Tecnologías de la Información

Universidad de Huelva

GRADO EN INGENIERÍA INFORMÁTICA

ESTRUCTURAS DE DATOS I

# Práctica 2
# Estructuras dinámicas (Listas, Pilas y Colas)

# PELUQUERÍA 2.O

La franquicia de peluquerías "Corte2.0", con grandes salones y altos volúmenes de clientes, trata de buscar un sistema informático que le permita mejorar su gestión y le permita al mismo tiempo un trato especializado al cliente. Para ello se han puesto en contacto con nosotros para que implementemos un sistema informático de atención presencial de los clientes. Debemos desarrollar un software para cada peluquería, que cumpla con los siguientes requisitos:

El sistema deberá contar con una **lista de peluqueros** que prestan su servicio en la peluquería. Para ellos recogemos:

- Código Peluquero
- Apellidos
- Nombre
- Tipo servicio (código numérico de 1 a 3 (viene a representar su tipología de corte/servicio demandado (rasurado, mechas, etc.))
- Cola de personas a atender

Esta *lista de peluqueros* se encuentra ordenada por código de empleado. Cada peluquero dispone de una **cola** de clientes que está a la espera de la atención del peluquero en cuestión.

La *cola de clientes* dispone de la siguiente información del cliente:

- Apellidos
- Nombre
- Edad
- Tipo de servicio
- Hora de llegada

Cuando un nuevo cliente llega a la peluquería, se le realizan las preguntas básicas que permite clasificar el tipo de servicio que desea recibir y dicho cliente pasa a la cola del peluquero que realiza ese servicio, que disponga de una menor cola de espera.

Hay que hacer el sistema ágil para contemplar los siguientes casos: los peluqueros podrían tener que ausentarse por motivos varios (enfermedad, desayuno, asuntos propios, etc). De igual forma, podría darse el caso de que un peluquero se incorporase a trabajar con posterioridad al resto y que podamos "traspasarle" parte del trabajo. Para ello debemos tener presente que debemos desarrollar opciones para:

- *Incorporar un peluquero al servicio*. Para ello se solicita los datos del *peluquero*, se incorpora de forma adecuada a la *lista* y, para que la atención sea adecuada, se le asignan parte de los clientes ya encolados en otros peluqueros. La idea es que del total de clientes en espera de ser atendidos (del tipo de servicio que realiza el peluquero que se incorpora (supongamos que había 17 en espera para un total de 4 peluqueros de ese servicio en cuestión)) se deben repartir con el nuevo peluquero (en este caso se le deben asignar 1/5 de los clientes, retirándose de los peluqueros con más solicitudes).
- *Retirar del servicio a un peluquero*. Para ello los clientes en espera de dicho *peluquero* deben repartirse entre los peluqueros que ofrezcan el mismo servicio que continúen (se debe seguir el criterio de que las colas de espera estén lo más balanceadas posibles). No podrá retirarse a un peluquero cuando éste sea el único que presta ese servicio (esto debe programarse para evitar errores de planificación y que puedan quedarse sin atención especializada los clientes).

La aplicación debe contar con las siguientes funcionalidades:

- *Listado completo de peluqueros, con sus clientes asignados*, en la forma:
  *Peluquero* (apellidos, nombre, código y tipo de servicio*)*
  *Cliente (*apellidos, nombre, edad, tipo de servicio y hora de llegada*)*
  *Cliente (*apellidos, nombre, edad, tipo de servicio y hora de llegada*)*
  ….
  *Peluquero (*apellidos, nombre, código y tipo de servicio*)*
  *Cliente (*apellidos, nombre, edad, tipo de servicio y hora de llegada*)*
  *Cliente (*apellidos, nombre, edad, tipo de servicio y hora de llegada*)*
  ….
  …

- Opción para *volcar la información de la peluquería a fichero* en la forma:

| Nº Peluqueros | Peluquero 1 | Peluquero 2 | … | Nº Clientes | Cliente 1 | Cliente 2 | … |
|---|---|---|---|---|---|---|---|

- Opción para *leer fichero* de la forma antes indicada (si el sistema cae debemos rescatar rápidamente la información desde la copia del sistema)

Tras estas especificaciones, nuestra empresa va a desarrollar un primer prototipo de la aplicación que sirva para ir testeando y avanzar en el desarrollo de la aplicación. Los tipos de datos y clases que se definen para la aplicación son los siguientes:

## Clase cola

```
typedef char cadena[50];
struct cliente
 {
  cadena Apellidos;
  cadena Nombre;
  int Edad;
  int TipoServicio;
  int HoraLlegada; //almacenada en  minutos
 };

class cola
 {
  cliente *elementos; //elementos de la cola
  int inicio, fin; //principio y fin de la cola
  int Tama; //Capacidad de la tabla
  int ne; //Nº de elementos
  public:
  cola(); // constructor de la clase
  ~cola();
  void encolar(cliente e);
  void desencolar();
  bool esvacia();
  cliente primero() ;
  int longitud();
 };
```

## Clase lista

```
struct peluquero
{
  int Codigo;
  cadena Apellidos;
  cadena Nombre;
  int TipoServicio;
  cola Col;
 };
class lista
{
  peluquero *elementos; // elementos de la lista
  int n;         // nº de elementos que tiene la lista
  int Tama;  // tamaño de la tabla en cada momento
  public:
  lista();    // constructor de la clase
  ~lista();  // destructor de la clase
  lista(peluquero &e);
  bool esvacia();
  int longitud();
  //void anadirIzq(peluquero e);    No necesario implementar
  //void anadirDch(peluquero e);   No necesario implementar
  //void eliminarIzq();        No necesario implementar
  //void eliminarDch();       No necesario implementar
  //peluquero observarIzq();    No necesario implementar
  //peluquero observarDch();  No necesario implementar
  //void concatenar(lista l); No necesario implementar
  bool pertenece(peluquero &e);
  void insertar(int i, peluquero &e);
  void eliminar(int i);
  void modificar(int i, peluquero &e);
  peluquero &observar(int i);
  int posicion(peluquero &e);
};
```

## Clase Peluqueria

```
class peluqueria
 {
  lista L; //lista de los peluqueros que están atendiendo a los clientes
 public:
  void AbrirPeluqueria(char *nombrefichero);
  void IncorporarPeluquero(peluquero t);
  bool RetirarPeluquero(int codigo);
  bool EliminarCliente(cadena Nombre, cadena Apelllidos);
  bool IncorporarCliente(cliente cli);
  void Mostrar();
  bool AtenderCliente(int CodigoPeluquero);
  void VolcarPeluqueria(char *nombrefichero);
 };
```

## Explicación de los métodos de la clase peluqueria

- **AbrirPeluqueria:** Método que carga nuestro objeto *peluqueria* con la información existente en el fichero. El fichero sigue el patrón indicado en la página 3. Para ello usamos la siguiente definición:

```
struct peluquerof
 {
  int Codigo;
  cadena Apellidos;
  cadena Nombre;
  int TipoServicio;
 };
```

- **IncorporarPeluquero:** Método que permite incorporar de forma ordenada el peluquero que nos pasan por parámetro. Entendemos que *no se tratará nunca de incorporar un peluquero ya introducido (no debemos contemplar esta funcionalidad)*

- **RetirarPeluquero:** Método que permite retirar el peluquero cuyo código nos pasan por parámetro. Devuelve *false si el peluquero no puede retirarse (por ser el último que queda que ofrezca ese servicio), true si todo correcto.*

- **EliminarCliente:** Este método tiene dos parámetros, nombre y apellidos del cliente que se marcha. Un cliente puede marcharse cuando quiera y este método debe buscar al mismo dentro de la peluquería y eliminarlo de la cola correspondiente.

- **IncorporarCliente:** Método que recibe un cliente y en base a el tipo de servicio que requiere es encolado en el peluquero, que realice ese servicio, que menos clientes tenga a su cargo.

- **AtenderCliente:** Método que procede a eliminar de la estructura al primer cliente en cola del peluquero cuyo *código* nos pasan por parámetro. Devuelve *false* si no se encuentra el peluquero en cuestión.

- **Mostrar:** Permite mostrar por pantalla el conjunto de *peluqueros* y *clientes* que hay actualmente en la peluquería.

- **VolcarPeluqueria:** Método que permite *crear fichero* con la estructura indicada en la página 3 correspondiente a los *peluqueros* y *clientes* en el sistema. Nos apoyamos en:

```
struct peluquerof
 {
  int Codigo;
  cadena Apellidos;
  cadena Nombre;
  int TipoServicio;
 };
```

## Programa principal

El programa contará con un menú con las siguientes opciones:

```
Peluqueria Corte 2.0 Huelva
_____________________________________________________
1 - Leer fichero (rescatar copia)
2 - Insertar peluquero
3 - Insertar cliente
4 - Retirar peluquero
5 - Atender cliente
6 - Mostrar peluqueria
7 - Eliminar un cliente
8 - Volcar a fichero (crear copia)
9 - Salir


Pulse la opcion deseada :
```

Las opciones del menú se corresponden casi directamente con las llamadas a los métodos de la clase *peluqueria*.

## Ficheros proporcionados

Se proporciona el fichero inicial.dat con la siguiente información:

```
Peluquero 1: Bravo Gil, Ana  -Tipo de Servicio:1
Clientes:
       Apellidos            Nombre        Edad   Tipo de Servicio  Hora de llegada
       .........            ......        ....   ................  ...............
      Rivera Puente        Aurelia         20           1            600
      Trinidad Lobo        Alberto         13           1           2745


Peluquero 2: Crespo Pastor, Alfonso  -Tipo de Servicio:2
Clientes:
       Apellidos            Nombre        Edad   Tipo de Servicio  Hora de llegada
       .........            ......        ....   ................  ...............
     Contreras Cano         Isabel         63           2            705


Peluquero 3: Gallego Ferrer, Manuela  -Tipo de Servicio:3
Clientes:
       Apellidos            Nombre        Edad   Tipo de Servicio  Hora de llegada
       .........            ......        ....   ................  ...............
       Ramos Alonso        Carlos         35           3            770
        Cruz Prieto          Juan         67           3            810


Peluquero 4: Palacios Rubio, Fernando  -Tipo de Servicio:2
Clientes:
       Apellidos            Nombre        Edad   Tipo de Servicio  Hora de llegada
       .........            ......        ....   ................  ...............
       Moreno Vidal        Josefa         56           2            720
```

## Observaciones

Como maneja objetos con memoria dinámica y tienen sus correspondientes destructores, para evitar efectos secundarios, dado que no ha visto aún la sobrecarga de operadores, deberá diseñar una función genérica que le permita copiar un peluquero en otro. A modo de ejemplo:

**void copiarpeluquero(peluquero &destino,peluquero &origen);**

(Le darán su explicación oportuna en clase práctica)