

**¿De qué tipo es cada una de las siguientes variables?:**

- a) `int* a,b;`  
  - a puntero, b puntero
  - a puntero, b entero
  - a entero, b puntero
  - a entero, b entero
- b) `int *a,b;`  
  - a puntero, b puntero
  - a puntero, b entero
  - a entero, b puntero
  - a entero, b entero
- c) `int *a,*b;`  
  - a puntero, b puntero
  - a puntero, b entero
  - a entero, b puntero
  - a entero, b entero
- d) `int* a,*b;`  
  - a puntero, b puntero
  - a puntero, b puntero doble
  - a entero, b puntero
  - a entero, b puntero doble

**Considerando las siguientes declaraciones y sentencias:**

```
int array[]={1,2,3,4,5,6};
int *puntero;
puntero = array;
puntero++;
*puntero=*puntero+6;
puntero=puntero+3;
```

a) ¿Cuál es el valor de lo que hay en lo que apunta puntero?

- 1
- 2
- 3
- 4
- 5
- 6

b) ¿Cual es el valor de array[1]?

- 2
- 4
- 6
- 8

**Crea una clase que contenga cuatro métodos con 0, 1, 2 y 3 argumentos de tipo int respectivamente.**

Crea un `main()` que haga un objeto de su clase y llame a cada método.

Ahora modifique la clase para que tenga solo un método con todos los argumentos por defecto. ¿Esto cambia en algo el código de `main()`?

**Escriba un programa en el cual intente:**

- 1.- Crear una referencia que no esté inicializada cuando se crea.
- 2.- Cambiar una referencia para que se refiera a otro objeto después de que se haya inicializado.
- 3.- Crear una referencia nula.

**Describir en una frase qué representan estas variables.**

```
const int* u;  
int const* v;  
int d = 1;  
int* const w = &d;  
const int* const x = &d;  
int const* const x2 = &d;  
int main() {}
```

**¿Qué no se puede hacer y por qué?**

```
int d = 1;  
const int e = 2;  
int* u = &d;  
int* v = &e;  
int* w = (int*)&e;
```

**Hacer una función que recibe algo por valor (y que sea grande), y que tenga un constructor que nos informe de que se le ha llamado.**

Cambiar luego esa función para que sea una referencia constante y comprobar que no cambia la forma de llamada para un programador cliente.

**Crear un vector de caracteres constante, después intentar cambiar uno de los caracteres.**

¿Qué te dice el compilador?

**Definir un puntero constante a objeto constante.** Probar que solamente se puede leer el valor de lo que apunta el puntero, pero no se puede cambiar el puntero ni lo que apunta.

**Crear una función que tome un argumento por valor como constante, después intentar cambiar el argumento en el cuerpo de la función.**

**Crear una clase con un método constante y otro ordinario.** Crear un objeto constante y otro no constante de esa clase e intentar invocar ambos métodos desde ambos objetos.

**Crear una función que me devuelva la suma de los enteros que se le han ido pasando por argumento utilizando una variable estática.**

```
int suma_anterior(int i);
```

**Crear una clase con un destructor que imprima un mensaje y después llame a exit().** Crear un objeto global de esta clase y mira qué pasa.

**Escribe una clase que en el constructor copia se anuncia a sí mismo a través de un cout.** Ahora crea una función que pasa un objeto de su nueva clase por valor y otro más que crea un objeto local de su nueva clase y lo devuelve por valor. Llama a estas funciones para demostrar que el constructor copia es, en efecto, llamado cuando se pasan y retornan objetos por valor.

**Crea una clase con un constructor que parezca un constructor copia, pero que tenga un argumento de más con un valor por defecto.** Comprueba que aun así se utiliza como constructor copia.

**Crea una clase simple sin constructor copia, y una simple función que tome un objeto de esa clase por valor.** Ahora cambia la clase añadiéndole una declaración (solo declare, no defina) privada de un constructor copia. Explica lo que ocurre cuando compila la función.

**En dos archivos de cabecera, crea dos espacios de nombres, cada uno conteniendo una clase (con todas las definiciones inline) con idéntico nombre que el del otro espacio de nombres.** Crea un archivo cpp que incluya ambos archivos. Crea una función y, dentro de la función, utilice la directiva using para introducir ambos espacios de nombres. Pruebe a crear un objeto de la clase y vea que sucede. Haga las directivas using globales (fuera de la función) para ver si existe alguna diferencia. Repare el problema usando la resolución de rango, y cree objetos de ambas clases.