# Taller 8. Punteros

| Clase | Lenguaje C |
| --- | --- |

1. **¿Cuál de los métodos de búsqueda, en su opinión, resultó ser el más eficiente y por qué?**

El método 3, ya que es sencillo solo pasar el nombre del vector como argumento y recibir el parámetro con la notación de corchetes, después accediendo a los valores como un vector normal, sin necesidad de usar punteros.

```
printf("Metodo 3\n");
start = clock();
imprimirMatriz_Metodo3(b, M, N);
busquedaLineal_Metodo3(b, M, N, num);
end = clock();
tiempo_m3 += (double)(end - start) / CLOCKS_PER_SEC;
```

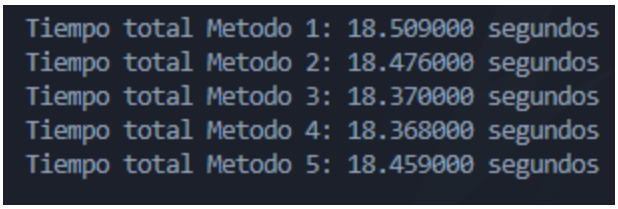
```
void busquedaLineal_Metodo3(int A[][3], int m, int n, int num)
{
    int i, j;
    for (i = 0; i < m; i++)
    {
        for (j = 0; j < n; j++)
        {
            if (A[i][j] == num)
            {
                printf("Numero %d encontrado en posicion [%d][%d]\n", num, i, j);
                return;
            }
        }
    }
    printf("Numero %d no encontrado\n", num);
}
```

---

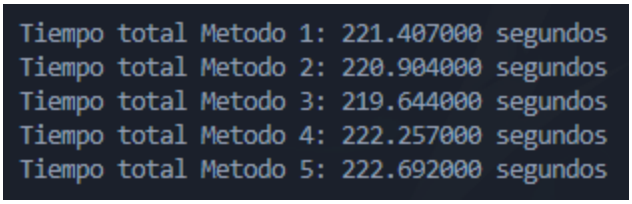
2. **¿Qué diferencias observaron al comparar los tiempos de ejecución de los métodos de búsqueda?**

Cuando son pocas iteraciones prácticamente ninguna, y aún a una escala mayor sigue siendo pequeño pero más notable:

100 iteraciones c/u:




1000 iteraciones c/u:




---

3. **¿Qué creen que podría estar influyendo en la diferencia de tiempo de impresión entre los métodos? ¿Cómo podrían optimizar la velocidad de impresión?**

Gracias a la manera en la que acceden a memoria cada método, y en algunos cuando se usan punteros y direcciones de memoria deben realizar más o menos procesos.

---

4. **¿Cuál es la ventaja de utilizar punteros y fórmulas de direccionamiento en comparación con la manipulación tradicional de arrays? ¿Pueden citar ejemplos específicos donde esto resultó beneficioso?**

Permite mas flexibilidad y eficiencia en la manipulación de datos. Con ellos se puede acceder a los datos de una matriz de diferentes maneras y nos permiten mayor modularidad y reutilización de código.

---

5. **¿Pueden identificar situaciones en las que el uso de punteros y fórmulas de direccionamiento podría no ser la mejor opción? ¿En qué casos la manipulación tradicional de arrays sería preferible?**

Cuando la simplicidad y legibilidad del código son más importantes que la eficiencia. La manipulación tradicional de arrays se puede preferir cuando se está trabajando con pequeñas cantidades de datos

---

6. **¿Cómo afecta la complejidad del código y la legibilidad al elegir entre métodos que utilizan punteros y fórmulas de direccionamiento y métodos**

**más convencionales? ¿Existen situaciones en las que la claridad del código es más importante que la eficiencia?**

En un script rápido para realizar una tarea simple, o si estás escribiendo código que será leído y mantenido por otros programadores, puede ser más importante que el código sea fácil de entender que extremadamente eficiente.

---

7. **¿Pueden proporcionar ejemplos de casos de uso en los que el rendimiento (tiempo de ejecución) es un factor crítico y, por lo tanto, los métodos con punteros y fórmulas de direccionamiento son esenciales?**

En aplicaciones que escalan a una gran cantidad de usuarios, en bases de datos que se hacen miles a millones de consultas diarias y en sistemas embebidos con poco poder de procesamiento.