# Práctica No. 9

## Manejo de apuntadores: repaso

**Objetivo:** Crear programas de software donde se almacene la información en memoria secundaria, seleccionando de manera responsable para cada escenario el tipo de archivo que sea adecuado y así garantizar el uso eficiente de los recursos y la integridad de los datos.

**Material:**
- Computadora Personal (PC)
- Programa Editor de texto (ASCII), compilador GCC

**Equipo:**
- Computadora Personal

**Introducción**

## Apuntadores

Un apuntador es una variable que contiene la dirección en memoria de otra variable. Se pueden tener apuntadores a cualquier tipo de variable.

El operador unario dirección & devuelve la dirección de memoria de una variable.

El operador de indirección o dereferencia * devuelve el "contenido de un objeto apuntado por un apuntador".

Para declarar un apuntador para una variable entera hacer:

```
int *apuntador;
```

Se debe asociar a cada apuntador un tipo particular. Por ejemplo, no se puede asignar la dirección de un short int a un long int.

Un apuntador a cualquier tipo de variables es una **dirección en memoria** -- la cual es una dirección entera (en computadoras de 32 bits miden 4 bytes y en computadoras de 64 bits 8 bytes), pero un apuntador no es un entero.

La razón por la cual se asocia un apuntador a un tipo de dato, es porque se debe conocer en cuántos bytes está guardado el dato. De tal forma, que cuando se incrementa un apuntador, se incrementa el apuntador por un "bloque" de memoria, en donde el bloque

está en función del tamaño del dato.

Por lo tanto para un apuntador a un char, se agrega un byte a la dirección y para un apuntador a entero o a flotante se agregan 4 bytes. De esta forma si a un apuntador a flotante se le suman 2, el apuntador entonces se mueve dos posiciones float que equivalen a 8 bytes.

**Desarrollo de la práctica**

Modificar el programa de ejemplo `bubbleSortGenerico.c` para ordenar arreglos de apuntadores a cualquier tipo de dato.

Es importante señalar que en la implementación del ejemplo, se mueven los bloques de memoria referentes a un objeto de cualquier tipo de dato cuando se requiere hacer el proceso de **swap**. En ciertas situaciones no sería muy eficiente ese proceso de copiado, por lo que requeriría una función **swap** que intercambie las **direcciones de memoria** que almacenan los **apuntadores** que conforman el **arreglo**.

Antes de empezar a programar, contestar las siguientes preguntas:

- ¿Qué configuración de apuntadores se requerirían para constituir la lista de elementos a ordenar?
- ¿Cómo sería el prototipo de la función `swap`?
- ¿Cómo sería el prototipo de la función `compare`?
- ¿Cómo sería el prototipo de la función `bubbleSort`?

1. Basándose en la práctica 8, en el programa de anuncios económicos, desarrollar una función para leer todos los anuncios guardados en el archivo y almacenarlos en un arreglo de apuntadores del tipo de dato correspondiente (ej. struct Anuncio).
2. Usando la función `bubbleSort` para ordenar las **direcciones de memoria**, ordenar los elementos contenidos en el arreglo. Desarrollar las funciones para comparar en base a:
    a. Número de anuncio
    b. Clasificación
    c. Responsable
    d. Teléfono
    e. Contenido
    f. Fecha
3. Crear una función para guardar los elementos del arreglo ya ordenados en un archivo, los datos en el archivo deberán estar ordenadas.
4. Responder las siguientes preguntas:
    a. ¿Cuáles son las diferencias en cuanto al proceso de ordenamiento en el ejemplo de la práctica 8 y en esta solución?
    b. En qué situaciones sería más apropiado hacer el ordenamiento directamente sobre el archivo.
    c. En qué situaciones sería más apropiado hacer el ordenamiento en memoria.

## Restricciones

1. El código fuente deberá contener comentarios que indique el objetivo del programa y descripción de instrucciones más relevantes.
2. Evitar nombres de variables y funciones que no reflejen su propósito claramente.
3. Cuando se haga uso de memoria dinámica es importante no olvidar liberar la memoria.
4. La función de ordenamiento bubbleSort deberá estar en una biblioteca creada por el alumno.

## Conclusiones y Comentarios.