Programación

Práctica iBuy – Aclaraciones y Preguntas Frecuentes

Autores:

M. Paz Sesmero Lorente
Paula de Toledo Heras
Fco. Javier Ordoñez Morales
Juan Gómez Romero
Jose A. Iglesias Martínez
Jose Luis Mira

- Las funciones:

```
leerFicheroUsuarios("usuarios.txt",usuarios, &N_Usuarios
leerFicheroProductos("productos.txt",productos, &N_Productos)

escribirFicheroUsuarios("copia_usuarios.txt",usuarios,N_Usuarios);
escribirFicheroProductos("copia_productos.txt",productos,N_Productos)
```

  solo se llaman una vez. Al principio (lectura) o al final (escritura).

- Función identificarse:
  Prototipo:

```
int identificarse(struct tipo_usuario users[], int N);
   /*Parámetros:
         struct tipo_usuario users[]  Vector de usuarios.
         int N;                       Número de usuarios registrados.
     Valor de retorno:                Identificador del usuario cuyo
                                      login y pasword coinciden con
                                      los introducidos por teclado.
                                      Si el usuario no exite (-1)
                                      Si es administrador (0) */
```

  Llamada desde el programa principal:

```
   int tipoUser=0          //Indica el tipo de usuario
                           //0:Anónimo; 1:Registrado; 2:Administrador
   int id                  //Identificador del usuario del sistema
                           //0:Administrador; [1-30]Registrado.....
   .....
   id=identificarse(usuarios, N_Usuarios);
   if (id==0)
      tipo_user=2;
   else if (id>0)
      tipo_user=1;
```

- Las funciones de búsqueda deben funcionar aún cuando los vectores estén vacios (N=0):

```
while ((encontrado==0) && (i<N)){
   .....
}
y no
do{
  ....
} while ((encontrado==0) && (i<N));
```

- Paso de una cadena como parámetro
  Se trata como un vector

- Antes de añadir elementos a un vector hay que comprobar que no se ha superado el tamaño máximo.

- Toda función que pueda ser reutilizada debe reutilizarse.

- No se pueden usar variables globales. Si una función necesita hacer uso de una variable definida en el programa principal o en otra función, esta variable debe pasarse como parámetro a la función.

- Cuidado con el operador postincremento y los parámetros pasados por referencia. Para que la operación postincremento funcione se requiere usar paréntesis:
  ```
  (*N)++;  //Se aumenta en una unidad lo apuntado por *N.
  ```
  Equivalente a:
  ```
  (*N)=(*N)+1;
  ```
  Distinto a:
  ```
  *N++;  //Aumenta en una unidad el valor de N. Por tanto, tras su ejecución, *N apunta a otra dirección de memoria.
  ```

- Ordenación de un vector de estructuras.
  Análogo a un vector de enteros.
  - La variable **aux** será del tipo correspondiente.
  - La comparación se hace por el campo correspondiente

- Escaparate:
  Para cada producto se mostrará su **identificador**, nombre y precio
  El escaparate se puede gestionar de dos formas:
  - Se crea una vez y luego:
    1. Cada vez que se añade un producto se modifica (si procede) la lista que almacena los últimos artículos puestos a la venta.
    2. Cada vez que se emite un voto Me Gusta o Buen precio se actualiza la lista de productos con más votos de estos tipos
  - Se usa un algoritmo para crearlo y siempre se llama a este algoritmo.
- Descriptor de formato para las variables de tipo `double` **`"%lf"`**. Si se quieren mostrar solo dos decimales "**`%.2lf`**"

- A la hora de trabajar con los vectores hay que tener presente que el usuario/producto con Identificador Id ocupa la **posición Id-1** del vector.

- A la hora de **mostrar más información de un producto** (versión avanzada) hay que **comprobar que se puede hacer.** Un usuario solo puede ver la información de los productos que acaba de visualizar.
  Se recomienda crear un vector
  ```
  int lista[MAX_PRODUCTOS];
  ```
  en el que el valor 1 en la *i-ésima* componente indica que el i-ésimo producto ha sido visualizado por el usuario y por tanto que es posible mostrar más información sobre él. El valor 0 en la *i-ésima* componente indica la situación opuesta.