos. En el caso de la estructura `Conductores` se utilizan dos arreglos paralelos, uno para almacenar los DNIs (campo `int *dnis`) y otro para almacenar los nombres de los conductores (campo `char **nombres`). El campo `cantidad` indica la cantidad de elementos que poseen ambos arreglos paralelos.



En el caso de la estructura `Infracciones` se utilizan cuatro arreglos paralelos, el primero para almacenar los códigos de las infracciones (campo `int *codigos`), el segundo para almacenar las descripciones (campo `char **descripciones`), el tercero para almacenar los tipos de infracción (campo `char **tipos`) y el último para almacenar el valor de la multa asociado a la infracción (campo `double *valores`). El campo `cantidad` indica la cantidad de elementos que poseen los cuatro arreglos paralelos.



El archivo `faltas.csv` deberá ser cargados usando la estructura `Faltas` que se presentan a continuación usando memoria dinámica y el método de **asignación de memoria por incrementos** iniciando con un tamaño de bloque igual a 0 e incrementando el bloque cada vez que se requiera en 2.

3: `faltas.hpp`

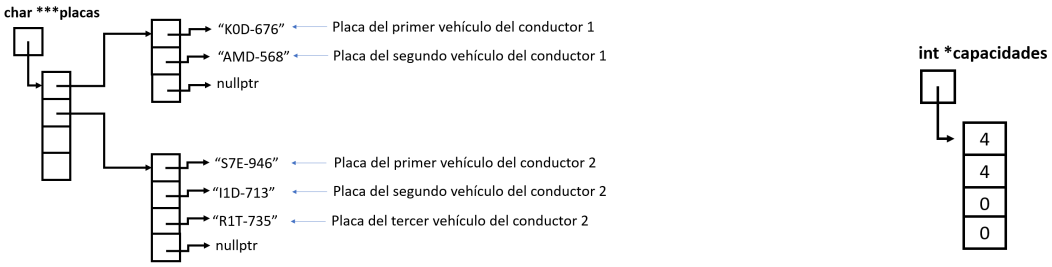
```
struct Faltas {
    Conductores conductores;
    Infracciones infracciones;

    char ***placas;
    int *capacidades;
};
```

La estructura `Faltas` contiene 4 campos. En primero de ellos contiene a todos los conductores (campo `Conductores conductores`), el segundo contiene a todas las infracciones (campo `Infracciones infracciones`), el tercero contiene las placas (campo `char ***placas`) y el cuarto contiene las capacidades de cada bloque de placas (campo `int *capacidades`).

El arreglo `char ***placas` almacena las placas de los diferentes vehículos por los cuales un conductor a cometido una infracción. Como un conductor puede tener varios vehículos, cada entrada (bloque `char**`) corresponde con bloque dinámico de memoria el cual es registrado usando el método por incrementos. El arreglo `char ***placas` se encuentra en paralelo con los arreglos de la estructura `Conductores`. Debido a esto, tanto el arreglo `char ***placas` como el arreglo `int *capacidades` poseen el mismo tamaño y este tamaño corresponde con el tamaño de los arreglos de la estructura `Conductores`.

Para saber la capacidad que posee cada bloque del campo `char ***placas` se usa el cuarto campo `int *capacidades`. Es campo almacena la capacidad de cada bloque. Para saber el tamaño usado de la capacidad, cada bloque de placas de vehículos tiene marcado el valor `nullptr`.



Pregunta 1

Se requiere implementar lo siguiente, asegurando que la gestión de memoria se lleve a cabo utilizando el método de **asignación exacta de memoria**:

- (3.5 puntos) Implemente la sobrecarga del operador `operator+=` de forma tal que la estructura `Conductores` permita cargar los conductores desde el archivo `conductores.csv` usando la siguiente instrucción `conductores += "conductores.csv";` siendo `conductores` del tipo `Conductores`.
- (3.5 puntos) Implemente la sobrecarga del operador `operator+=` de forma tal que la estructura `Infracciones` permita cargar los infracciones desde el archivo `infracciones.csv` usando la siguiente instrucción `infracciones += "infracciones.csv";` siendo `infracciones` del tipo `Infracciones`.

Pregunta 2

Se solicita implementar la función `cargar_faltas_de_los_conductores`, la cual deberá recibir como parámetro una estructura del tipo `Faltas`. La gestión de memoria para los bloques de placas (es decir el bloque de tipo `char **`) debe realizarse utilizando el método de **asignación de memoria por incrementos**, comenzando con un tamaño inicial de bloque igual a 0 e incrementándolo en 2 unidades cada vez que sea necesario. La función deberá realizar lo siguiente:

- (1 punto) Cargar los campos `conductores` e `infracciones` usando las sobrecargas desarrolladas en la pregunta 1 e inicialice los campos `placas` e `capacidades`.
- (1 puntos) Leer cada falta del archivo `faltas.cvs` y ubique la posición del conductor en el arreglo.
- (3 puntos) Cargar las placas de cada vehículo de cada conductor en la posición que le corresponda. Cada placa deberá ser registrada una única vez no importando si es que el vehículo posee más de una infracción.

- (2 puntos) Actualizar el arreglo de capacidades conforme sea requerido.

Pregunta 3

(6 puntos) Se le pide que implemente la función `imprimir_faltas_de_los_conductores` que tome únicamente un parámetro del tipo de estructura `Faltas` e imprima un reporte de los conductores que poseen infracciones, según el siguiente formato:

reporte

```
==================================================
  DNI    CLIENTE                       CANT VEHICULOS
--------------------------------------------------
63736112 ZAMORA ZAVALETA RONAL MANUEL       2
45043076 VEGA VILCARA CARMELA TERESA        3
53853385 SOLIS NARVAEZ MARIA MADELEINE      3
==================================================
```

Profesores del curso: Miguel Guanira, Andrés Melgar, Rony Cueva, Eric Huiza, Erasmo Gómez

Pando, 25 de abril de 2025