

Algoritmos y Estructuras de Datos II

Parcial 05-05: Tema C - Cálculo de precios

Ejercicio 1: Implementación de Tablas de Precios

En el directorio del ejercicio se encuentran los siguientes archivos:

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

Abrir el archivo `input/example45.in` para ver cómo se estructuran los datos.

Cada línea contiene los datos de precios de una tienda de venta. Los primeros dos datos corresponden al código y número de tienda. Luego siguen **dos grupos de cinco columnas** (diez columnas en total), las cinco columnas del primer grupo se corresponden a precios de los productos, mientras que las cinco segundas columnas se corresponden a descuentos. Los productos se ordenan de la siguiente manera:

- **Potatoes** (p): Papas
- **Cabbages** (cb): Repollos
- **Carrots** (ca): Zanahorias
- **Onions** (o): Cebollas
- **Radishes** (r): Rábanos

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

<code>_<código>_<número>_</code>	<code><p></code>	<code><cb></code>	<code><ca></code>	<code><o></code>	<code><r></code>	<code><p></code>	<code><cb></code>	<code><ca></code>	<code><o></code>	<code><r></code>
--	------------------------	-------------------------	-------------------------	------------------------	------------------------	------------------------	-------------------------	-------------------------	------------------------	------------------------

Consideraciones:

- El primer grupo de cinco datos (en el esquema pintados de verde) siempre tiene tipo *price*
- El siguiente grupo de cinco datos siempre tiene tipo *discount*
- Por cada tienda, hay una única fila de precios y descuentos.
- Los precios son naturales, mayores que cero.
- Los descuentos son naturales entre 1 y 99.
- Los códigos de tienda son representados por un carácter. El número de la tienda siempre estará en el rango 1 a `STORES` en los archivos de entrada.
- La cantidad de tiendas en los archivos de entrada siempre será exactamente `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 usando los índices adecuados. Entonces por ejemplo en `array[4][price]` se deberían haber cargado los datos de los precios de la tienda que en el archivo de entrada figuraba con *número de tienda* (index) igual a 5. 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/example45.in
```

Ejercicio 2: Análisis de los datos

Completar la siguiente función, definida en `array_helpers`:

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

Esta función debe retornar el *menor precio relativo* de entre todas las tiendas.

El *precio relativo* se calcula sumando el precio de comprar 1 unidad de **cada** producto, aplicando el descuento ofrecido por la tienda **por producto**.

Recuerden utilizar paréntesis para manejar explícitamente la precedencia de las operaciones en esta función.

Por ejemplo, para la siguiente tienda:

_M_22_	10	25	32	15	50	0	10	85	20	7
--------	----	----	----	----	----	---	----	----	----	---

El *precio relativo* calculado sería: **95.8**, pero como `best_relative_price` redondea el resultado de manera truncada (retorna `unsigned int`), el resultado en este caso sería **95**. Tengan en cuenta que los datos de entrada siempre son enteros, por lo que no es complejo lograr el redondeo.

Finalmente modificar el archivo `main.c` para que se muestre el *menor precio relativo* de la tabla

Nota: cada archivo de ejemplo incluye el resultado correcto en el nombre, es decir, para el ejemplo `input/example45.in`, el resultado correcto es 45.