<https://www.cnblogs.com/AndyJee/p/4550391.html>

在看《程序员面试笔试宝典》时，发现了这样一个问题，书中只给出了++i的效率高一些，但并没有给出具体的解释和说明。

在网上找到下面的答案：

**1、从高级层面上解释**

++i 是i=i+1,表达式的值就是i本身

i++ 也是i=i+1,但表达式的值是加1前的副本，由于要先保存副本，因此效率低一些。

对于C++内置类型而言，大部分编译器会做优化，因此效率没什么区别。但在自定义类型上，就未必有优化，++i 效率会高一些。

**2、从底层汇编来看内置类型**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | int a,i=0; a=++i;汇编代码如下：    int a,i=0;  01221A4E mov dword ptr [i],0  a=++i;  01221A55 mov eax,dword ptr [i]  01221A58 add eax,1  01221A5B mov dword ptr [i],eax  01221A5E mov ecx,dword ptr [i]  01221A61 mov dword ptr [a],ecx    int a,i=0; a=i++;汇编代码如下：    int a,i=0;  009E1A4E mov dword ptr [i],0  a=i++;  009E1A55 mov eax,dword ptr [i]  009E1A58 mov dword ptr [a],eax  009E1A5B mov ecx,dword ptr [i]  009E1A5E add ecx,1  009E1A61 mov dword ptr [i],ecx |

从上述汇编代码可以看到，对于内置类型，它们的执行数目是一样的，效率没有差别。

**3、从重载运算符来看自定义类型**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | Operator Operator::operator++()  {  ++value; //内部成员变量  return \*this;  }    Operator Operator::operator++(int)  {  Operator temp;  temp.value=value;  value++;  return temp;  } |

从上面代码可以看出，后置++多了一个保存临时对象的操作，因此效率自然低一些。

**总结：**

对于C++内置类型，两者的效率差别不大；

对于自定义的类而言，++i 的效率更高一些。

参考文章：

http://www.cplusplus.me/1303.html