Metodología de la Programación

Examen de teoría. Grado en Ingeniería Informática. Julio 2011.

1. (1.5 puntos) Supongamos que en la parte privada de la clase `VectorDinamico` tenemos definido:

```
int NumMaxElems;  // Núm. de casillas reservadas en v
int NumOcupados;  // Núm. de casillas ocupadas
int * v;          // Acceso a los datos
```

de manera que mediante `v` se accede a los datos almacenados en el vector dinámico. Construir el método privado:

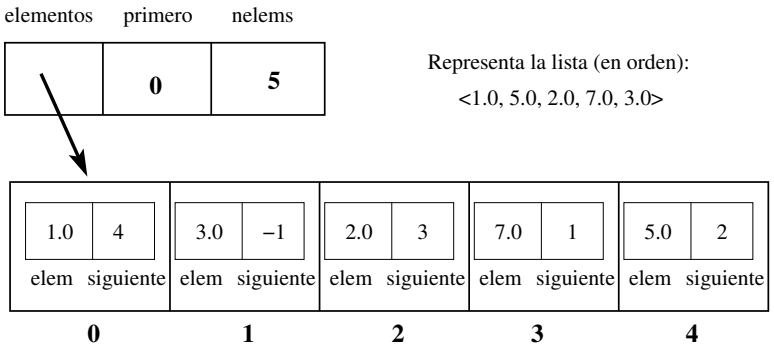
```
bool redimensionar (int nuevotam);
```

que cambia el número de casillas reservadas (`NumMaxElems`) al valor dado por `nuevotam`. El nuevo tamaño puede ser mayor o menor que el que tenía, y el vector debe conservar todos los elementos que sean posibles al cambiar el tamaño. Devuelve un valor booleano indicando si se han conservado todos los elementos "ocupados".

2. Supongamos que tenemos una clase que denominamos `Lista`.

El código que declara la estructura de datos, así como un ejemplo con 5 elementos son:

```
struct dato {
    float elem;
    int siguiente
};
class Lista {
    private:
      dato *elementos;
      int primero;
      int nelems;
    public:
    .......
    .......
};
```



donde el campo `elem` es el elemento de cada posición de una lista de float, y `siguiente` es el índice en el vector **elementos** donde se encuentra el siguiente elemento de la lista (-1 si no hay más elementos).

Para este nuevo tipo de dato que almacena listas de elementos float, implemente los siguientes métodos:

a) (1 punto) El constructor de copia.

b) (1 punto) El operador de asignación.

c) (1.5 puntos) El operador lógico de igualdad `==`. Dos listas son iguales si tienen el mismo número de elementos y éstos están dispuestos en el mismo orden *lógico* (esto es, si al recorrerlas se obtiene la misma secuencia de valores).

d) (1 punto) Un método que recibe un nombre de archivo (a través de un parámetro de tipo cadena estilo C) y almacena la `Lista` en dicho fichero y en formato **binario**. El fichero está compuesto por un `int` que corresponde al número de elementos de la lista (sea n este número), seguido de otro `int` que indica el índice en el vector del primer elemento de la lista, y seguido finalmente por una secuencia de n parejas `float-int`.

e) (1.5 puntos) Un método que, dado el nombre de un archivo con el formato indicado en el punto anterior, cargue el contenido de la lista almacenada en dicho fichero.

3. (2.5 puntos) Escribir un programa similar a `grep` que busque una palabra en una serie de ficheros de texto. La palabra a buscar y los ficheros en los que buscar se proporcionan en la línea de órdenes. Por ejemplo:

```
busca examen fich1 fich2 fich3
```

busca la palabra `examen` en los ficheros `fich1`, `fich2` y `fich3`.
Cada vez que encuentre la cadena buscada, debe indicar el fichero en el que se ha localizado, el número de línea, y la línea completa que la contiene. Un ejemplo de salida de este programa es:

```
fich1 (línea 33): El examen ha sido fácil
fich3 (línea 2): ya te dije ayer que hoy era el examen
fich3 (línea 242): finalmente, el examen tiene tres preguntas
```

Las restricciones que se imponen, y que se deben cumplir en la resolución son:

a) El número de ficheros que se pueden proporcionar es ilimitado.

b) Cada uno de los ficheros sólo puede ser leído una única vez, y no pueden copiarse completos en memoria.

c) Se desconoce a priori el número de líneas de los ficheros.

d) Las líneas de los ficheros tienen una longitud indeterminada, aunque nunca mayor de 500.

Duración del examen: 3 horas.

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 6×6 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 4×4, 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

DECSAI
Departamento de Ciencias de la Computación e I.A.

Metodología de la Programación

Convocatoria de Junio. Curso 2011/2012

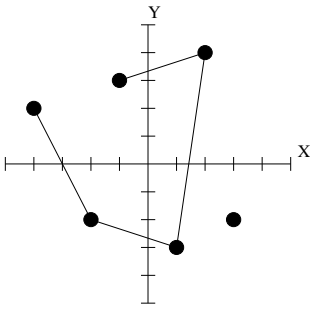
15 de Junio de 2012

Para poder gestionar figuras planas en una aplicación de gráficos 2D deseamos crear una estructura de datos capaz de representar adecuadamente esas figuras. La estructura escogida será la de una línea poligonal construida en base a una serie de puntos. Dispondremos de las clases `Punto` y `PoliLinea`

A continuación mostramos la declaración de ambas clases (sólo la representación interna) y un ejemplo de un punto en las coordenadas $(3, -2)$ y de una polilínea definida por los puntos $(-4, 2)$, $(-2, -2)$, $(1, -3)$, $(2, 4)$ y $(-1, 3)$

```
class Punto {
   int x, y;      // Coordenadas de un punto 2D
   ....
};

class PoliLinea {
   Punto *p;      // Vector de puntos
   int num;       // Número de puntos
   ....
};
```



1. Implemente los siguientes métodos para la clase `PoliLinea`:

   a) (0.5 punto) El constructor sin parámetros (crea una línea poligonal vacía) y el destructor.

   b) (1.5 puntos) El constructor de copia y el operador de asignación.

   c) (1 puntos) El operador de acceso `[]`, que permite acceder (tanto para lectura como para escritura) a un dato de tipo `Punto` en una `PoliLinea`

   d) (2 puntos) Los operadores lógicos de igualdad `==` y desigualdad `!=`. Dos datos `PoliLinea` son iguales si tienen el mismo número de puntos, éstos son iguales y están dispuestos en el mismo orden o en orden inverso (en definitiva, son iguales cuando al representarse gráficamente se obtiene la misma figura).

   e) (2 puntos) Sobrecargar el operador `+` para poder añadir un punto a una línea poligonal de manera que podamos ejecutar operaciones de la forma:

      1) polilínea + punto. Se crea una **nueva** polilínea añadiendo un punto al final de la misma.
      2) punto + polilínea. Se crea una **nueva** polilínea añadiendo un punto al inicio de la misma.

      En ambos casos, la `PoliLinea` inicial **no** se modifica.

   Si fuera preciso implementar algún método para la clase `Punto`, deberá hacerlo.

   Escribir el código correctamente modularizado en ficheros `.cpp` y `.h`.

2. Considere el siguiente formato que permite almacenar una `PoliLinea` en un fichero de texto:

   ```
   POLILINEA
   # Comentario opcional
   N
   X1 Y1
   X2 Y2
   X3 Y3
   ...
   Xn Yn
   ```

   El fichero siempre comienza por la cadena POLILINEA
   El comentario es opcional y, en caso de estar:
   - Comienza por el carácter #
   - Sólo ocupa una línea
   - No hay límite de caracteres

   N es el número de puntos que tiene la polilínea
   Despues de N tenemos la lista de coordenadas de cada punto
   - Cada punto se escribe en una nueva línea y las coordenadas están separadas por un espacio

   a) (2 puntos) Implementar un método para cargar en memoria una `PoliLinea` desde un fichero:
   `void LeerPolilinea (const char *nombre);`

   b) (1 puntos) Implementar un método para escribir en un fichero una `PoliLinea`:
   `void EscribirPolilinea (const char *nombre, const char *comentario=0);`

Duración del examen: 2 horas y media.

DECSAI
Departamento de Ciencias de la Computación e I.A.

Metodología de la Programación

Convocatoria de Septiembre. Curso 2011/2012

11 de Septiembre de 2012

Considere el tipo de dato *Matriz*, diseñado para almacenar una estructura bidimensional de elementos *double*, con la siguiente representación:

```
class Matriz {
      int nfilas;          // Número de filas de la matriz (ejemplo: 4)
      int *ncolumnas;      // Número de columnas de cada fila (ejemplo: 3,2,5,2)
      double **datos;      // Datos de la matriz
    public:
        .......
};
```

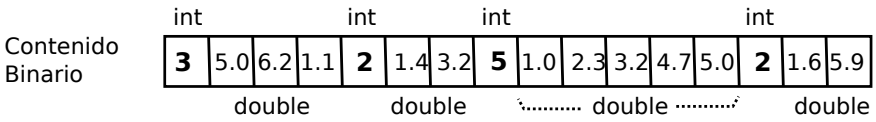
	0	1	2	3	4
0	5.0	6.2	1.1		
1	1.4	3.2			
2	1.0	2.3	3.2	4.7	5.0
3	1.6	5.9			

Observe que este tipo de dato **difiere de las matrices 2D clásicas** en que el número de columnas puede ser distinto para cada fila. Para este tipo de dato, implemente los siguientes métodos de la clase:

1. (0.5 puntos) El constructor sin parámetros (crea una matriz vacía) y el destructor.
2. (1.25 punto) El constructor de copia y el operador de asignación.
3. (0.75 punto) Sobrecargue los operadores de E/S, es decir, `>>` y `<<` para la entrada y salida de una matriz completa en formato de texto.
4. (0.5 puntos) Dos métodos `Get` y `Set`, para acceder o modificar el contenido de una posición, respectivamente.
5. (0.75 punto) Método `Vflip` que devuelva una **nueva** matriz *intercambiando* las filas (primera por la última, segunda por la penúltima, ...)
6. (0.5 puntos) Método `Max` que devuelva la posición (*fila* y *columna*) donde se encuentra el mayor valor de la matriz.
7. (0.75 puntos) Método `Escribir` para salvar un objeto de tipo *Matriz* en un fichero. Este fichero debe tener la siguiente estructura:

```
MP
# Comentario
4
<contenido en binario>
```

Cadena mágica: Dos caracteres seguidos por un separador (salto de línea)
OPCIONAL: Comienza por el carácter '#', hasta salto de línea
Un entero en texto con el número de filas, seguido por un separador
Número de columnas más datos en binario (consulte detalles a continuación)



Observe que después de las tres primeras líneas de texto (dos si no hay comentario), aparece una lista de números en binario. Corresponde a las filas, desde la primera a la última, cada una con un *int* -que indica el número de columnas- seguido de los correspondientes datos *double*.

8. (1 puntos) Método `Leer` que carga una matriz desde un fichero que tiene la estructura indicada en el apartado anterior.

Duración del examen: 2 horas.

DECSAI
Departamento de Ciencias de la Computación e I.A.

Metodología de la Programación

Convocatoria de Septiembre. Curso 2011/2012

Examen práctico

11 de Septiembre de 2012

Una empresa dedicada al transporte de viajeros por carretera dispone de un sistema informático en el que almacena, entre otra información, los datos relativos a los servicios que realiza. Dispone de los siguientes ficheros, que almacenan la información en formato **binario**:

`Conductores`

- Código conductor (alfanumérico, 5 caracteres)
- Apellidos (alfanumérico, 40 caracteres)
- Nombre (alfanumérico, 20 caracteres)
- Día alta (numérico, int)
- Mes alta (numérico, int)
- Año alta (numérico, int)

`Rutas`

- Código de ruta (alfanumérico, 10 caracteres)
- Ciudad origen (alfanumérico, 20 caracteres)
- Ciudad destino (alfanumérico, 20 caracteres)
- Kilómetros (numérico, double)

`Viajes2012`

- Código de conductor (alfanumérico, 5 caracteres)
- Código de ruta (alfanumérico, 10 caracteres)
- Dia viaje (numérico, int)
- Mes viaje (numérico, int)
- Año viaje (numérico, int)

La empresa desea, con urgencia, disponer de una información que debe extraerse de estos ficheros. Para ello realizará unos programas que se ejecutarán desde la línea de órdenes, proporcionando los argumentos necesarios en la llamada a cada programa.

1. Dado el código de un conductor, ¿Cuántos kilómetros recorrió y cuántos viajes hizo?

   `Kilometros cod-conductor`

   Indicar, además de la información solicitada, el nombre completo (*nombre* y *apellidos*) del conductor.
2. Mostrar un listado, con todos los conductores, en el que se incluyan los siguientes datos: código conductor, nombre completo de conductor y número de viajes.
3. ¿Qué ruta ha tenido más conductores?
4. Dado un conductor y un mes, ¿Cuántos kilómetros recorrió ese mes?

   `Kilometros cod-conductor mes`

Duración del examen: 1 hora.

DECSAI
Departamento de Ciencias de la Computación e I.A.

Metodología de la Programación

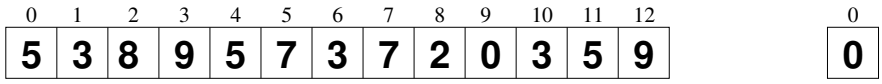
Convocatoria de Junio. Curso 2012/2013

24 de Junio de 2013

Se desea resolver el problema de sumar un número indeterminado de enteros no negativos, con la dificultad añadida de que dichos números pueden llegar a ser muy largos, siendo imposible usar el tipo *int* del lenguaje. Para resolverlo, se propone crear una clase `BigInt` (entero largo) que puede almacenar un entero no negativo de longitud indeterminada.

La clase representará un entero mediante un array -de longitud variable- de objetos de tipo *int*, reservado en memoria dinámica, para poder almacenar todos y cada uno de los dígitos de un entero de longitud indeterminada. Tenga en cuenta que un objeto *int* puede almacenar un rango muy amplio de valores, sin embargo, por simplicidad, será el tipo que usaremos para almacenar cada dígito -del 0 al 9- del *BigInt*.

Además, los dígitos del *BigInt* se almacenarán de forma que el menos significativo -las unidades- se situará en la posición cero del array. Por ejemplo, a continuación se presentan dos ejemplos: los enteros largos 9530273759835 y 0. Observe que para el primer caso se ha reservado y usado un array de objetos *int* de longitud 13 y, para el cero, un array de longitud uno, con el dígito 0.



Considerando este problema, **resuelva las siguientes preguntas**:

1. **(0.25 puntos)** Implemente el constructor sin parámetros y el destructor. Dicho constructor crea un entero largo con valor cero.
2. Implemente dos constructores adicionales:
   a) **(0.75 puntos)** Constructor que crea un *BigInt* a partir de una cadena de caracteres. Esta cadena de caracteres contiene los dígitos del entero largo en formato decimal.
   b) **(1.5 puntos)** Constructor que crea un *BigInt* a partir de un objeto de tipo *unsigned int*.
3. **(1 punto)** Implemente el constructor de copia y operador de asignación.
4. **(2 puntos)** Sobrecargue el operador de suma para que, a partir de dos objetos *BigInt*, se obtenga un nuevo objeto que corresponde al entero largo resultado de su suma.
5. **(1 punto)** Sobrecargue el operador de salida (operador `<<`) para la clase `BigInt` de forma que nos permite escribir el entero largo en un flujo de salida. Tenga en cuenta que deberá presentar todos los dígitos desde el más significativo al menos significativo (escritura habitual de enteros).
6. **(1.5 puntos)** Sobrecargue el operador de entrada (operador `>>`) para la clase `BigInt` de forma que nos permite leer el entero largo desde un flujo de entrada. Tenga en cuenta que deberá leer todos los dígitos desde el más significativo al menos significativo. El operador de entrada debe comportarse de forma similar al caso del tipo *int*, es decir, asumiendo que cada *BigInt* -secuencia de dígitos consecutivos- se encuentra separado de otros datos por "espacios blancos" (espacios, tabuladores, saltos de línea, etc.).
7. **(1 punto)** Considerando todas las operaciones que se han descrito en los puntos anteriores, escriba el correspondiente archivo de cabecera ("bigint.h") teniendo en cuenta que no se incluye ninguna función en línea (*inline*). Es decir, sólo aparecerán cabeceras de funciones, sin su definición.
8. **(1 punto)** Considere que tiene correctamente implementadas las preguntas anteriores. Supongamos que tenemos un archivo con un número indeterminado de enteros largos en formato texto, separados por "espacios blancos". Implemente un programa *"suma.cpp"* que use la clase *BigInt* para leer todos los valores que hay en el fichero y escribir, como resultado final, la suma de todos ellos. Por ejemplo, si el archivo "datos.txt" contiene los siguientes enteros:

```
9347829037470000000000000000
9327887198348931
```

Una posible ejecución del programa podría ser la siguiente:

```
% suma datos.txt
9347829037479327887198348931
```

Duración del examen: 2 horas y media.

Metodología de la Programación

Examen de teoría. Septiembre 2013.

1. (**2.5 puntos**) Considere la siguiente clase:

```
class Entero {
    int *v;
  public:
    Entero (int i=0) { v= new int; *v=i; }
    ~Entero () { delete v; }
    int Set(int i) { *v= i; }
    int Get() const { return *v; }
};
```

   Realice las modificaciones que crea convenientes sobre la clase *Entero* para que el siguiente código funcione correctamente, describiendo brevemente la razón de dichos cambios.

```
Entero Doble(Entero e)
  { return e.Get()*2;  }
int main()
{ Entero x(5), y;
  y= Doble(x);
  cout << "Resultado: " << y << endl;
}
```

2. Se desea crear una clase *Menu* para simplificar el desarrollo de programas que usan menús en modo texto. Para ello, se propone la siguiente representación:

```
class Menu {
    char *titulo; // Encabezado que damos al menú (0 si no existe)
    char **opc;   // Cadenas de longitud variable que describen cada una de las opciones
    int nopc;     // Número de opciones actualmente en el menú
  public:
    ...
};
```

   a) (**1.5 puntos**) Escriba la implementación de las funciones miembro correspondientes al constructor por defecto (crea un menú vacío), destructor, constructor de copias y operador de asignación.

   b) (**1.5 puntos**) Escriba tres funciones:
   - *SetTitulo* que asigne un nuevo valor al título.
   - *GetNumeroOpciones* que devuelva el número de opciones que hay actualmente en el menú.
   - *AddOpcion* que reciba una cadena de caracteres y la añada como una nueva opción después de la última.

   c) (**1.5 puntos**) Sobrecargue los operadores `<<` y `>>` para poder usar el menú para seleccionar una de las opciones. Concretamente:
   - Sobrecargue el operador `<<` para que podamos imprimir el menú. Un ejemplo de llamada podría ser `cout<<menu`, que imprime en la salida estándar el título del menú seguido por cada una de las opciones (numerándolas, del 1 al número de opciones, en líneas distintas).
   - Sobrecargue el operador `>>` para que podamos escoger una nueva opción. La sintaxis de llamada será `menu>>entero`, y provoca la solicitud de una opción (como un número del 1 al número de opciones desde la entrada estándar) de entre las que incluye el menú. El efecto es que imprime el menú en la salida estándar y pide un valor entero de entre las opciones disponibles. Esta petición se repite hasta que el valor es una opción válida.

   d) (**0.5 puntos**) Complete el entorno *Class*, que hemos presentado en la pregunta, con la parte pública correspondiente.

3. (**2.5 puntos**) Escriba un programa *alreves* que recibe en la línea de órdenes el nombre de un fichero, y escribe en la salida estándar el mismo flujo de caracteres, pero en orden inverso. Por ejemplo, si el archivo *abecedario.txt* contiene los caracteres:

```
abcdefghijklmn
ñopqrstuvwxyz
```

   una posible ejecución del programa obtendría lo que sigue:

```
% alreves abecedario.txt
zyxwvutsrqpoñ
nmlkjihgfedcba
```

Duración del examen: 2 horas y 30 minutos.

DECSAI
Departamento de Ciencias de la Computación e I.A.

Metodología de la Programación

Convocatoria de Septiembre. Curso 2012/2013

Examen práctico

20 de Septiembre de 2013

1. El problema

   El objetivo es implementar completamente una solución al problema de calcular la clasificación final de un circuito de n carreras de fondo.

   Cada corredor inscrito participa con el mismo dorsal en todas las pruebas del circuito y puede participar en un número indeterminado de pruebas (podría, incluso, no participar en ninguna).

   El sistema de puntuación es el siguiente: en cada prueba se asigna 1 punto al primer clasificado, 2 al segundo, 3 al tercero, y así sucesivamente. El ganador del circuito es el que **menos** puntos totales obtiene, siempre que complete un número mínimo de pruebas, m. En el caso de completar más pruebas que el número mínimo, se contabilizarán para el total las m mejores clasificaciones.

2. Los datos

   Los **resultados de cada carrera** se almacenan en un fichero *binario* (un fichero por cada prueba). Cada fichero consiste en una secuencia de datos `int` (codificados en binario) que representan los dorsales, por orden de llegada, de los corredores que han terminado la carrera.

   Evidentemente, los ficheros pueden tener diferentes tamaños (el número de corredores que terminan cada prueba es diferente) y los datos estarán en distinto orden (el orden de llegada será diferente).

3. El programa

   La aplicación que se va a desarrollar podría aplicarse en diferentes circuitos, por lo que el número de corredores inscritos, el número de pruebas del circuito y el número mínimo de pruebas a completar no se fijan de antemano.

   El programa se ejecutará:

   ```
   % clasificacion <m> <fich_1> [<fich_2>...<fich_n>]
   ```

   donde:

   - `m` es el número mínimo de carreras que deben completar los inscritos al circuito para poder optar a la clasificación final.
   - `fich_1`,`fich_2`,... son los nombres de los ficheros de resultados, descritos en el punto 2. Observe que el número de ficheros es n (uno por cada carrera del circuito). Se requiere, al menos, un fichero y no se establece un número máximo de ficheros.

   El programa calculará la clasificación del circuito y mostrará en la salida estándar un listado (ordenado por los puntos totales) indicando en cada línea la siguiente información de cada uno de los corredores que han completado el número mínimo de carreras: número de orden, dorsal, número de carreras completadas y puntos totales.

   Escribir un fichero `makefile` que se encargue de generar el ejecutable `clasificacion` a partir de todos los ficheros fuente (`.cpp` y `.h`) que se consideren necesarios.

4. Otras consideraciones

   Para la resolución de este problema se respetarán las siguientes restricciones:

   a) El mínimo dorsal es el 1, pero el máximo no se conoce cuando se ejecuta el programa.
   b) Se valorará especialmente la eficiencia del programa desarrollado.
   c) Se valorará especialmente la modularidad de la solución.

Duración del examen: 1.5 horas.

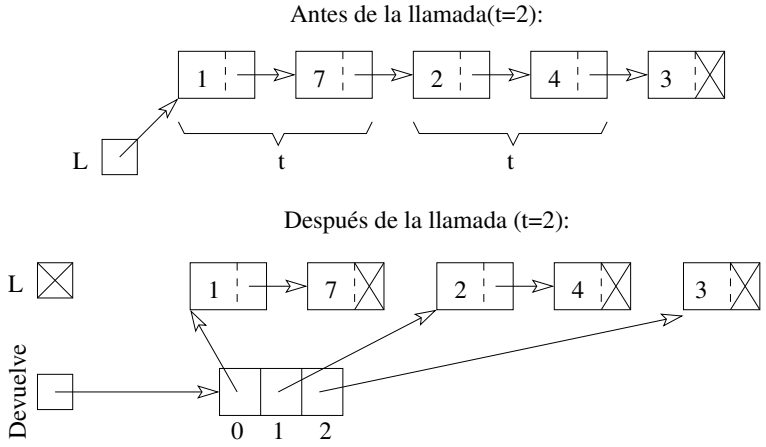
DECSAI
Departamento de Ciencias de la Computación e I.A.

Metodología de la Programación

Convocatoria ordinaria. Curso 2013/2014

7 de Julio de 2014

1. (**1.0 puntos**) Se desea dividir una lista de celdas enlazadas en secuencias de celdas de tamaño t. Escriba una función que recibe una lista de celdas enlazadas que almacena enteros y devuelva un vector de listas que contiene cada una de estas secuencias. La siguiente figura muestra gráficamente el efecto de la llamada:



Observe que en la parte superior se muestra la lista original y el valor `t=2`. La lista, que contiene 5 elementos tendrá que dividirse en 3 trozos. En la parte inferior se muestra el efecto de la llamada. Tenga en cuenta que:

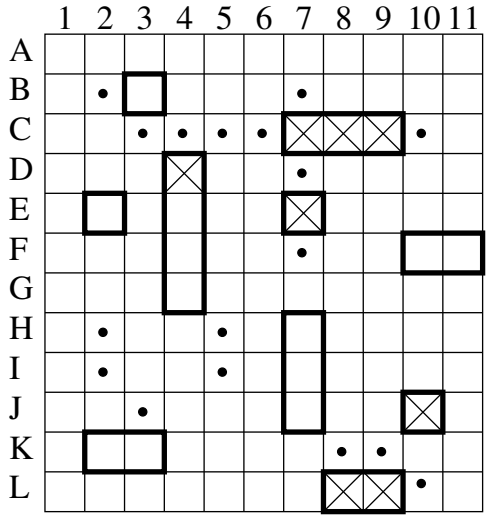
- La lista original queda vacía.
- No es necesario reservar ni liberar ninguna celda.
- La última lista del vector podría quedar con menos de t casillas. En este ejemplo, se queda sólo con 1.
- Será necesario reservar el vector para almacenar cada una de las sublistas. En este caso es un vector con 3 listas.
- Cada lista es una secuencia de celdas enlazadas terminada en el puntero nulo.
- Puede suponer que la lista no está vacía.

Define una estructura adecuada para almacenar cada una de las celdas enlazadas y escriba la función que implementa la operación descrita.

2. Se desea resolver el problema del juego de los barquitos. Para ello, se propone diseñar una clase *Barquitos* que contiene el tablero de un jugador. La clase debe incluir una matriz de enteros para codificar el estado del tablero. El estado se codifica con una estructura de dos dimensiones de tamaño variable, reservada en memoria dinámica.

   En la siguiente figura se presenta una matriz de codificación con enteros, la correspondiente representación gráfica y las indicaciones para entender la representación.

   En este ejercicio debe implementar parte de la clase *Barquitos*. Para realizarlo, comience proponiendo la **representación de la parte privada de la clase**. Una vez establecida la representación, resuelva los siguientes apartados:



	1	2	3	4	5	6	7	8	9	10	11
A	9	9	9	9	9	9	9	9	9	9	9
B	9	–9	1	9	9	9	–9	9	9	9	9
C	9	9	–9	–9	–9	–9	–3	–3	–3	–9	9
D	9	9	9	–4	9	9	–9	9	9	9	9
E	9	1	9	4	9	9	–1	9	9	9	9
F	9	9	9	4	9	9	–9	9	9	2	2
G	9	9	9	4	9	9	9	9	9	9	9
H	9	–9	9	9	–9	9	3	9	9	9	9
I	9	–9	9	9	–9	9	3	9	9	9	9
J	9	9	–9	9	9	9	3	9	9	–1	9
K	9	2	2	9	9	9	9	–9	–9	9	9
L	9	9	9	9	9	9	9	–2	–2	–9	9

Codificación:
Valor negativo: se ha disparado en la casilla
Valor 9: agua
Valor entero i: parte de un barco de i casillas

Ejemplos:
B3: valor 1. Barco de 1 casilla
B2: valor –9. Agua donde se ha disparado
D4: valor –4. Barco de 4 alcanzado
C8: valor –3. Barco de 3 alcanzado

a) (**0.75 puntos**) Implemente el constructor de la clase y el destructor. Este constructor recibe las dimensiones del tablero –filas y columnas– y construye un tablero en memoria dinámica con todas las casillas con agua.

b) (**1.25 puntos**) Implemente el constructor de copias y la sobrecarga del operador de asignación.

c) (**0.75 puntos**) Implemente un método que recibe la posición de un barco y devuelve si es posible colocarlo. La posición de un barco viene determinada por:

- Una casilla: un carácter para codificar la fila (desde la 'A') y un entero para la columna (desde el 1). Puede suponer que no habrá más filas que letras consecutivas en la tabla ASCII.
- El tamaño del barco: un número positivo mayor que cero y menor que 9.
- La dirección de dibujo: 'H' o 'V' indicando horizontal-derecha y vertical-abajo, respectivamente.

Por ejemplo, en la figura anterior, el barco de tamaño 3 hundido (con todas las casillas tachadas) está en la posición: casilla *C7*, de tamaño 3 y dirección H.

Nota: El barco se puede poner si las casillas no están ocupadas y no hay ningún barco con el que "se toque". Por ejemplo, en la figura anterior no se puede poner un barco de tamaño 1 en la posición A2.

d) (**1.0 puntos**) Implemente un método para insertar un barco en el tablero. Este método debe recibir el tamaño del barco e insertarlo en una posición y dirección aleatorias. Supondremos como precondición que seguro que hay algún sitio en el tablero que permite insertarlo.

Para resolverlo suponga que dispone del método del apartado anterior y la siguiente función:

```
int Aleatorio (int min, int max); // un valor aleatorio del intervalo [min,max]
```

Nota: Tenga en cuenta que para generar una dirección no tiene más que generar un valor `Aleatorio(1,2)` *para escoger entre dos posibilidades.*

3. En este ejericio se desea ampliar la clase *Barquitos* con operaciones de E/S. Se desea almacenar el contenido de un tablero en un fichero. El formato de almacenamiento es el siguiente:

- Una cadena "mágica" compuesta por los caracteres *"MP-BARQ-V1.0"* seguida de un salto de línea.
- Un entero (número de filas), un espacio, un entero (número de columnas) y un salto de línea.
- Tantos valores enteros como casillas tiene el tablero codificados en binario.

a) (**0.75 punto**) Implemente un método *Leer* que recibe el nombre de un fichero y lee el contenido del tablero. Devuelve si ha tenido éxito.

b) (**0.5 puntos**) Implemente un método *Escribir* que recibe el nombre de un fichero y guarda el contenido del tablero. Devuelve si ha tenido éxito.

Duración del examen: 2 horas y media.

Metodología de la Programación

Examen de teoría. Convocatoria extraordinaria. Curso 2013/2014

5 de Septiembre de 2014

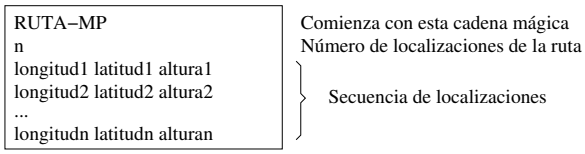
Se desea crear un programa para el análisis de rutas terrestres. Una ruta es una secuencia de 0 o más localizaciones sobre la superficie terrestre. Una ruta con cero localizaciones se considera la ruta *NULA*. Una localización está determinada por la latitud y la longitud. Para resolverlo, se proponen dos nuevos tipos:

```
struct Localizacion {
  double longitud, latitud, altura; // Localización geográfica
};
class Ruta {
  Localizacion *locs; // Secuencia de localizaciones: locs[i]!=locs[i+1]
  int n;              // Números de localizaciones en locs: n>=0
  ...
};
```

Observe que hemos indicado las condiciones que debe cumplir la representación del tipo `Ruta`. Por ejemplo, una ruta de 5 localizaciones tendrá 4 tramos: *0-1*, *1-2*, *2-3* y *3-4*.

1. Implemente las siguientes operaciones:
   a) (0.5 puntos) El constructor sin parámetros y el destructor. El constructor inicializa la ruta como *NULA*.
   b) (1.5 puntos) El constructor de copia y el operador de asignación.
   c) (1 punto) Sobrecargar el operador `+` para poder añadir una localización al final o al principio de la ruta. Deberá implementar dos funciones (observe que, en ambos casos, la `ruta` inicial **no** se modifica):
   - ruta + loc. Se crea una **nueva** ruta añadiendo una localización al final de la misma.
   - loc + ruta. Se crea una **nueva** ruta añadiendo una localización al inicio de la misma.

2. (1.5 puntos) El almacenamiento de una ruta se realiza en formato de texto con el siguiente formato:



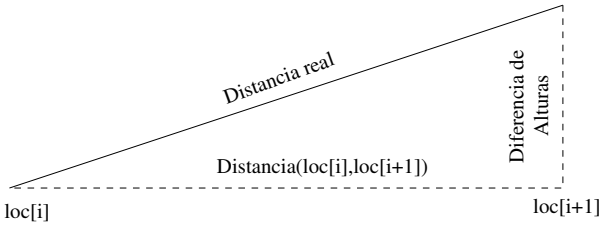
   Implemente dos métodos en la clase `Ruta` para la lectura y escritura, respectivamente. En ambos casos, los métodos reciben el nombre de un archivo y devuelven un booleano que indica si ha tenido éxito.

3. Considere un módulo `ruta.h` donde está implementada la clase anterior y la siguiente operación:

   ```
   double Distancia(const Localizacion& l1, const Localizacion& l2);
   ```

   que corresponde a la distancia entre dos localizaciones. Resuelva lo siguiente:

   a) (0.5 puntos) Implemente un método que devuelve la longitud L de la ruta. La longitud corresponde a la suma de las distancias de los tramos que componen la ruta. Tenga en cuenta que la distancia entre dos localizaciones consecutivas de la ruta corresponde a la distancia real, es decir, debe tener en cuenta el valor devuelto por la función anterior y la diferencia de alturas:



   b) (1 punto) Escriba un programa *longitud.cpp* que se llama desde la línea de órdenes con el nombre de un archivo y usa todo lo anterior para escribir en la salida estándar la longitud de la ruta almacenada en el archivo. Un ejemplo de ejecución sería el siguiente:

   ```
   % longitud excursion.trk
   La longitud de la ruta "excursion.trk" es: 4590.456
   ```

Duración del examen: 2 horas y media.