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

Fundamentos de Programación. Convocatoria de Febrero
Curso 2010/2011
8 de Febrero de 2011

Tiempo: dos horas y media.
IMPORTANTE: Los algoritmos han de ir correctamente explicados.

1. (2.5 puntos) El método de *bisección* usado para calcular de forma aproximada el punto de corte de una función f con el eje de abscisas (la raíz de la función) se puede enunciar como sigue:

   Sea $f(x)$ una función real, estrictamente monótona en $[i, d]$, donde $f(i)$ y $f(d)$ tienen distinto signo y sea ϵ una constante real pequeña (del orden de 10^{-4}, por ejemplo).

   a) Calcular m, el punto medio entre i y d,
   b) Si $|i - d| < \epsilon$, terminar.
   c) Si $f(m)$ tiene igual signo que $f(i)$, repetir considerando el intervalo $[m, d]$
   d) Si $f(m)$ tiene igual signo que $f(d)$, repetir considerando el intervalo $[i, m]$

   Implemente una función **recursiva** que reciba como datos de entrada los extremos i, d del intervalo, así como el valor de ϵ y devuelva la raíz de la función. *Se supone que se dispone de la función f(x) ya implementada en C++.*

2. (2 puntos) Desarrolle un programa para localizar una cadena dentro de otra. El programa leerá dos cadenas desde la entrada estándar, buscará la primera en la segunda, y escribirá en la salida estándar la posición donde se encuentra -en caso de encontrarla- o un mensaje indicando que no se ha localizado. Un ejemplo de ejecución es el siguiente:

```
Introduzca la primera cadena: mundo
Introduzca la segunda cadena: Hola mumumundo
"mundo" Se encuentra en la posición 5  de "Hola mundo"
```

   Tenga en cuenta que no podrá usar ningún método de la clase *string* salvo el acceso a cada uno de los caracteres de las cadenas y al tamaño de las cadenas. En particular, no se puede utilizar el método *find*.

3. Defina la clase `MatrizEnteros` para poder trabajar con una matriz de enteros, de forma que todas las filas tengan el mismo número de columnas.

   a) (0.25 puntos) Definir un método para añadir una fila completa y otro método para acceder a una componente i, j de la matriz.
   b) (1.75 puntos) Definir un método para insertar una fila completa en una posición (de fila) determinada. Por ejemplo,

$$\begin{pmatrix} 3 & 1 & 4 & 5 & 6 \\ 9 & 0 & 4 & 3 & 2 \\ ? & ? & ? & ? & ? \end{pmatrix} \rightarrow \begin{matrix} \text{Insertar} \\ (8,5,0,1,2) \\ \text{en la fila 1} \end{matrix} \rightarrow \begin{pmatrix} 3 & 1 & 4 & 5 & 6 \\ 8 & 5 & 0 & 1 & 2 \\ 9 & 0 & 4 & 3 & 2 \end{pmatrix}$$

   dónde ? representa un valor no utilizado. Se quiere que el módulo sea *robusto*, por lo que deben comprobarse las precondiciones pertinentes.
   c) (1.5 puntos) Definir un método que construya y devuelva una matriz con los datos traspuestos de la primera, es decir,

$$\text{matriz traspuesta}_{ij} = \text{matriz original}_{ji}$$

   d) (2 puntos) Definir un método para sumar aquellos valores M_{ij} de la matriz M que verifiquen que $i + j$ sea igual a:

$$\sum_{h=1}^{k} h^2$$

   para algún valor de $k \geq 1$.

Fundamentos de Programación. Convocatoria de Septiembre. Curso 2010/2011

Recuperación de la parte práctica de Febrero (40 % de la nota final)

Tiempo: una hora y media.
IMPORTANTE: Los algoritmos han de ir correctamente explicados.

Se pide construir un programa que lea una matriz y que calcule la posición de aquel elemento que sea el mayor de entre los mínimos de cada fila. El programa mostrará la posición de dicho elemento (fila y columna). Por ejemplo, dada la matriz M (3×4),

$$\begin{array}{cccc} 9 & 7 & 4 & 5 \\ 2 & 18 & 2 & 12 \\ 7 & 9 & 1 & 5 \end{array}$$

el máximo entre 4, 2 y 1 (los mínimos de cada fila) es 4 que se encuentra en la posición $(0, 2)$. Los datos de entrada al programa se deben dar en el siguiente orden:

1. Número de filas de la matriz.
2. Número de columnas de la matriz.
3. Los elementos de la matriz.

Restricciones del problema:

1. Debe implementar una clase "Matriz". Dicha clase debe incorporar la funcionalidad necesaria para resolver el problema.
2. No se admitirá que la entrada o la salida de datos se realice dentro de la clase que incorpora la funcionalidad para resolver este problema. Esas operaciones se resolverán fuera de la clase.

Ejemplo de fichero de validación:

```
3
4
9  7   4   5
2  18  2   12
7  9   1   5
```

Salida del programa: Dos enteros que representan la fila y columna del elemento buscado:

```
0  2
```

Fundamentos de Programación. Convocatoria de Septiembre. Curso 2010/2011

Recuperación de la parte escrita de Febrero (60 % de la nota final)

Tiempo: dos horas y media.
IMPORTANTE: Los algoritmos han de ir correctamente explicados.

1. (2 puntos) Definir una función **recursiva** a la que se le pasen dos valores de tipo `int` n y k y devuelva como resultado la suma de todos los k dígitos menos significativos de n. Por ejemplo para $n = 61427$ y $k = 3$ el resultado es $7 + 2 + 4 = 13$, y para $n = 23$ y $k = 4$ el resultado es $2 + 3 = 5$.

2. Implemente la clase `Texto` para representar una colección de cadenas de caracteres (`string`). Esta clase debe implementar la siguiente funcionalidad pública:

   Bloque 1 (2 puntos). **Tareas generales**. Escribir métodos para: 1) consultar cuántas cadenas contiene, 2) obtener la cadena i-ésima, 3) añadir una nueva cadena, 4) insertar una cadena en la posición i-ésima, y 5) eliminar la cadena i-ésima. Para todas estas funciones, $0 \leq i < n$ donde n es el número actual de cadenas.

   Bloque 2 (2 puntos). **Tareas de "limpieza"**. Escribir métodos para: 1) eliminar los "blancos" *iniciales* de todas las cadenas, 2) eliminar los "blancos" *finales* de todas las cadenas, y 3) eliminar los "blancos" *iniciales* y *finales* de todas las cadenas.

   Bloque 3 (2 puntos). **Tareas de reorganización**. Puede ser interesante reordenar las cadenas que componen la colección en un momento dado. En lugar de los criterios clásicos (orden lexicográfico, longitud, ...) proponemos emplear otro criterio, basado en lo que llamaremos *índice de ocupación de una cadena*. Un índice de ocupación es un número real entre 0.0 y 1.0, que indica cuántas casillas de la cadena contienen espacios en blanco. El valor 1.0 indica que no hay ninguna casilla en blanco y el valor 0.0 que todas son blancos. El alumno debe escribir un método que modifique el objeto `Texto` de manera que reorganice las cadenas de mayor a menor índice de ocupación.

   Notas:

   a) En todos los casos se deberán comprobar y tratar las situaciones posibles de error.

   b) Se pueden implementar tantos métodos/funciones como considere oportuno para conseguir la solución óptima.

3. (2 puntos)

   Desarrolle un programa que lea una secuencia de números enteros en el rango de 0 a 100 terminada en un número mayor que 100 o menor que 0 y encuentre la subsecuencia de números creciente (ordenada de menor a mayor), de mayor longitud, dentro de dicha secuencia. Supondremos que dos valores iguales consecutivos forman parte de la misma subsecuencia creciente. El programa nos debe decir la posición (empezando desde 1) donde comienza la subsecuencia y su longitud.

Entrada:	23 23 7 45 45 45 73 73 71 4 9 101
Salida:	POSICION = 3 LONGITUD = 6

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

Fundamentos de Programación

Convocatoria de Febrero. Curso 2011/2012
14 de Febrero de 2012

1. (**2.5 puntos**) Defina una función o método **recursivo** para comprobar si una cadena de caracteres es un palíndromo, es decir, que se lee de izquierda a derecha igual que de derecha a izquierda. No se tendrán en cuenta los espacios en blanco, ni se distinguirá entre letra mayúscula y minúscula. Por ejemplo, la siguiente cadena es un palíndromo:

   ```
   (a,h, , ,B, ,c, ,C,b, ,h, ,A)
   ```

   Para resolver este problema, el alumno puede usar el tipo de dato que considere oportuno para almacenar una cadena de caracteres, y puede resolverlo tanto con una función como con un método de una clase, con las siguientes condiciones:

   - El algoritmo **debe ser recursivo**.
   - No se pueden usar bucles ni modificar la cadena de caracteres inicial.

2. (**3 puntos**) Considere un **conjunto** de enteros como una secuencia **ordenada** de números enteros **sin repetidos**. Se desea crear un programa que lea dos conjuntos de enteros y escriba el resultado de calcular la unión y la intersección de ambos conjuntos.

   a) Escriba un módulo que calcule la *unión* de dos conjuntos. Recuerde que en la unión se deben incluir los elementos que estén en cualquiera de ellos.

   b) Escriba un módulo que calcule la *intersección* de dos conjuntos. Recuerde que en la intersección se deben incluir los elementos que sean comunes a ambos conjuntos.

   c) Escriba una función `main` como ejemplo de prueba de ambos módulos. En esta prueba, deberá leer dos conjuntos desde la entrada estándar y escribir el resultado de la unión y la intersección.

   Notas:

   i) Debe implementar cuantas funciones y/o métodos requiera para la correcta resolución del problema.
   ii) No está permitido el uso de ningún algoritmo de ordenación.
   iii) Se valorará la eficiencia de los algoritmos implementados.
   iv) Ambas funciones/métodos tienen como precondición que los conjuntos son correctos, es decir, ya están ordenados y sin repetidos. El resultado debe ser igualmente correcto.
   v) Las funciones/métodos de unión e intersección no contendrán ninguna operación de E/S.

3. Diseñamos la clase **Sopa** para poder manejar *sopas de letras*. Un objeto de la clase **Sopa** almacena $F \times C$ caracteres organizados en F filas por C columnas. Por tanto, es posible recorrer secuencias de caracteres en las 8 direcciones correspondientes a la horizontal, vertical y las dos diagonales. Resuelva las siguientes cuestiones:

   a) (**1 punto**) Defina la representación de la clase. Indique brevemente cómo representa la clase y escriba el código C++ correspondiente.

   b) (**1.5 puntos**) Defina un método *"Traspuesta"* de la clase **Sopa** que nos sirva para obtener una nueva sopa de letras que contiene las mismas palabras. En concreto, tendrá un tamaño de $C \times F$ caracteres de forma que la fila (columna) i-ésima de la nueva sopa corresponderá a la columna (fila) i-ésima de la sopa original.

   c) (**2 puntos**) Defina un método *"Diagonal"* de la clase **Sopa**. Este método tiene como entrada una cadena de caracteres con la palabra a buscar, y obtiene como resultado si se encuenta en alguna de las diagonales, así como la localización de la palabra buscada. Observe que,

   - es posible que la palabra no esté en la sopa, y
   - el método busca la solución sólo en las dos diagonales, es decir, puede encontrarla en 4 direcciones: primera diagonal (abajo-derecha o arriba-izquierda), segunda diagonal (arriba-derecha o abajo-izquierda).

   Importante: Las soluciones pueden incluir llamadas a otros métodos de la clase, en cuyo caso, será **obligatorio** incluir la implementación de dichos métodos. Además, observe que no es necesario leer/escribir ningún dato en la entrada/salida estándar.