Relación de prácticas de la asignatura
METODOLOGÍA DE LA PROGRAMACIÓN
Segundo Cuatrimestre
Curso 2021-2022

1º Grado en Informática

Práctica 2: Memoria dinámica, Recursividad, Bibliotecas

Objetivos

- Practicar conceptos básicos sobre memoria dinámica y recursividad. 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 hacer uso de memoria estática.
- Dividir todos los ejercicios en varios ficheros y utilizar las directivas de inclusión condicional de código.

Temporización

3 sesiones de prácticas

Memoria Dinámica

1. Supón una matriz dinámica (*float ** tabla*) de 2x3 elementos, con los siguientes valores.
   { {1.1, 1.2, 1.3}, {2.1.,2.2, 2.3}}
   - ¿Cual es el significado de *tabla*?
   - ¿Cual es el significado de *(tabla+1)*?
   - ¿Cual es el significado de ***(tabla+1)*?
   - ¿Cual es el significado de *(*(tabla+1)+1)*?
   - ¿Cual es el significado de *(*(tabla)+1)*?
   - ¿Cual es el valor de ***(*(tabla+1)+1)*?
   - ¿Cual es el valor de ***(*(tabla)+1)*?
   - ¿Cual es el valor de ***(*(tabla+1))*?

2. Escribe una función que, dado un vector dinámico, su longitud y un número entero (*num*), devuelva dos vectores dinámicos. Uno contendrá los elementos mayores que *num* y otro los elementos menores o iguales que *num*. Implementa un pequeño programa para probar la función.

3. 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:
   a) Crear un vector dinámico de jugadores.
   b) Listar los jugadores registrados en el equipo, con las características de cada uno de ellos (dorsal, peso, estatura).
   c) Borrar todos los jugadores con una '*a*' en su nombre.
   d) Listar de nuevo los jugadores.
   e) Liberar memoria al terminar.

   Deberás implementar al menos las siguientes funciones (también puede utilizar otras funciones auxiliares que considere oportunas):
   - Función para reservar memoria para un vector de estructuras de jugador.
   - 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.*
   - Función para rellenar un vector de jugadores usando la función para leer un jugador.
   - Función para listar los jugadores del equipo.
   - 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 número de jugadores que quedan en el vector.

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 una *'a'* en su nombre:

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

4. Escribe un programa que implemente las siguientes funciones sobre matrices dinámicas y las llame de manera secuencial mostrando, de manera adecuada, la salida por pantalla.
   - *int ** reservarMemoria (int nFil, int nCol).* Reserva memoria para una matriz de *nFil* filas y *nCol* columnas.
   - *void rellenaMatriz (int **matriz, int nFil, int nCol).* Rellena una matriz con valores aleatorios en el intervalo [1,20].
   - *void imprimeMatriz (int **matriz, int nFil, int nCol).* Imprime una matriz por pantalla. Usa la notación de aritmética de punteros para recorrer la matriz.
   - *int * minCol (int **matriz, int nFil, int nCol).* Devuelve un vector dinámico con los mínimos de cada columna.
   - *void liberarMemoria(int ***matriz, int nFil).* Libera la memoria de una matriz reservada dinámicamente. La función pondrá el puntero *matriz* a NULL antes de terminar.

5. Escribe un programa que lea una frase y, a partir de ella, cree un vector dinámico de cadenas con las diferentes palabras de la frase. A partir de este vector, el programa deberá:
   - Calcular la longitud media de las palabras de la frase, así como la longitud mayor y menor (usando paso de parámetros por referencia).
   - Construir un vector dinámico con la frecuencia de aparición de cada longitud.

   Realiza una adecuada modularización, implementando todas las funciones que sean necesarias. Suponer que las palabras de la frase están separadas por un único espacio en blanco.

Bibliotecas y Doxygen

6. 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 del ejercicio 4, pero esta vez haciendo uso de la biblioteca creada (por tanto sólo necesitarás un *main()*, la inclusión del *.h* de la biblioteca y enlazar con ella). Utiliza Doxygen para documentar todas las funciones de la biblioteca.

7. 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.

Recursividad

8. Escribe una solución recursiva que calcule el algoritmo de *Euclides*, usado para calcular el máximo común divisor de dos enteros. El algoritmo de *Euclide*s se describe del siguiente modo:

```
mcd(x, y) = x, si y = 0
mcd(x, y) = mcd (y, mod(x,y)) si y > 0
Antes de llamar a la función, comprobar que  x >= y
```

9. Construye una función recursiva que calcule la división entera de dos números mediante el métodos de restas sucesivas. Implementa un pequeño programa para probarla.

10. Dada una cadena *c,* diseña una función recursiva que cuente la cantidad de veces que aparece un carácter *x* en *c*.
    Ejemplo*: para c = "elementos de programacion" y x = 'e', el resultado es 4*.

11. Codifica una función recursiva que permita sumar los dígitos de un número. Implementa un programa para probarla. Ejemplo: Entrada: 124 Resultado: 7.