Algoritmos y Estructuras de Datos II

Parcial 05-05: Tema D - Cálculo de Ventas

Ejercicio 1: Implementación de Tablas de Precios

En el directorio del ejercicio se encuentran los siguientes archivos:

Archivo	***Descripción***
`main.c`	Contiene la función principal del programa
`store.h`	Declaraciones relativas a la estructura de las tiendas y de funciones de carga y escritura de datos.
`store.c`	Implementaciones **incompletas** de las funciones
`array_helpers.h`	Declaraciones / prototipos de las funciones que manejan la tabla de precios
`array_helpers.c`	Implementaciones **incompletas** de las funciones que manejan el arreglo

Abrir el archivo **`input/example1600.in`** para ver cómo se estructuran los datos.

Cada línea contiene los datos de precios de una **tienda de venta de vinos**. El **primer grupo de cuatro columnas** corresponden a precios de los productos, mientras que el **segundo grupo de cuatro columnas** se corresponden a la cantidad de ventas. Finalmente, los últimos dos datos corresponden al ID y número de tienda.

Los productos de vinos se ordenan de la siguiente manera:

- **Syrah** (sh)
- **Malbec** (mb)
- **Cabernet Sauvignon** (cs)
- **Merlot** (mo)

Lo que quiere decir que cada fila tiene el siguiente formato:

<sh>	<mb>	<cs>	<mo>	<sh>	<mb>	<cs>	<mo>	#<ID>-<número>#

Consideraciones:

- El primer grupo de cuatro datos (en el esquema pintados de verde) siempre tiene tipo *price*
- El segundo grupo de cuatro datos (en el esquema pintados de rosa) siempre tiene tipo *selling_amount*
- Por cada tienda, hay una única fila de precios y cantidad de venta.
- Los precios son naturales, mayores que cero.
- Las cantidades de ventas son naturales.

- Los códigos de tienda son representados por un número positivo.
- El número de la tienda siempre estará en el rango 1 a `NO_STORES` en los archivos de entrada.
- La cantidad de tiendas en los archivos de entrada siempre será exactamente `NO_STORES`.
    - Si hay menos datos el programa debería comunicar un error.
    - Queda a criterio del alumno decidir cómo manejar el caso en el cual los archivos de entrada contengan datos de más.

El ejercicio consiste en completar el procedimiento de carga de datos en los archivos **`array_helpers.c`** y **`store.c`**. Los datos deben cargarse de manera correcta en el arreglo multidimensional usando los índices adecuados. Entonces por ejemplo en **`array[6][price]`** se deberían haber cargado los datos de la tienda que en el archivo de entrada figuraba con *número de tienda* (number) igual a 7. Es decir, el arreglo de *stores* que se imprime por pantalla tiene que estar ordenado por número de tienda, y debe tener todos sus datos completos.

Recordar que el programa tiene que ser ***robusto***, es decir, debe tener un comportamiento bien definido para los casos en que la entrada no tenga el formato esperado.

Una vez completada la lectura de datos se puede verificar si la carga funciona compilando,

```
$ gcc -Wall -Werror -Wextra -pedantic -std=c99 -c array_helpers.c store.c main.c
$ gcc -Wall -Werror -Wextra -pedantic -std=c99 array_helpers.o store.o main.o -o
ptable
```

y luego ejecutar por ejemplo

```
$ ./ptable input/example1600.in
```

Ejercicio 2: Análisis de los datos

Completar la siguiente función, definida en **`array_helpers:`**

```
unsigned int max_total_sales(PricesTable ptable);
```

Esta función debe retornar el *máximo total de ventas* de entre todas las tiendas.

El *total de ventas* por tienda se calcula sumando el *subtotal de ventas* de cada producto. El *subtotal de ventas* de cada producto se calcula como:

precio del producto * la cantidad de venta

Por ejemplo, para la siguiente tienda:

40	50	32	35	0	10	85	20	#115-15#

El cálculo del *total de ventas* sería:

(40 * 0) + (50 * 10) + (32 * 85) + (35 * 20) = 3920

Finalmente modificar el archivo **`main.c`** para que se muestre el *máximo total de ventas* de la tabla

Nota: cada archivo de ejemplo incluye el resultado correcto en el nombre, es decir, para el ejemplo **`input/example5606.in`**, el resultado correcto es 5606.