PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

PROGRAMACIÓN 2
3ra práctica (tipo b)
(Segundo Semestre 2024)

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 "Lab03_2024_2_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 el método de **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 cuenta con un proyecto en C++ que permite gestionar los siguientes archivos de texto: `platos.csv` (código, nombre, precio y categoría), `repartidores.csv` (código, nombre, tipo de vehículo) y `pedidos.txt` (DNI, código del plato, cantidad de platos, código del repartidor).

`platos.csv`

```
AP-500,CHORIZOS COCKTAIL,12.90,APERITIVO
AP-410,ANTICUCHO,12.90,APERITIVO
...
```

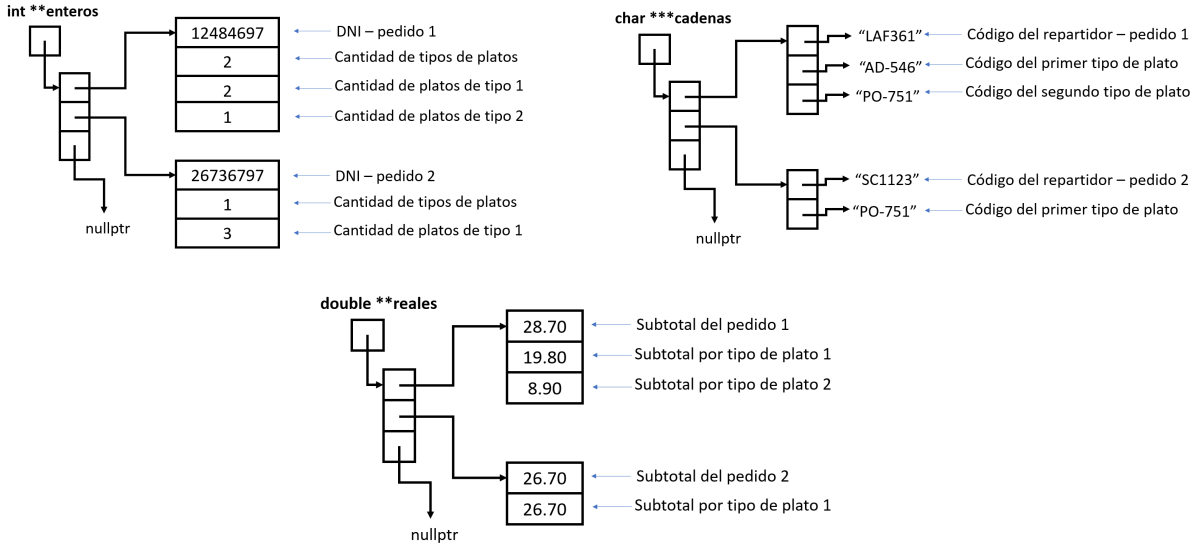
`pedidos.txt`

```
12484697 AD-546 2 LAF361
12484697 PO-751 1 LAF361
...
```

`repartidores.csv`

```
JNV387,Justino Norabuena Virginia Karina,Motocicleta
PRT150,Pairazaman Raffo Tatiana Delicia,Bicicleta
...
```

Con los datos del archivo `pedidos.txt` se han cargado las estructuras `enteros` y `cadenas`. Con la información de archivo `platos.cvs` y las estructuras `enteros` y `cadenas` se ha cargado la estructura `reales`. Estas estructuras se pueden apreciar en la siguiente figura:



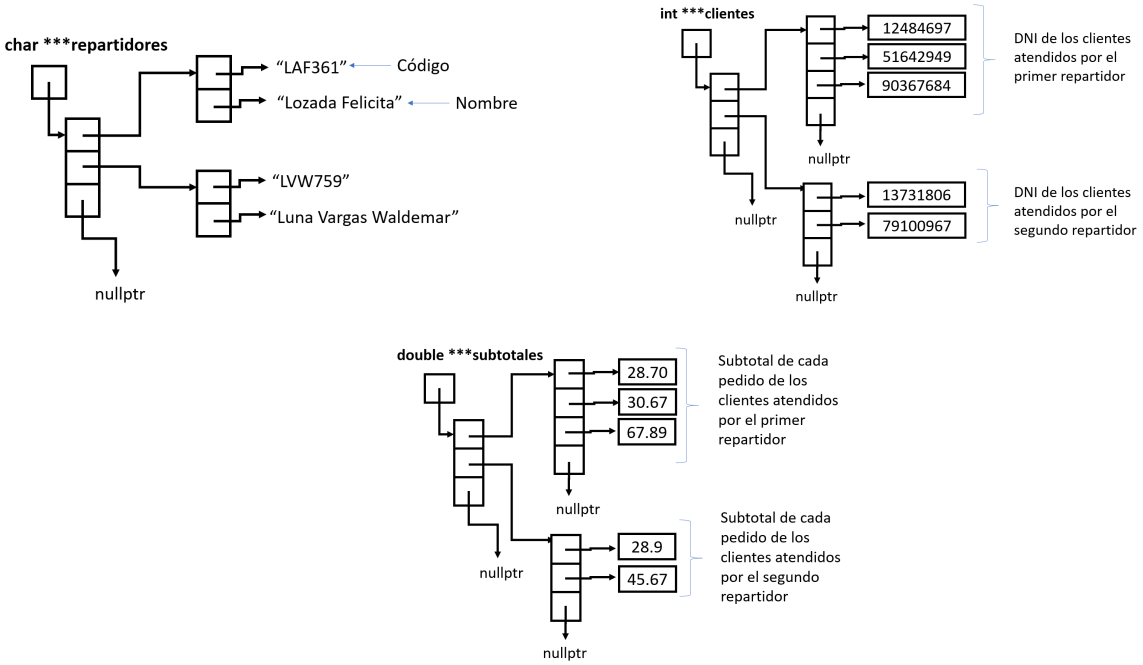
Con esta información, se le solicita completar el proyecto `2024_2_Lab03_Dev` en `Netbeans` cuya función `main` estará compuesta de la siguiente manera. El proyecto mencionado ya cuenta con la implementación del TAD `ConjuntoDePedidos`, así como la función `cargar_conjunto_de_pedidos` y la sobrecarga del operador ++ del TAD en cuestión.

1: `main.cpp`

```
1  #include "ConjuntoDePedidos.hpp"
2  #include "MemoriaPorIncrementos.hpp"
3
4  int main(int argc, char** argv) {
5      ConjuntoDePedidos conjuntoDePedidos; //TAD que almacena los pedidos en las estructuras antes mencionadas
6      cargar_conjunto_de_pedidos(conjuntoDePedidos, "pedidos.txt"); //se cargan los pedidos al TAD
7      conjuntoDePedidos++; //el operador sobrecargado que permite crear la estructura reales, conforme figura anterior,
           internamente carga los platos para determiner el precio del mismo
8
9      int **enteros= conjuntoDePedidos.enteros;
10     char ***cadenas = conjuntoDePedidos.cadenas;
```

```
11      double **reales = conjuntoDePedidos.reales;
12
13      char ***repartidores;
14      int ***clientes;
15      double ***subtotales;
16      cargar_estructuras(repartidores, clientes, subtotales, cadenas, enteros, reales);
17      reporte_de_repartidores("reporte_repartidores.csv", repartidores, clientes, subtotales);
18      reporte_por_repartidor(repartidores, clientes, subtotales);
19      return 0;
20  }
```

Pregunta 1



Se le pide implementar la función `cargar_estructuras` en el proyecto `2024_2_Lab03_Dev` que permite cargas las estructuras `repartidores`, `clientes` y `subtotales`. Los espacios de memoria asignados deberán ser dinámicos, gestionados con el método **asignación de memoria por incrementos** y con incremento en 5 en todos los casos.

- (3 puntos) La función debe incluir el soporte a la estructura `repartidores` considerando que el espacio para el codigo y el nombre del repartidor deberá ser generado dinámicamente. Los repartidores no se pueden repetir en la estructura.
- (3 puntos) La función debe incluir el soporte a la estructura `clientes` considerando que la cantidad de clientes atendidos por cada repartidor es variable. Para determinar el fin de los clientes atendidos, incluya como delimitador de fin de bloque `nullptr`.
- (3 puntos) La función debe incluir el soporte a la estructura `subtotales` considerando que la cantidad de clientes atendidos por cada repartidor es variables. Para determinar el fin de los clientes atendidos, incluya como delimitador de fin de bloque `nullptr`.

Sugerencia

- En caso necesite cargar los repartidores, se recomienda usar el TAD `ConjuntoDeRepartidores` y la función `cargar_conjunto_de_repartidores`.
- En caso necesite buscar el nombre de un repartidor dado su código, se recomienda usar el operador == sobrecargado en la estructura `ConjuntoDeRepartidores`.

Sugerencia (continuación)

- En el archivo `Comunes.cpp` del proyecto en `Netbeans` encontrará algunas funciones implementadas que podrían ser de su interés:
  - Las funciones `mi_strdup`, `retorna_referencia_a_entero`, `retorna_referencia_a_real` permiten "clonar" un dato, asignandole memoria dinámica y retornar dicha referencia al dato clonado.
  - Las funciones `obtener_cantidad_de_cliente` y `obtener_cantidad_de_subtotales` permite calcular la cantidad de clientes y subtotales que existen en un bloque de memoria que finaliza con un `nullptr`.

Pregunta 2

(4 puntos) Implementar la función `reporte_de_repartidores` que genere un archivo `csv` incluyendo el nombre del repartidor, la cantidad de clientes atendidos, así como el total de todos los pedidos. Para la apertura de archivos tanto para leer como para escribir, debe usuar la librería `lib_apertura_archivos_windows.a` que se encuentra en el directorio raíz del proyecto y la definiciones de las funciones que implementa se encuentran en `AperturaDeArchivos.h`. El archivo deberá ser similar a:

`reporte_repartidores.csv`

```
Lozada Arguedas Felicita Valentina,3,383.70
Luna Vargas Waldemar,2,229.00
Zarate Perez Alexander,2,185.20
Mercado Coronel Elia,2,311.50
...
```

Pregunta 3

(5 puntos) Implementar la función `reporte_por_repartidores` que genere un archivo de texto por cada repartidor conforme al siguiente formato. El nombre del archivo será el código del repartidor más la extensión (por ejemplo "LAF361.txt"). El archivo deberá guardarse en el directorio `reportes` que se encuentra en la carpeta raíz del proyeto

`LAF361.txt`

```
============================================================
 CODIGO REPARTIDOR: LAF361
 NOMBRE REPARTIDOR: Lozada Arguedas Felicita Valentina
============================================================
    DNI CLIENTE    MONTO POR PEDIDO
------------------------------------------------------------
     12484697............152
     51642949............149
     90367684............81
============================================================
```

Sugerencia

En el archivo `Comunes.cpp` del proyecto en `Netbeans` encontrará algunas funciones implementadas que podrían ser de su interés. En particular la función `obtener_nombre_archivo_por_repartidor` podría ser usada para obtener el nombre del archivo que debe ser generado en esta pregunta.

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

Pando, 20 de septiembre de 2024