3 methods of parameter passing in C++
 - by value
 - by reference
 - by constant reference

**pass-by-value**
 - the actual parameter in the function call is copied into the formal parameter in the function
 - changes made to the formal parameter aren't reflected in the actual parameter
 - copying takes time; therefore, if the parameter is bigger than one of the unit data types, it's better to use one of the other parameter passing modes.

**pass-by-reference**
 - a pointer to the actual parameter is put in the formal parameter
 - any changes to the formal will be made to the actual
 - use this method if the actual parameter needs to change
 - C++ arrays are always passed by reference without using an &
 - other data types need an & before the formal parameter

**pass-by-constant-reference**
 - like pass-by-reference, but the parameter does not change
 - place **const** in front of the type in the formal parameter list
 - const can also be used with arrays that are formal parameters
 - if the function should only access but not change a parameter, using const enforces its not changing

& and const go in the formal parameter list
they are not in the function call
use the most restrictive method that works - low coupling