[Why copy constructor is passed by reference not by pointer](https://www.codeproject.com/Questions/695541/Why-copy-constructor-is-passed-by-reference-not-by)

References are variables that act as an alias to another variable.References are implicitly const so they must be given a value upon declaration.  
  
First, Const references are often used as function parameters because const references allow us to access but not change the value of an object, they can be used to give a function access to an object, but give assurance to caller that the object will not be changed at all by the function.  
  
Secondly, a reference acts like a const pointer that is implicitly dereferenced.  
Because references always “point” to valid objects, and can never be pointed to deallocated memory, references are safer to use than pointers. Thus the reference should generally be preferred. Pointers should generally only be used in situations where references are not sufficient. So, to prevent memory corruption issues References are preferred to pointers.  
  
Taking both these factors we can give a copy-constructor like :

Hide   Copy Code

*// Copy constructor*

MyClass(const MyClass &cSource)

{

m\_nVariable1 = cSource.m\_nVariable1;

}

Thus the source object being passed is never modified within the copy-constructor and is used solely for read-only purposes.