

1. Cree una clase conteniendo un **double** y una función **print()** que imprime el **double**. En **main()**, cree punteros a miembros para el dato y la función miembro. Cree un objeto de su clase y un puntero a ese objeto y manipule estos elementos a través de punteros a miembro utilizando tanto el objeto como el puntero.
2. Cree una función con una variable estática que sea un puntero a enteros, la función debe recibir un argumento del tipo puntero entero con un valor por defecto 0. Cuando el llamador da un valor distinto de 0 para este argumento se apunta el puntero interno al puntero pasado (comienzo de un arreglo de enteros). Si se llama a la función sin argumentos, la función retorna el próximo valor en el arreglo hasta que vea un -1 en el arreglo (actúa como indicador de final del arreglo). Ejercite la función en un **main()**;
3. Cree una clase **Monitor** que mantiene la cuenta del número de veces que la función miembro **incident()** es llamada, Agregue una función miembro **print()** que visualiza el número de incidentes. Ahora cree una función global (no una función miembro) que contenga un objeto **Monitor** estático. Cada vez que se activa esta función debería llamar a **incident()** y luego a la **print()** de la instancia de **Monitor** para visualizar el contador de incidentes. Ejercite la función en un **main()**;
4. Cree una clase con un destructor que imprima un mensaje y luego llame a la función **exit()**. Cree un objeto global de esa clase y vea cómo se comporta.
5. Cree un archivo header con un namespace anónimo. Incluya ese header en 2 archivos cpp y muestre que el espacio anónimo es único en cada unidad de compilación.
6. Determine el valor de la constante dummy que su compilador pasa par a las versiones postfijo de los operadores **operator++** and **operator--**.
7. Escriba una clase **Number** que mantiene un **double**, agregue los operadores **+**, **-**, *****, **/**, y asignación. Elija el valor de retorno de esas funciones para que puedan ser encadenadas. Escriba un conversor de tipo automático **operator double()**.
8. Cree una clase que contenga un puntero y demuestre que si Ud. permite que el compilador sintetice el **operator=** el resultado de ese operador creará alias al mismo almacenamiento. Ahora solucione el problema definiendo su propio **operator=** y demuestre que esto corrige el aliasing. Chequee la auto-asignación y manipule el caso correctamente.
9. Cree dos clases, **Apple** y **Orange**. En **Apple**, cree un constructor que toma una **Orange** como argumento. Cree una función que tome un **Apple** y llame a esa función con una **Orange** para ver cómo trabaja.
10. Cree una clase simple que contenga un **int** y sobrecargue el **operator+** como función miembro. También implemente una función miembro **print** que reciba un **ostream&** como argumento e imprima sobre ese **ostream&**. Pruebe su clase para verificar su funcionamiento.
11. Agregue un operador binario **operator-** al ejercicio anterior como función miembro. Pruebe que puede utilizar sus objetos en expresiones complejas como **a + b - c**.
12. Agregue **operator++** y **operator--** al ejercicio anterior, tanto prefijo como postfijo y retornen el objeto incrementado o decrementado. Compruebe que la versión postfijo retorna el valor correcto.

13. Modifique los operadores implementados en el ejercicio anterior para que la versión prefijo retorne una referencia non-**const** y las versiones postfijo retornen un objeto constante. Muestre que trabajan correctamente. Explíquelo.
14. Modifique el ejercicio anterior para que los operadores **operator+** y **operator-** sean funciones no miembro. Pruebe que todavía siguen funcionando.