UNIVERSIDAD NACIONAL DE INGENIERÍA
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

Laboratorio 04

Punteros I, II

Básico

1. **Mostrar la dirección y el contenido de una variable**:
   Declara un puntero int* p; y haz que apunte a x.
   Muestra en pantalla la dirección de memoria de x y el contenido de esa dirección usando el puntero.

2. **Modificar una variable usando un puntero:**
   Crea una variable float numero = 3.14.
   Crea un puntero float* ptr que apunte a numero.
   Modifica el valor de numero usando el puntero y muestra el nuevo valor.

3. **Simulación de referencias múltiples**
   Crea tres punteros: `p1`, `p2`, `p3`, todos apuntando a una misma variable. Cambia el valor desde `p2` y muestra que el cambio es visible desde `p1` y `p3`. Luego, apunta `p2` a otra variable y analiza los efectos.

4. **Intercambiar valores de dos variables usando punteros:**
   Pide al usuario que ingrese dos números.
   Escribe una función intercambiar(int* a, int* b) que intercambie sus valores usando punteros.

5. **Sumar dos números usando una función con punteros:**
   Escribe una función sumar(int* a, int* b, int* resultado) que reciba dos punteros a enteros y almacene la suma en la dirección apuntada por resultado.

6. **Recorrer un arreglo usando punteros**:
   Declara un arreglo de 5 enteros e inicialízalo. Usa un puntero para recorrer el arreglo e imprimir los elementos sin usar índices.

7. **Contar elementos pares usando punteros**.
   Dado un arreglo de enteros, crea una función que reciba un puntero al arreglo y su tamaño, y que devuelva cuántos números pares contiene.

8. **Mostrar los valores de un arreglo al revés usando punteros:**
   Escribe una función que reciba un puntero a un arreglo y lo imprima en orden inverso sin usar índices.

9. **Diferencia entre valor directo y puntero:**
   Declara una variable int a = 10; y un puntero int* p = &a;
   Muestra la diferencia entre modificar a directamente y hacerlo mediante el puntero p.

10. **Comparar dos arreglos elemento a elemento**:
    Escribe una función que reciba dos arreglos del mismo tamaño y compare sus elementos usando punteros. La función debe retornar true si todos los elementos son iguales.
11. **Eliminar múltiplos de un número en un arreglo:**
    Dado un arreglo de enteros, elimina (lógicamente) los múltiplos de un número dado usando solo punteros. No uses índices. Imprime el resultado sin modificar la memoria original (solo salta esos elementos).

12. **Búsqueda lineal con punteros**
    Implementa una función que recorra un arreglo con punteros y retorne la dirección del primer elemento que sea igual a un valor dado (si existe), o nullptr en caso contrario.

Intermedio:

13. **Contador de caracteres en un char[]**
    Escribe una función que reciba una cadena de caracteres (char[]) y cuente cuántas veces aparece un carácter específico. Usa solo punteros

14. **Suma y promedio usando funciones con punteros**
    Declara un arreglo de 10 enteros. Escribe una función que reciba un puntero al arreglo y su tamaño, y calcule la suma y el promedio. Devuelve ambos valores usando punteros como parámetros por referencia.

15. **Rellenar un arreglo desde una función:**
    Escribe una función que reciba un puntero a un arreglo de n enteros y lo rellene con números aleatorios. Luego imprime el arreglo desde main.

16. **Contar pares e impares usando punteros:**
    Escribe una función que reciba un arreglo y su tamaño, y cuente cuántos números pares e impares hay usando punteros. Los resultados deben ser devueltos a través de argumentos por puntero.

17. **Función que retorne el menor valor (punteros)**
    Escribe una función `int* menorElemento(int* arr, int n)` que retorne un **puntero** al menor elemento del arreglo. Luego imprime su valor y dirección desde `main`.

18. **Ordenar caracteres (arreglo de char)**
    Ordena un arreglo de caracteres (char[]) alfabéticamente usando punteros y un algoritmo de ordenamiento sencillo como inserción o burbuja

19. **Ordenamiento con punteros**
    Reescribe los algoritmos de ordenamiento vistos en clase para un arreglo de enteros usando punteros en lugar de índices. Compara e intercambia valores directamente mediante desreferenciación.

20. **Ordenar con función de comparación personalizada**
    Escribe una función ordenar(int* arr, int n, bool (*comparar)(int, int)) que ordene un arreglo según el criterio definido en la función comparar. Por ejemplo, orden ascendente o descendente.

Avanzado

21. **Matriz como puntero a arreglo**:
    Declara una matriz int matriz[3][4]. Escribe una función que reciba la matriz como puntero a arreglo y que imprima todos los elementos, sin usar índices.

22. **Acceder a matriz con puntero simple:**
    Declara int matriz[3][4]. Usa un puntero int* ptr = &matriz[0][0]; y recorre la matriz como si fuera un arreglo unidimensional usando aritmética de punteros

23. **Arreglo de punteros**
    Crea un arreglo de punteros a int (int* ptrs[5];) que apunten a 5 variables distintas. Modifica las variables a través de los punteros y muestra sus valores originales

24. **Puntero a arreglo**
    Declara un arreglo int numeros[10];. Luego crea un puntero a ese arreglo: int (*ptr)[10] = &numeros;. Accede a los elementos del arreglo usando (*ptr)[i].

25. **Pasar un arreglo a una función usando diferentes tipos de punteros**
    Declara una función que reciba:
    Un puntero simple: int*
    Un puntero a arreglo: int (*)[10]
    Un parámetro tipo arreglo: int arr[]
    Imprime los elementos del mismo arreglo desde cada función y observa diferencias.

26. **Explorar tipos de punteros**
    Declara diferentes tipos de punteros:
    int*, char*, float* Inicializa variables de cada tipo, apunta a ellas y muestra:
    Su dirección
    Su contenido
    Cuántos bytes ocupa cada uno
    Usa sizeof(*ptr) para analizar qué tipo están apuntando.

27. **Simular recorrido bidimensional con arreglo unidimensional**

Declara un arreglo unidimensional de 12 enteros. Simula una matriz de 3x4 accediendo a la posición [i][j] como *(ptr + i*4 + j).

28. **Puntero a función:**
    Escribe dos funciones sumar y restar, y un puntero int (*operacion)(int, int);. Pide al usuario elegir qué operación aplicar a dos números y llama a la función usando el puntero.

29. **Puntero a función como parámetro:**
    Declara una función void aplicar(int* arr, int n, int (*f)(int)); que aplique f a cada elemento del arreglo. Define funciones como cuadrado, doble, negativo, etc. y pásalas como argumento.