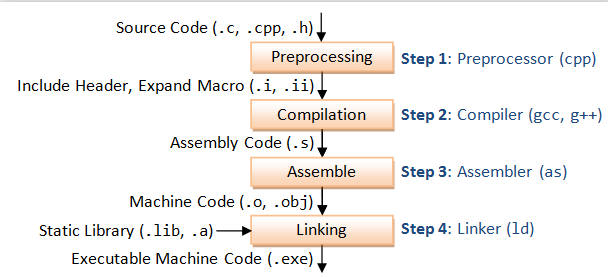
C++和java和C#的编译区别：

总所周知，正常编程所用的高级语言，计算机是无法是别的。需要先将高级语言翻译为机器语言，才能被机器运行。

在标准C\C++中，编译过程是这样的：



源代码首先经过预处理器，对头文件以及宏进行解析，然后经过编译器，接着，经过汇编生成机器指令，最后将所有文件链接起来。

这种编译方式的优点在于，最终直接生成了机器码，可以直接被计算机识别和运行，无需任何中间运行环境，但缺点也在于，由于不同平台能够识别的机器码不同，因此程序的跨平台能力较差。

而Java中，源代码不会被直接翻译成机器码，而是编译成一种中间代码。因此运行java程序需要一个额外的JRE（java运行环境），在JRE中存在JVM（java虚拟机），通过它会将中间代码进一步解释为机器码，这样就能实现更好的跨平台性。（java先编译再解释）

类似于Java，C#也是先将源代码编译成中间代码，（MSIL）然后再由.NET中的CLR将中间代码编译成机器码。（C#两次编译）

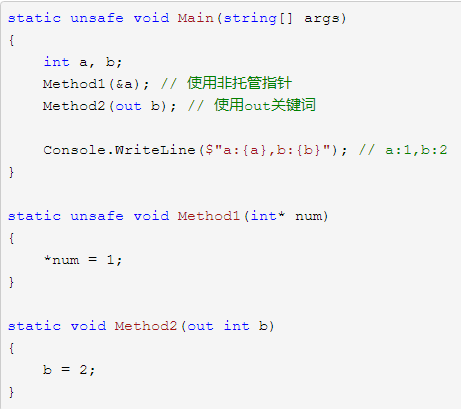
**托管和非托管是微软.NET framework中特有的概念**。

简单来说，托管的代码就是把有关内存管理（内存申请、内存释放、垃圾回收之类的）全部都是.net的CLR来管理，就是说使用托管的代码把底层一些操作全部封装起来了，不能直接进行内存的读取之类的和硬件相关的操作，优点是比较安全，不会出现内存泄漏之类的问题。

缺点也很明显，就是不能直接读取内存，性能上会有损失，使用起来也不够灵活。

非托管的刚好相反，可以直接进行硬件操作，性能比较高，但是对开发人员要求也比较高。

相反，非托管代码是非安全的，可以使用指针操作内存。对于一些对速度要求比较高的部分功能可以考虑使用非安全代码（unsafe关键字），使用指针等读写内存。



在C\C++中，资源都是需要手动释放的，比如，你new了一个指针，用过之后就需要delete掉，否则就会造成内存泄漏。

而在java中，不需要考虑资源释放的问题，java的垃圾回收机制（GC）会保证失效的资源被自动释放。

而C#的机制与java类似，运行于.net平台上的代码，分配的资源一般会自动由平台的垃圾回收器释放，这样的资源就是托管资源。