



Tecnologías de la Información



Universidad de Huelva

GRADO EN INGENIERÍA INFORMÁTICA

ESTRUCTURAS DE DATOS I

Práctica 2

Estructuras dinámicas (Listas, Pilas y colas)

Estando en plena campaña de declaración de la Renta, Hacienda nos pide que implementemos el sistema informático de atención presencial de declaraciones de la renta. Para ello el sistema deberá contar con una **lista** de empleados que actualmente realizan este servicio de atención, compuesta de la siguiente información:

- Código técnico hacienda
- Apellidos
- Nombre
- Especialidad (código numérico de 1 a 3 (viene a representar su especialización en el tipo de declaración de renta (individual, conjunta, con rentas de capital inmobiliario, etc.))
- Cola de declarantes

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

La cola de declarantes dispone de la siguiente información por declarante:

- Dni
- Apellidos
- Nombre
- Edad
- Especialidad
- Hora de llegada

Cuando un nuevo declarante llega a la oficina de Hacienda, es recibido por un técnico que realiza unas preguntas básicas que permite clasificar el tipo de declaración que desea realizar (especialidad) y dicho declarante pasa a la cola del técnico de la especialidad que disponga de una menor cola de espera.

Las personas que trabajan en el servicio de atención al público de la oficina de hacienda no están dedicadas en exclusividad a éstas tareas y pueden ser requeridas en cualquier momento para otros menesteres o incorporarse a la atención en cualquier momento. Para ello debemos tener presente que debemos desarrollar opciones para:

- Incorporar un técnico al servicio. Para ello se solicita los datos del técnico, se incorpora de forma adecuada a la lista y, para que la atención sea adecuada, se le asignan parte de los declarantes ya encolados en otros técnicos. La idea es que del total de declarantes en espera de ser atendidos (de la especialidad a la que pertenezca el técnico que se incorpora (supongamos que había 23 en espera para un total de 4 técnicos de la especialidad en cuestión)) se deben repartir con el nuevo técnico (en este caso se le deben asignar 1/5 de las solicitudes, retirándose de los técnicos con más solicitudes).
- Retirar del servicio a un técnico. Para ello los declarantes en espera de dicho técnico deben repartirse entre los técnicos de la especialidad que continúen (se debe seguir el criterio de que las colas de espera estén lo más balanceadas posibles). No podrá retirarse a un técnico cuando éste sea el último de la especialidad que está atendiendo (esto debe programarse para evitar errores de planificación y que puedan quedarse sin atención especializada los declarantes).

La aplicación a desarrollar debe proporcionar las siguientes estadísticas que pueden ser requeridas en cualquier momento:

- Tiempo medio de espera de los declarantes que actualmente están a la espera de ser atendidos, ya sea con carácter general o por especialidad.

Dichos valores pueden ser usados por los responsables de atención para incorporar nuevos técnicos o poder pasar a otras tareas a su personal.

La aplicación debe contar adicionalmente de las siguientes funcionalidades:

- Listado completo de técnicos, con sus declarantes asignados, en la forma:
 Técnico (apellidos,nombre, código y Especialidad)
 Declarante (apellidos, nombre, dni, especialidad, edad y hora de llegada)
 Declarante (apellidos, nombre, dni, especialidad, edad y hora de llegada)

 Técnico (apellidos,nombre, código y Especialidad)
 Declarante (apellidos, nombre, dni, especialidad, edad y hora de llegada)
 Declarante (apellidos, nombre, dni, especialidad, edad y hora de llegada)

 ...

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

Nº tecnicos	Tecnico 1	Tecnico 2	...	Nº Declarantes	Dec 1	Dec 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 declarante
{
    char Dni[20];
    cadena Apellidos;
    cadena Nombre;
    int Edad;
    int Especialidad;
    int HoraLlegada; /*almacenamos los minutos que representa la hora en cuestión, a modo de
ejemplo las 11:30 serían 11*60 + 30 = 690 */
};

class cola
{
    declarante *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(declarante e);
void desencolar();
bool esvacia();
declarante primero() ;
int longitud();
};
```

Clase lista

```
struct tecnico
{
    intCodigo;
    cadena Apellidos;
    cadena Nombre;
    int Especialidad;
    cola Col;
};

class lista {
    tecnico *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(tecnico &e);
    bool esvacia();
    int longitud();
    //void anadirIzq(tecnico e); No necesario implementar
    //void anadirDch(tecnico e); No necesario implementar
    //void eliminarIzq();No necesario implementar
    //void eliminarDch();No necesario implementar
    //tecnico observarIzq();No necesario implementar
    //tecnico observarDch();No necesario implementar
    //void concatenar(lista l); No necesario implementar
    bool pertenece(tecnico &e);
    void insertar(int i, tecnico &e);
    void eliminar(int i);
    void modificar(int i, tecnico &e);
    tecnico &observar(int i);
    int posicion(tecnico &e);

};
```

Clase oficina

```

class oficina
{
    lista L; //lista de los técnicos que están atendiendo
public:
    void AbrirOficina(char *nombrefichero);
    void IncorporarTecnico(tecnico t);
    bool RetirarTecnico(int codigo);
    int TiempoEspera(int especialidad,int horaactual);
    bool IncorporarDeclarante(declarante dec);
    void Mostrar();
    bool AtenderDeclarante(intCodigoTecnico);
    void VolcarOficina(char *nombrefichero);
};

```

Explicación de los métodos de la clase oficina

AbrirOficina: Método que carga nuestro objeto oficina con la información existente en el fichero. El fichero sigue el patrón indicado en la página 3. Atención=> En el fichero la estructura que se usa para el técnico es la siguiente:

```

struct tecnicof
{
    int Codigo;
    cadena Apellidos;
    cadena Nombre;
    int Especialidad;
};

```

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

RetirarTecnico: Método que permite retirar el técnico cuyo código nos pasan por parámetro. Devuelve false si el técnico no puede retirarse (por ser el último de la especialidad), true si todo correcto.

TiempoEspera: Este método tiene dos parámetros: especialidad que puede ser -1 o un valor válido de 1 a 3 y horaactual. Si especialidad es -1 es que se quiere obtener el tiempo medio de espera de todos los declarantes, con independencia de la especialidad. En caso de ser un valor distinto de -1, se trata de analizar el tiempo medio sólo de los declarantes de dicha especialidad. La hora actual que se recibe como segundo parámetro es un valor entero que corresponde a los minutos pasados desde las 00:00 del día actual. En esencia para pasar la hora actual de las 10:30, nos pasarían $10 \cdot 60 + 30$. El método devuelve el tiempo de espera medio de los declarantes.

IncorporarDeclarante: Método que recibe un declarante y en base a su especialidad es encolado en el técnico de su especialidad que menos declarantes tenga a su cargo. Si no hubiera técnico de su especialidad devuelve false

AtenderDeclarante: Método que procede a eliminar de la estructura al primer declarante en cola del técnico cuyo código nos pasan por parámetro. Devuelve false si no se encuentra el técnico en cuestión.

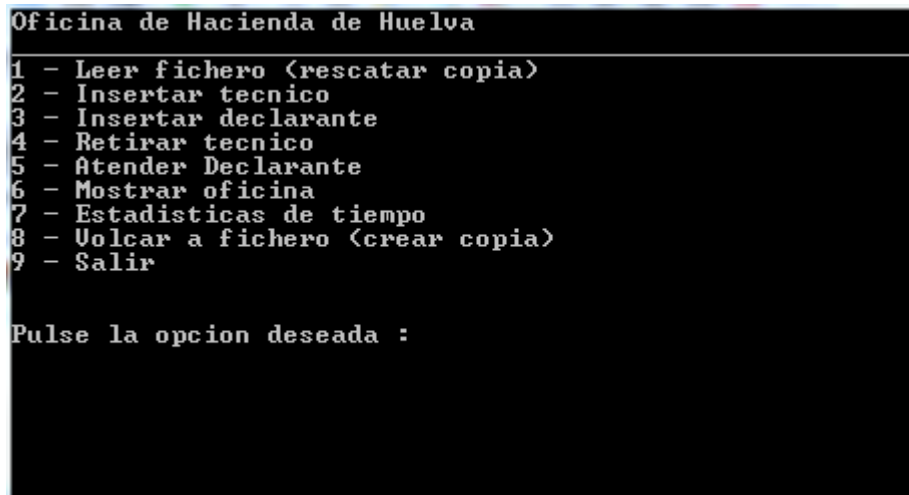
Mostrar: Permite mostrar por pantalla el conjunto de técnicos y declarantes actualmente en el sistema.

VolcarOficina: Método que permite crear fichero con la estructura indicada en la página 3 correspondiente a los técnicos y declarantes en el sistema. Atención=> En el fichero la estructura que se usa para el técnico es la siguiente:

```
struct tecnicof
{
    int Codigo;
    cadena Apellidos;
    cadena Nombre;
    int Especialidad;
};
```

Programa principal

El programa contará con un menú con las siguientes opciones



```
Oficina de Hacienda de Huelva
1 - Leer fichero <rescatar copia>
2 - Insertar tecnico
3 - Insertar declarante
4 - Retirar tecnico
5 - Atender Declarante
6 - Mostrar oficina
7 - Estadisticas de tiempo
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 *oficina*.

Ficheros proporcionados

Se proporciona el fichero inicial.dat con la siguiente información

Tecnico 1	Apellidos	Perez	Nombre: Juan	Especialidad 1	con los declarantes:		
	Apellidos		Nombre:	Dni	Hora Llegada	Edad:	
	Romero	Rafael		29444555	503	67	
	Fdez	Daniel		44888999	525	1	
Tecnico 2	Apellidos	Fdez	Nombre: Jose	Especialidad 2	con los declarantes:		
	Apellidos		Nombre:	Dni	Hora Llegada	Edad:	
	Alvarez	Felipe		31666666	532	63	
Tecnico 3	Apellidos	Rivero	Nombre: Ualle	Especialidad 3	con los declarantes:		
	Apellidos		Nombre:	Dni	Hora Llegada	Edad:	
	Hnez	Rocio		34123123	585	45	
	Colon	Pablo		3456789	595	47	

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 técnico en otro. A modo de ejemplo:

void copiartecnico(tecnico &destino, tecnico &origen);

Le darán su explicación oportuna en clase práctica