TICS200: App #1

Universidad Adolfo Ibañez

Miguel Solís
miguelandres.solis@edu.uai.cl

Mirko Zitkovich
mirko.zitkovich@edu.uai.cl

Danilo Bórquez Paredes
danilo.borquez.p@uai.cl

26 de febrero de 2024

Objetivos

- Aprender a estructurar un programa en varios archivos fuente de manera ordenada, lógica y modular.
- Utilizar herramientas clásicas de apoyo a la programación como GIT, make, debuggers, etc.
- Utilización de punteros

1. Pizzería

Una reconocida pizzería de Namekusei guardó todas sus ventas en un archivo CSV, el cual disponibilizó a los terícolas normales (excluído Krilin) para su análisis. Ellos desean un análisis simple, donde las métricas son sólo fórmulas matemáticas básicas.

Se le ha pedido a usted, como experto en C y paradigma procedural, que genere un programa capaz de entregar los valores de las métricas solicitadas por nuestra pizzería.

1.1. Requerimientos funcionales del sistema

Su programa consistirá en un ejecutable, que se llamará **app1** (puede llevar la extensión .exe en caso de trabajar en windows). Este programa será ejecutado desde una consola (terminal), con el nombre del archivo de ventas como parámetro. Su contenido serán los datos que corresponden a las ventas de la pizzería. La línea de código que ejecutará su programa se muestra a continuación, donde *ventas.csv* es el archivo en formato *comma separated values*, y *métricaN* corresponde a la métrica seleccionada:

```
TICS200@localhost:~$     ./app1 ventas.csv metrica0 metrica1 ...
```

El formato del archivo será de la siguiente manera:

ventas.csv

```
pizza_id,order_id,pizza_name_id,quantity,order_date,
    ↪ order_time,unit_price,total_price,pizza_size,
    ↪ pizza_category,pizza_ingredients,pizza_name
1.00,1.00,hawaiian_m,1.00,1/1/2015,11:38:36,13.25,13.25,M,
    ↪ Classic,"Sliced Ham, Pineapple, Mozzarella Cheese",The
    ↪  Hawaiian Pizza
2.00,2.00,classic_dlx_m,1.00,1/1/2015,11:57:40,16.00,16,M,
    ↪ Classic,"Pepperoni, Mushrooms, Red Onions, Red Peppers
    ↪ , Bacon",The Classic Deluxe Pizza
3.00,2.00,five_cheese_l,1.00,1/1/2015,11:57:40,18.50,18.5,L,
    ↪ Veggie,"Mozzarella Cheese, Provolone Cheese, Smoked
    ↪ Gouda Cheese, Romano Cheese, Blue Cheese, Garlic",The
    ↪ Five Cheese Pizza
```

Su programa sólo será consultivo acerca de las métricas pedidas por la pizzería
Las métricas a pedir son las siguientes:

1. pizza más vendida (comando por consola: pms)
2. pizza menos vendida (comando por consola: pls)
3. Fecha con más ventas en términos de dinero (junto a la cantidad de dinero recaudado) (comando por consola: dms)
4. Fecha con menos ventas en términos de dinero (junto a la cantidad de dinero recaudado) (comando por consola: dls)
5. Fecha con más ventas en términos de cantidad de pizzas (junto a la cantidad de pizzas) (comando por consola: dmsp)
6. Fecha con menos ventas en términos de cantidad de pizzas (junto a la cantidad de pizzas) (comando por consola: dlsp)
7. Promedio de pizzas por orden (comando por consola: apo)
8. Promedio de pizzas por día (comando por consola: apd)
9. Ingrediente más vendido (comando por consola: ims)
10. Cantidad de pizzas por categoría vendidas (comando por consola: hp)

1.2. Supuestos

1. El usuario siempre ingresará información válida. No es necesario que el programa valide cada texto ingresado por el usuario.
2. No habrán campos vacíos en el archivo de ventas.
3. El orden en que se piden por pantalla las métricas debe ser el orden en que se muestran.

```
TICS200@localhost:~$ ./app1 ventas.csv pms pls
TICS200@localhost:~$ Pizza mas vendida: The Hawaiian Pizza
TICS200@localhost:~$ Pizza menos vendida: The classic Deluxe Pizza
```

1.3. Restricciones

- Debe usar punteros a funciones para cada métrica. La firma del puntero a funciones de la métrica:

```
char*(metrica)(int *size, struct order *orders);
```

  Donde *metrica* es el nombre del puntero a función, *size* es el tamaño del string retornado, y *orders* es el arreglo que contiene todas las ordenes realizadas.

- Debe existir sólo **un puntero a función** para todas las métricas. En su lugar se acepta también un arreglo de punteros a funciones, con el orden en que se irán llamando las funciones.

2. Sobre la entrega

- La tarea debe ser hecha en lenguaje de Programación C.
- Cada grupo puede ser de 1 o 2 personas.
- La tarea se debe entregar el día **Lunes 01 de Abril** a las 23:59.
- Por cada día de atraso se descontará 1 punto, comenzando a las 00:00 horas del siguiente día. Por ejemplo si entrega la tarea a las 00:00 del siguiente día, la nota máxima que puede obtener es un 6.0
- Para la corrección se utilizará un compilador gcc v 5.1 o superior
- La entrega se realiza por la plataforma Webcursos[1]
- El archivo a entregar debe ser un zip que contenga una carpeta en su interior (y sólo una carpeta) con el nombre **tarea1**. Dentro de esa carpeta debe haber un README con la información del grupo y la línea de código (o Makefile[2]) que permite compilar su app.c. Además el nombre de su zip debe ser grupoX-app1.zip, donde X es el número de su grupo.

[1] `http://webcursos.uai.cl`

[2] Este archivo deben generarlo. Información útil pueden encontrarla en `https://stackoverflow.com/questions/1484817/how-do-i-make-a-simple-makefile-for-gcc-on-linux`. Pueden probar su Makefile en un computador con MAC o Linux, o en VSCode en la terminal con UBUNTU.