1. Escribir una clase **`Simple`** con un constructor y destructor que impriman algo indicando que han sido llamados. Instanciar objetos de esa clase en un **`main`**, con distintos scopes y verificar su correcta activación.
2. Modificar el ejercicio anterior para que la clase tenga un miembro del tipo **`int`**, el constructor reciba un entero con el cual inicializar el miembro y modificar los mensajes en el constructor y destructor para que incluyan al nuevo miembro. Incluir sentencias **`goto`** en la función **`main`** para verificar que los destructores son llamados igualmente.
3. ¿Cuán grande es una estructura? Escriba un programa que imprima el tamaño de una estructura. Defina una estructuras que tenga solo datos miembros y otra similar que además contenga funciones miembro, por último defina una estructura que no tenga datos y solo tenga funciones miembro. Imprima el tamaño de cada estructura. Justifique los resultados.
4. Cree una clase sin constructor y muestre que se pueden crear objetos utilizando el constructor por defecto. Ahora cree un constructor que reciba un argumento y reintente el mismo **`main`**.
5. Cree una clase **`Message`** con un constructor que tome un argumento del tipo **`string`** con un valor por defecto. Cree un miembro privado del tipo **`string`** e inicialice el mismo con el argumento recibido en el constructor. Cree dos funciones sobrecargadas llamadas **`print`**, una que no reciba argumentos e imprima el miembro **`string`** y otro que reciba un argumento del tipo **`string`** que se imprima juntamente con el miembro. Tiene sentido realizar esto o sería mejor utilizar un argumento por defecto como en el constructor?
6. Cree una nueva versión de la clase **`Stack`** que contenga un constructor por defecto y un segundo constructor que reciba un vector de punteros a objetos y el tamaño del vector. Este constructor debería recorrer el arreglo e ir **`push`**eando cada puntero sobre el **`stack`**. Probar la clase con un arreglo de punteros a **`string`**.
7. Defina tres constantes enteras, luego súmelas para generar un valor que determina el tamaño de un arreglo en una definición. Intente compilar el mismo código en C y vea que ocurre.
8. Cree una definición **`const`** en un archivo de encabezado (.h), incluya ese archive desde dos archivos fuente (.cpp), compile esos archivos y haga el link de ellos. No debería tener errores. Intente lo mismo con archivos fuentes en C.
9. Cree un arreglo constante de caracteres y trate de cambiar algún caracter. Probar con:

```
char *p = "ojo";
char q[] = "chau";
```

```
const char *r = "hola";
```

10. Cree una función que retorne el próximo valor en una sucesión de Fibonacci cada vez que es llamada. Agregue un argumento **bool** con valor por defecto **false**, que indique cuando resetear la secuencia para volver a empezar. Pruebe la función en un **main**.
11. Cree una clase con un destructor que imprima un mensaje y luego llame a **exit()**. Cree un objeto global de esta clase y vea que ocurre.
12. Cree una clase que contenga un entero, un constructor que inicialice el entero con el argumento del constructor, una función miembro que imponga un valor a ese entero con su argumento, y una función **print()** que lo imprima. Ponga a la clase en un archivo header e inclúyalo en dos archivos fuentes. En uno de ellos defina un objeto instancia de su clase y en el otro declare el mismo identificador como **extern** y pruébelo desde un **main.** Recuerde que tendrá que linkar ambos archivos objetos.
13. En el ejercicio anterior haga **static** a la instancia y verifique que el linker da un error.