Programación II

Práctica II - Gestión básica de listas y colas

1. Objetivos

En esta práctica utilizaremos los TAD lista y cola para modelar e implementar una solución a la gestión de colas y cobro de pluses en un supermercado. En la práctica trataremos de:

- Reutilizar los TAD lista y cola, ya implementados y discutidos en clase de teoría.
- Valorar que TAD es el más adecuado para el diseño de las diferentes tareas del programa.

2. Descripción del programa

La compañía XYZ quiere construír un nuevo supermercado en Santiago y desea elaborar un sistema informático para gestionar las colas de clientes y los pagos de pluses a los cajeros. Cada cajero gestiona una cola de clientes que son atendidos por orden de llegada. Cuando un cliente ingresa a la cola debe introducir el número de productos que lleva.

El sueldo de los cajeros pasa ahora a componerse de una parte fija y una parte variable (pluses) en función del número de productos que pasen por su caja, a razón de 0,50 euros de plus por cada producto. En cualquier momento el cajero puede comprobar los pluses que tiene disponibles en ese momento y retirar de la caja una parte o la totalidad de los mismos.

3. Especificaciones básicas del problema

- La cola de clientes debe gestionarse con el TAD cola.
- La lista de los pluses que va obteniendo el cajero con cada cliente debe gestionarse con el TAD lista.

4. Descripción del programa a desarrollar

- El programa presentará un menú con las opciones siguientes:

  1. Anotarse a la cola: en esta opción el cliente debe anotar el número de productos que lleva, y es introducido en la cola. Se deberá comprobar que el número de productos es válido.
  2. Atender cliente: cuando el cajero atiende a un cliente debe notificarlo al sistema, que automáticamente le asigna el cliente que lleva más tiempo en la cola. Además, el plus generado por ese cliente se añade a su lista de pluses (al final de la lista). Se deberá imprimir por pantalla la primera posición de la cola y la lista completa de pluses actualizadas al finalizar la operación.
  3. Cobrar pluses: se permitirá al cajero cobrar parte de los pluses acumulados hasta el momento. Una vez que el cajero introduce la cantidad a retirar y se verifique que esta cantidad está disponible, los pluses se van recaudando por orden: primero se recaudan los pluses de mayor cantidad y luego los más pequeños. Si al ir recaudando los pluses nos pasamos de la cantidad solicitada, se debe modificar el plus correspondiente para permitir cobrar la cantidad exacta. Por ejemplo, si la lista de pluses es [10 20 15 40] y el cajero pretende retirar 65 euros de pluses, se deberían realizar los siguientes pasos:
     a) Retirar 40. El plus acumulado será de 40 euros y la lista actualizada será [10 20 15].
     b) Retirar 20. El plus acumulado será de 60 euros y la lista actualizada será [10 15]
     c) Retirar 15 y dejar 10, modificando ese plus. El plus acumulado será de 65 euros y la lista actualizada será [10 10].

4. Calcular pluses: el cajero puede verificar cuánto dinero tiene disponible de pluses. Obviamente, los pluses que ya se hayan recaudado no se tienen en cuenta, ya que no están en la lista.
5. Salir: se pedirá confirmación antes de abandonar el programa

- La cola de espera se podrá inicializar directamente desde línea de comandos con un conjunto de números enteros que se corresponden con el número de productos que va a comprar cada cliente.
- Cuando se salga del programa se deberá liberar memoria correctamente.
- Cada vez que se modifique la cola de espera, se imprimirá la cantidad de productos del siguiente cliente que será atendido.
- Se recomienda realizar parte del código del archivo main.c en funciones auxiliares:
  - void liberaListaPluses(TLISTA *listaPluses): libera la memoria de la lista.
  - void liberaColaEspera(TCOLA *colaEspera): libera la memoria de la cola.
  - void imprimirCola(TCOLA colaEspera): imprime el primer elemento de la cola.
  - void imprimirListaPluses(TLISTA listaPluses): imprime toda la lista de pluses.
  - int comprobacionNumeroProductos(TIPOELEMENTOCOLA n): comprueba si el número de productos es válido (1) o no (0).
  - int comprobacionValorARecaudar(TIPOELEMENTOLISTA v): comprueba si el valor del plus a cobrar es válido (1) o no (0). Esta función no evalua si hay importe suficiente de pluses acumulados.
  - TIPOELEMENTOLISTA totalPluses(TLISTA listaPluses): devuelve el total de pluses acumulados.
  - void recaudarPluses(TLISTA *listaPluses, TIPOELEMENTOLISTA dineroARecaudar): permite al cajero cobrar el plus indicado en la variable *dineroARecaudar* si el saldo es suficiente, y actualiza la lista de pluses.

5. Uso de TAD

El programa deberá implementar obligatoriamente la lista y cola indicadas con los TAD correspondientes, cuya descripción y funcionalidades se han explicado en clase de teoría.

Antes de iniciar el desarrollo de la práctica debéis repasar detalladamente las transparencias y notas de teoría de los temas correspondientes y de forma especial los ejemplos, para recordar el funcionamiento del TAD.

Para la lista de pluses se utilizará el TAD lista implementado en los ficheros adjuntos (listas.c y .h). Para la cola de espera se utilizará el TAD cola implementado en los ficheros adjuntos (cola.c y .h).

No será posible realizar ninguna modificación sobre los ficheros de los TAD proporcionados (ni en los .c, ni en los .h).

6. Entregables

Se deberá realizar la entrega en el Campus Virtual. Las fechas de entrega se especifican en el Campus Virtual, y las instrucciones para generar el fichero son las siguientes:

- Se deberá subir un único fichero comprimido con el nombre apellido1_apellido2 y la extensión correspondiente.
- Se incluirá ÚNICAMENTE el fichero main.c y el correspondiente makefile. Para generar el makefile se asumirá que los ficheros de los TAD necesarios estarán en una carpeta denominada TAD, que se encontrará en el mismo directorio que el main.c y el makefile.
- No se incluirá NINGUNA carpeta dentro del fichero comprimido.

Cualquier ejercicio que no compile directamente con el makefile (sin opciones) en Linux será evaluado con la calificación de 0. Este criterio se mantendrá en el resto de prácticas de la asignatura.