

**Relación de prácticas de la asignatura
METODOLOGÍA DE LA PROGRAMACIÓN
Segundo Cuatrimestre
Curso 2017-2018.**

1º Grado en Informática

Práctica 4: Listas, pilas, colas, ordenación, makefiles y aplicaciones avanzadas de punteros

Objetivos

Practicar conceptos básicos sobre estructuras de datos lineales: listas, pilas y colas.

- Se practicará los conceptos de puntero a función y punteros *void*.
- Se implementarán algunos algoritmos de ordenación básicos.
- También se manejará la herramienta *makefile*.

Distribución temporal

- 2 sesiones de prácticas

¿Qué hay que entregar?

- El *makefile* del ejercicio 6

¿Cuándo hay que entregar el makefile?

Grupo	GM1-GM2	GM3 -GM4	GM5
Fecha	15/05/18	16/05/18	18/05/18

Ordenación, punteros a funciones y punteros void *

1. Queremos evaluar las funciones $f(x)$, $g(x)$ y $z(x)$ en todos los valores de x en el intervalo $0 \leq x < N$ con incremento de 0.2

- $f(x) = 3 * e^x - 2x$
- $g(x) = -x * \sin(x) + 1.5$
- $z(x) = x^3 - 2x + 1$

Realiza un programa que:

- a) Solicite al usuario el valor de N .
- b) Solicite la función a evaluar ($f(x)$, $g(x)$ y $z(x)$)
- c) Muestre la evaluación de la función elegida en el intervalo indicado.
 - Utiliza un puntero a función para hacer la llamada a la función.

2. Dada la siguiente estructura:

```
struct Ficha_alumno {  
    char nombre[50];  
    int DNI;  
    float nota;  
};
```

- Escribe un programa que rellene un vector dinámico de tipo *struct Ficha_alumno* y lo ordene mediante el método de ordenación básico que prefieras (selección, inserción o burbuja).
- El vector dinámico se rellenará a partir de los datos de un fichero binario.
- La ordenación se hará usando como campo clave el DNI y podrá ser ascendente o descendente.
- La función de ordenación recibirá como parámetro un puntero a una función de comparación para realizar la ordenación en uno u otro sentido.
- El programa recibirá dos argumentos en la línea de órdenes:
 - El sentido de la ordenación (ascendente o descendente)
 - El fichero binario con los datos para rellenar el vector
- Al terminar el programa, deberá liberar la memoria usada.

3. Escribe un programa en C que lea de un fichero binario un vector dinámico de elementos de tipo *struct Ficha_alumno* (definido en el ejercicio 2) y lo ordene ascendentemente por el campo *nombre* o por el campo *nota* utilizando la función *qsort* de *stdlib.h*

Estructuras de datos dinámicas

4. Polinomio codificado mediante una lista simple

- **Descripción**

- Un polinomio es una expresión algebraica de la forma:

$$a_n x^n + a_{n-1} x^{n-1} + a_{n-2} x^{n-2} + \dots + a_1 x + a_0$$

- A cada $a_i x^i$ se le denomina *monomio*, siendo a_i el coeficiente del monomio e i el exponente del monomio.
- Se denomina *polinomio* a la suma algebraica de varios monomios.
- Algunos ejemplos de polinomios son:

(1) $2x + 3$

(2) $x^3 + 7x^2 + 3x + 9$

(3) $2x^8 + x^3 + 6x$

- Un polinomio se puede representar como una lista enlazada.
 - El primer nodo de la lista representa el primer monomio del polinomio, el segundo nodo el segundo monomio del polinomio, y así sucesivamente.
 - Cada nodo representa un monomio del polinomio y tiene como campo dato el coeficiente del monomio (a) y el exponente (e).
- Escribe un programa que permita:
 - Crear un polinomio. El programa preguntará al principio cuántos monomios tendrá el polinomio.
 - Obtener una tabla de valores de un polinomio para valores de $x = 0.0, 0.5, 1.0, 1.5, \dots, 5.0$
 - Para el polinomio (1) tendríamos la siguiente salida:
 $(x=0.0, 3), (x=0.5, 4), (x=1.0, 5), (x=1.5, 6), \dots, (x=5.0, 13)$
 - Eliminar del polinomio el término con exponente E que se pedirá por pantalla.

- **Objetivo**

- Implementa para ello las siguientes funciones:
 - *anyadeMonomio*. Inserta (por delante) un nuevo monomio en el polinomio.
 - *eliminaMonomio*. Elimina, si existe, el monomio de exponente E (parámetro de la función).
 - *evaluaPolinomio*. Evalúa el polinomio para un valor concreto de x .
 - *muestraPolinomio*. Muestra por pantalla el polinomio.

5. Pilas de contenedores

- **Descripción**

- Para mover los contenedores de mercancía de un importante puerto comercial, se utiliza un método de almacenamiento basado en el concepto de pila.
 - De este modo, el contenedor situado más abajo en la pila fue el primero que se apiló, y, para moverlo, es necesario mover a otra pila todos los contenedores que hay encima de él.
- Cada contenedor de mercancía está identificado por un código entero, X .
- Por motivos de seguridad, como mucho se pueden apilar N contenedores en una misma pila.
 - De este modo, si la pila no está llena, entonces se puede apilar un nuevo contenedor.
- Si se desea sacar un contenedor de código X entonces:
 - Se deben desapilar previamente los contenedores encima de él colocándolos en una nueva pila auxiliar.
 - Se extrae el contenedor X y se vuelven a introducir los contenedores extraídos previamente.

- **Objetivo**

- Codifica un programa que, utilizando las funciones *push* (apilar), *pop* (desapilar), y *vacia* que están implementadas en la biblioteca **pilas.a**, permita gestionar una pila de contenedores con la siguiente funcionalidad:
 - Crear una pila de N contenedores.
 - Listar los contenedores que hay en pila.
 - Se muestra por pantalla un listado de los contenedores contenidos en la pila.
 - Conocer si un contenedor de código X está en la pila.
 - Sacar el contenedor de código X que puede estar en cualquier posición de la pila.
- **NOTA:** no se podrá recorrer **en ningún caso** la pila secuencialmente como si fuera una lista, sino que se hará uso de una pila auxiliar.

Makefiles

6. Proyecto de pasatiempos

- **Descripción**

- Para el desarrollo de un proyecto sobre pasatiempos, se tienen los siguientes ficheros:
 - *reservaMemoria.c*
 - funciones para la reserva de memoria de diferentes estructuras de datos
 - *liberaMemoria.c*
 - funciones para liberar memoria
 - *memoria.h*
 - prototipos de las funciones de reserva y liberación de memoria
 - *ficheros.c – ficheros.h*
 - funciones relacionadas con la E/S de datos en archivos y sus prototipos
 - *crucigrama.c – crucigrama.h*
 - funciones específicas para la creación de crucigramas y sus prototipos
 - *main.c*
 - programa que llama a las funciones de los crucigramas y ficheros
- El resultado final del proyecto será el ejecutable *crucigrama.x* que permitirá la creación de crucigramas.

- **Objetivo**

- Crea un fichero *makefile* con las siguientes características:
 - Construirá una biblioteca (*libMemoria.a*) a partir de los ficheros objetos de *reservaMemoria.c* y *liberaMemoria.c*.
 - Construirá el ejecutable *crucigrama.x* a partir de la biblioteca y los ficheros objetos de *main.c*, *ficheros.c* y *crucigrama.c*
 - Permitirá eliminar los ficheros objetos generados.
 - Para probarlo, puedes utilizar los ficheros que se encuentran en Moodle.