juanka1995
www.wuolah.com/student/juanka1995

930

# MP - 2011 - Septiempre.pdf

*Antiguos*

**1° Metodología de la Programación**

**Grado en Ingeniería Informática**

**Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación**
**UGR - Universidad de Granada**

# Metodología de la Programación

## Examen de teoría. Grado en Ingeniería Informática. Septiembre de 2011.

1. Considere que tenemos almacenada una secuencia **ordenada** de números enteros en una lista de celdas enlazadas definidas con la siguiente estructura:

```
struct Celda {
 int dato;     // Dato en la celda actual
 Celda *sig;   // Puntero al siguiente elemento de la lista
};
```

   *a*) (0.75 puntos) Defina una función que recibe un entero y una lista y modifica dicha lista ordenada insertando el entero en la posición correspondiente.

   *b*) (0.75 puntos) Defina una función que recibe un entero y una lista y modifica dicha lista eliminando la primera aparición de ese entero en la lista.

   Nota: Tenga en cuenta que en ambas funciones se puede dar el caso de que la lista que se pasa esté vacía.

2. Se define una **matriz bilineal simétrica** como una matriz de $n \times n$ enteros en la que todos los elementos significativos (distintos de un valor por defecto) están situados en las 2 diagonales principales y tal que al recorrer ambas diagonales en orden creciente de filas, presentan los mismos elementos. Un ejemplo de este tipo de matrices es la matriz del ejemplo. Es una matriz $6 \times 6$ de valores enteros y con el 0 como valor por defecto.

$$\begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 2 \\ 0 & 4 & 0 & 0 & 4 & 0 \\ 0 & 0 & 7 & 7 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 9 & 0 & 0 & 9 & 0 \\ 6 & 0 & 0 & 0 & 0 & 6 \end{pmatrix}$$

   Se quiere construir la clase **MatrizBS**. Resuelva los siguientes problemas:

   *a*) (0.75 puntos) Definir la parte privada de la clase. Debe minimizarse el uso de memoria, guardando lo estrictamente necesario, para lo que será necesario usar memoria dinámica. Nótese que no se deben guardar los $n \times n$ valores de la matriz, ya que serían valores redundantes.

   *b*) (0.75 puntos) Implementar el constructor por defecto y el destructor. El constructor por defecto creará una matriz de tamaño $4 \times 4$, en la que los elementos de las dos diagonales principales serán todos 1 y el valor por defecto será 0.

   *c*) (0.75 puntos) Implementar un constructor que reciba tres valores: `n` (el número de filas y columnas),un vector de enteros que contiene $n$ elementos correspondientes a los valores en las diagonales y, finalmente, el valor que corresponde a las posiciones fuera de las diagonales. Este último será un parámetro opcional cuyo valor por defecto será cero.

   *d*) (0.75 puntos) Implemente la sobrecarga del operador de asignación de la clase **MatrizBS**.

3. (1.5 puntos) Escribir un programa que reciba como parámetros -en la línea de órdenes- tres nombres de ficheros de texto. Los dos primeros ficheros contienen números reales ordenados en orden creciente y separados por espacios en blanco. El programa tomará los datos de esos ficheros y los irá copiando ordenadamente (de forma creciente) en el tercer fichero, de forma que al finalizar también esté ordenado.

**El tiempo para realizar la parte teórica del examen es de 2 horas**

## Examen de Prácticas. Grado en Ingeniería Informática. Septiembre de 2011.

Definimos una permutación de tamaño $n$ como una secuencia de los $n$ enteros desde el 0 al $n-1$ en un determinado orden. Se desea realizar un programa que genere una permutación de forma aleatoria y la imprima en la salida estándar. Para ello, se propone la creación de la clase `Permutacion` -que se diseña para contener una de estas secuencias- y un programa principal que la use para imprimirla. Concretamente, la solución estará compuesta por los siguientes archivos:

1. `Makefile`. Contendrá las reglas necesarias para crear el ejecutable escribiendo `make`, y para eliminar los archivos intermedios escribiendo `make clean`.

2. `permutacion.h`. Contiene la clase `Permutacion` y las cabeceras de las funciones asociadas al uso de permutaciones, junto con una breve especificación sobre lo que hace cada función.

3. `permutacion.cpp`. Contiene la definición de las funciones del archivo `permutacion.h`.

4. `generar.cpp`. Contiene el programa que hace uso de la clase `Permutacion`. Este programa lee un número entero ($n$) desde la entrada estándar, genera una permutación aleatoria, y la imprime en la salida estándar.

(3 puntos) Implemente los archivos `Makefile`, `permutacion.h`, `generar.cpp` y `permutacion.cpp`.

**El tiempo para realizar la parte práctica del examen es de 1 hora**