Relación de prácticas de la asignatura
METODOLOGÍA DE LA PROGRAMACIÓN
Segundo Cuatrimestre
Curso 2014-2015.

1º Grado en Informática

Práctica 2: Memoria dinámica y Bibliotecas

Objetivos

- Practicar conceptos básicos sobre memoria dinámica. Se hará uso de directivas de compilación para incluir de forma adecuada los ficheros de cabecera.
- Se hará uso de bibliotecas ya creadas y se crearán otras nuevas.

Recomendaciones

- No se podrá hacer uso de memoria estática.
- Dividir todos los ejercicios en varios ficheros y utilizar las directivas de inclusión condicional de código.
- Todas las funciones que se escriban deben tener un comentario de cabecera que contenga, al menos, la información que se indicó en la Práctica 1 sobre punteros.

Distribución temporal

- 2 sesiones de prácticas

¿Qué hay que entregar?

El análisis y diseño de la función de borrado del ejercicio 3.

El análisis consistirá en el estudio del problema que plantea el ejercicio.

- ¿Qué datos de entrada necesita y de qué tipo son?
- ¿Cómo van a llegar esos datos?
- ¿Qué resultado se va a obtener y de qué tipo es?
- ¿Cómo se obtiene el resultado a partir de los datos de entrada?
- ¿Cómo se va a presentar al usuario el resultado final?
- Ejemplo de que la solución propuesta funciona, utilizando los nombres dados a los datos

El diseño incluirá un algoritmo en pseudocódigo o diagrama de flujo que resuelva el problema y que servirá como base para la posterior codificación. Se debe recordar que el diseño es independiente del lenguaje de programación utilizado

¿Cuándo hay que entregar el análisis y el diseño?

Grupo	P3, P5, P6 y P9	P4, P8	P1, P2 y P7
Fecha	10/03/2015	11/03/2015	06/03/2015

Memoria Dinámica

1. Cuestiones sobre punteros y matrices dinámicas

Considérese una matriz dinámica (*float ** tabla*) de 2 x 3 elementos, con los siguientes valores.

{ {1.1, 1.2, 1.3}, {2.1.,2.2, 2.3}}

a) ¿Cual es el significado de *tabla*?
b) ¿Cual es el significado de *(tabla+1)*?
c) ¿Cual es el significado de **(tabla+1)*?
d) ¿Cual es el significado de *(*(tabla+1)+1)*?
e) ¿Cual es el significado de *(*(tabla)+1)*?
f) ¿Cual es el valor de **(*(tabla+1)+1)*?
g) ¿Cual es el valor de **(*(tabla)+1)*?
h) ¿Cual es el valor de **(*(tabla+1))*?

2. Vectores dinámicos

- Escribe una función que reciba un vector dinámico y devuelva otro vector dinámico que contenga sólo los valores impares del vector original.
- Implementa un pequeño programa para probar la función.

3. Gestión de jugadores de baloncesto

- Escribe un programa que permita gestionar los jugadores de baloncesto del equipo de una ciudad.
- Para ello se guardará la información de cada jugador en la siguiente estructura:

```
struct Ficha_jugador
{
        char nombre[50];
        int dorsal;          /* Nº entero */
        float peso;          /* Expresado en kilos */
        int estatura;        /* Expresado en centímetros*/
};
```

- El programa realizará secuencialmente las siguientes operaciones:
  1. Crear un vector dinámico de jugadores.
  2. Listar los jugadores registrados en el equipo, con las características de cada uno de ellos (dorsal, peso, estatura).
  3. Borrar todos los jugadores con una 'a' en su nombre.
  4. Listar de nuevo los jugadores.
  5. Liberar memoria al terminar.

- Se deben codificar las siguientes funciones (también puedes utilizar otras funciones auxiliares que considere oportunas):
  a) Función para **reservar memoria** para un vector de estructuras de jugador.
  b) Función para **leer** un nuevo jugador. La función pedirá al usuario los datos de un jugador y los devolverá en una estructura *struct Ficha_jugador*
  c) Función para **rellenar** un vector de jugadores.

d) Función para **listar** los jugadores del equipo.
e) Función para **borrar** jugadores cuyo nombre contenga un carácter que se pasará como argumento.
   - Al terminar la ejecución, el vector de jugadores habrá reducido su tamaño usando la función *realloc.*
   - La función devolverá el nº de jugadores que ha borrado.
   - Ejemplo: se desea borrar los jugadores cuyo nombre contenga el carácter 'a'

Pablo	4	80.5	192
Luis	5	90.2	201
Antonio	6	112.0	214
Rodrigo	7	85.7	194
Juan	8	93.0	198
Miguel	9	101	205

Vector después de realizar la eliminación de los jugadores con un 'a' en su nombre

Luis	5	90.2	201
Rodrigo	7	85.7	194
Miguel	9	101	205

4. Matrices dinámicas

- Escribe un programa que implemente las siguientes funciones sobre matrices dinámicas y las llame de manera secuencial (no es necesario un menú) mostrando, de manera adecuada, la salida por pantalla.

a) *int ** reservarMemoria (int nFil, int nCol)*.
   - Reserva memoria para una matriz de “nFil” filas y “nCol” columnas.

*b) void rellenaMatriz (int **matriz, int nFil, int nCol).*
   - Función que rellene una matriz con los valores que resulten de sumar el número del índice de la fila (i) y el número del índice de la columna(j).

Ejemplo:

+	0	1	2	3
0	0	1	2	3
1	1	2	3	4
2	2	3	4	5

c) *void imprimeMatriz (int **matriz, int nFil, int nCol)*.
   - Función que imprima una matriz por pantalla.
   - Usa la notación de aritmética de punteros para recorrer la matriz.

d) *int ** eliminarFila (int **matriz, int nFil, int nCol, int fila_borrada)*.
- Función que elimina una fila de la matriz, devolviendo una nueva matriz sin dicha fila.

e) *void liberarMemoria(int ***matriz, int nFil)*.
- Función para liberar la memoria asignada a la/s matriz/ces que haya reservado.

5. Frases

- Codifica un programa que lea varias frases desde el teclado
- La lectura de cada frase finalizará cuando se pulse el carácter de salto de línea o cuando se hayan leído un máximo de 160 caracteres.
- El programa dejará de leer frases cuando se pulse el carácter $.
- Las frases recogidas se almacenarán en un vector de cadenas sin longitud predefinida, es decir, la longitud del vector irá aumentando a medida que el usuario vaya introduciendo cadenas.
- Cuando finalice la obtención de frases (el usuario escribió "$"), se mostrará el vector que contiene dichas frases.
- A continuación se seleccionará los elementos (frases) que ocupan posiciones impares en el vector y se creerá una nueva frase llamada "*cadResult*" con la concatenación de esos elementos, dejando un espacio en blanco entre elemento y elemento.
- **NOTAS**
  - No utilices la función de concatenación predefinida en la biblioteca *string.h*
  - Utiliza las funciones que estimes pertinentes para la resolución del ejercicio, pero al menos debes crear una función que se llame *concatenar* que reciba como parámetros dos cadenas, *cadResult* y la cadena a concatenar.

Bibliotecas

6. Bibliotecas 1

- Crea una biblioteca (*libMatrices.a*) a partir de las cuatro funciones del ejercicio 4 (*reservarMemoria*, *liberarMemoria*, *rellenaMatriz* e *imprimeMatriz*) y su correspondiente fichero de cabecera.
- Reproduce los resultados obtenidos en el ejercicio 4, pero, esta vez, haciendo uso de la biblioteca creada (por tanto, sólo necesitará la función *main*(), la inclusión del .h de la biblioteca y enlazar con ella).

7. Bibliotecas 2

- Implementa una función que permita multiplicar matrices dinámicas. Utiliza las funciones incluidas en la biblioteca *libMatrices.a* para implementar el programa que te permita probar la función.