## Problemas de Programación 1



## Problemas con vectores de registros

#### Recorridos de vectores de registros

Se va a trabajar con datos de tipo Permiso, definido en el módulo permiso correspondiente a la clase de problemas de la semana pasada. Como referencia, se reproducen a continuación las declaraciones de su fichero de interfaz (las declaraciones junto con sus especificaciones pueden consultarse en el enunciado de la semana pasada, y su implementación está publicada en la web de la asignatura):

```
const int MAX_LONG_NOMBRE = 200;
const int MAX_NUM_MOVIMIENTOS = 200;
const int MESES_NOVEL = 24;

struct Permiso {
    char nombreCompleto[MAX_LONG_NOMBRE];
    int antiguedadMeses;
    int movimientosPuntos[MAX_NUM_MOVIMIENTOS];
    int numMovimientos;
};

void inicializarComoNovel(Permiso& conductorNovel, const char nombre[]);
bool esNovel(const Permiso& p);
int puntos(const Permiso& p);
void registrarSancion(Permiso& p, const int sancion);
void registrarBonificacion(Permiso& p, const int bonificacion);
```

Completa el código de las siguientes funciones que trabajan con vectores de registros del tipo Permiso:

```
/*

* Pre: «v» tiene «n» componentes.

* Post: Ha devuelto el número de permisos de conducir del vector «v» con una

* cantidad de puntos negativa o igual a 0.

*/
int contarSinPuntos(const Permiso v[], const int n);
```

```
/*
 * Pre: «v» tiene «n» componentes.
 * Post: Ha devuelto el permiso de conducir del vector «v» que tiene el menor
 * saldo de puntos.
 */
Permiso peorConductor(const Permiso v[], const int n);
```

```
/*
 * Pre: «v» tiene «n» componentes.
 * Post: Ha devuelto un índice de una componente del vector «v» que contiene un
 * permiso con un «puntosBuscados» puntos, o un valor negativo si no existe
 * ninguno en el vector.
 */
int buscarPorPuntos(const Permiso v[], const int n, const int puntosBuscados;
```

```
/*
 * Pre: «v» tiene «n» componentes.
 * Post: Ha recorrido el vector «v» y ha aumentado la antigüedad de todos los
 * permisos en un mes.
 */
void actualizarMes(Permiso v[], const int n);
```

## Problemas de Programación 1



# Problemas con vectores de registros

```
Pre: «v» tiene «n» componentes.
  Post: Ha recorrido el vector «v» y, cuando ha encontrado permisos
        correspondientes a conductores que han dejado de ser noveles (conductores
         con exactamente 24 meses de antiqüedad, les ha bonificado con 4 puntos.
        Ha devuelto el número de permisos de conductores a los que se ha
        bonificado por dejar de ser noveles.
int bonificarPorDejarDeSerNovel(Permiso v[], const int n);
 * Pre: «v» tiene «nV» componentes y «resultado», al menos «nV» componentes.
 * Post: El vector «resultado» contiene, en sus primeras «nR»
         componentes, únicamente aquellos permisos del vector «v» que tienen un
        saldo de puntos estrictamente positivo.
void purgar(const Permiso v[], const int nV, Permiso resultado[], int& nR);
 * Pre: «v» tiene «n» componentes.
 * Post: Ha devuelto true si y solo si el vector «v» está ordenada de
        forma que los permisos de sus componentes tienen
         valores de puntos no decrecientes.
bool estaOrdenadaPorPuntos(const Permiso v[], const int n);
 * Pre: «v» tiene «n» componentes.
 * Post: Ha devuelto true si y solo si el vector «v» está clasificada de
        forma tal que todos los permisos correspondientes a conductores
        noveles aparecen primero (en las componentes de índices más bajos) y
         todos los correspondientes a conductores experimentados, después (en
         las componentes de índices más altos)
bool estaOrdenadaPorNovel(const Permiso v[], const int n);
 * Pre: «v» tiene «n» componentes.
 * Post: El vector «v» es una permutación de los permisos iniciales del vector «v»
        y están clasificadas de forma que todos los permisos
         correspondientes a conductores noveles aparecen primero (en las
         componentes de índices más bajos) y todos los correspondientes a
         conductores experimentados, después (en las componentes de índices más
         altos).
void clasificararPorNovel(Permiso v[], const int n);
 * Pre: «v» tiene «n» componentes.
 * Post: El vector «v» es una permutación de los permisos iniciales del vector
         «v» y están ordenadas de forma que tienen valores de puntos crecientes.
void ordenarPorPuntos(Permiso v[], const int n);
```