REFERENCIAS

CADA APARTADO PRINCIPAL CORRESPONDE CON EL NOMBRE DEL PROYECTO CON EJEMPLOS DE LO APUNTADO AQUÍ

1. INTRODUCCIÓN A LAS REFERENCIAS

- Las referencias son una forma de establecer distintos alias para nuestras variables y poder tener otros nombres para modificar esa misma variable.
- Supongamos que tenemos la siguiente variable:
  `int var = 33`, que vive en la dirección de memoria '0x12ab'.
  Si creamos un alias de nuestra variable '`var`', por ejemplo '`varAlias`' podremos usar este alias de la misma manera que si usáramos la variable original y esto puede ser útil en distintas ocasiones.

2. DECLARACIÓN Y USO DE REFERENCIAS

- Primero creas la variable original y luego nuestra referencia con la siguiente sintaxis:
  tipoDato& nombreReferencia = nombreVariableOriginal
  Ejemplo:
  ```
  int variable1 = 3;
  int& referenciaAVariable1 = variable1;
  ```
  Una vez creada nuestra referencia podremos usarla como si de nuestra variable se tratase.
- Las referencias deben de ser declaradas e inicializadas en la **misma sentencia**.
- La dirección de memoria de la referencia creada, es la misma que la de la variable original.
- Si modificas el valor de una referencia, el cambio se va a ver reflejado en la variable original. A la inversa ocurre lo mismo.

3. REFERENCIAS VS. PUNTEROS

- Sabemos que un puntero almacena la dirección de memoria de una variable y que podemos trabajar con el valor original de la variable, sin embargo con las referencias podemos hacer lo mismo, por lo que ¿cuál es la diferencia entre ellos? La siguiente tabla compara las referencias con respecto a los punteros para visualizar claramente lo que diferencia al uno del otro:

Referencias	Punteros
NO es necesario desreferenciar la referencia para leer o modificar el valor de la variable	**Es necesario** desreferenciar el puntero para leer o modificar el valor de la variable
No puede ser modificada para referenciar a otra variable	**Puede ser modificado** para apuntar a otra variable (si no es un puntero constante)
Debe declararse e inicializarse en la misma sentencia	**Puede ser declarado sin inicializarse** (contendrá una dirección basura)

- Podemos pensar en las referencias como si fueran punteros constantes, pero con una sintaxis mucho más *user-friendly* ya que no es necesario desreferenciar las referencias para labores de lectura o escritura.

4. REFERENCIAS Y PUNTEROS CONSTANTES

- Si estableces una referencia como constante, mediante la palabra reservada `const` delante del tipo de dato de nuestra referencia, no podrás modificar el valor de la variable original a la que nuestro alias referencia.
- Se pueden replicar el comportamiento de las referencias constantes en punteros de la misma forma, usando la palabra reservada `const`, pero esta vez tanto delante del tipo de dato como detrás, para indicar, tanto que la variable a la que apunta es constante, como también el puntero en sí. Si solo pones const delante del tipo de dato va a funcionar pero tendrá un significado ligeramente diferente:

- Puntero constante que apunta a un `int` constante, el puntero **NO** puede apuntar a otra dirección de memoria y **NO** puede modificar el valor de la variable a la que apunta:
  `const int* const pConstAVariableConst = &variableConst;`
- Puntero **NO** constante que apunta a un `int` constante, el puntero **SÍ** puede apuntar a otra dirección de memoria, pero NO puede modificar el valor de la variable a la que apunta:
  `const int* pAVariableConst = &variableConst;`

- La sintaxis de las líneas de arriba no existe para las referencias, el concepto de referencia constante que referencia a una variable constante no existe, la variable puede ser constante o no, pero no es algo que haya que indicar en la sintaxis de la referencia.
  `const int& const refEdad = edad; //ERROR → Esta sintaxis no existe`
  En este caso la variable '`edad`' puede ser constante o no serlo.