Actividad en clase

SEBASTIAN IZQUIERDO SAAVEDRA

SISTEMAS OPERATIVOS

24/10/2025

Actividad en clase

Actividad No. 1 – Variables y punteros

Se realiza con una variable entera a la que se le asignó un valor inicial. Posteriormente se imprimió su dirección en memoria y se modificó su contenido utilizando un puntero. Con este ejercicio se evidenció cómo un puntero permite acceder al valor de una variable y cambiarlo indirectamente, entendiendo que el puntero guarda la dirección de la variable original.

Resultado:
Se observó el valor inicial y final de la variable, junto con la dirección de memoria donde fue almacenada, comprobando el funcionamiento del puntero.

Actividad No. 2 – Punteros y referencias

El propósito de esta parte fue comparar el uso de punteros y referencias. Para ello se declaró una variable entera, un puntero que apuntaba a ella y una referencia que hacía lo mismo. Se modificó el valor de la variable desde ambos métodos y se verificó que las tres compartían la misma dirección en memoria.

Resultado:
El valor de la variable cambió correctamente en ambas modificaciones, lo que permitió comprender que punteros y referencias acceden al mismo espacio en memoria aunque se utilicen de manera distinta.

Actividad No. 3 – Arrays y punteros

Se trabajó con un arreglo de números enteros. Se utilizó un puntero para recorrer los elementos y modificar sus valores directamente en memoria. Este ejercicio permitió reforzar el concepto de que el nombre del arreglo representa la dirección base del mismo, y que mediante aritmética de punteros se puede acceder a cada posición.

Resultado:
Cada elemento del arreglo fue modificado correctamente, comprobando que los punteros son útiles para recorrer estructuras de datos y acceder a sus posiciones sin necesidad de índices tradicionales.

Actividad No. 4 – Memoria dinámica

Aquí se implementó una matriz dinámica de enteros utilizando los operadores new y heap. Se asignó memoria en el heap, se llenó la matriz con datos, se imprimieron los resultados y finalmente se liberó la memoria utilizada.

Resultado:
La matriz se creó y se llenó con éxito, y al liberar la memoria se evitó cualquier fuga. Este ejercicio permitió comprender la diferencia entre el stack y el heap

Conclusiones

- El direccionamiento de memoria es un concepto esencial para entender cómo los programas almacenan y manipulan información.
- Los punteros permiten acceder directamente a los datos y son una herramienta clave en el manejo de memoria.
- Las referencias son una forma más sencilla y segura de acceder a los mismos valores sin usar operadores especiales.
- El uso adecuado de la memoria dinámica es fundamental para el rendimiento y estabilidad de los programas.
- Comprender el funcionamiento del stack, el heap y el código en C++ ayuda a desarrollar programas más eficientes y controlados.