1. **头文件会放置哪些东西？**

函数声明、类定义、const常量定义、extern变量声明。

关于函数声明：为了让函数有个统一的声明形式，当函数原型（返回类型、函数名、形参）发生变化时只需要修改头文件中其唯一的声明式即可。（c++ primier 7.4）。

**Const常量定义：const int a =5;**

注意！：定义式一般是不允许放在头文件中的（避免重复定义），除const常量定义式、inline函数和类定义之外。

**2. c++预处理器：**

**3. 迭代器和指针相同的地方：**

指针和iterator都支持与整数进行+，-运算，而且其含义都是从当前位置向前或者向后移动n个位置。

指针和iterator都支持减法运算，指针-指针得到的是两个指针之间的距离，迭代器-迭代器得到的是两个迭代器之间的距离。

通过指针或者iterator都能够修改其指向的元素通过上面这几点看，两者真的很像，但是两者也有着下面的几个不同地方out操作符可以直接输出指针的值，但是对迭代器进行在操作的时候会报错。通过看报错信息和头文件知道，迭代器返回的是对象引用而不是对象的值，所以cout只能输出迭代器使用\*取值后的值而不能直接输出其自身。

指针能指向函数而迭代器不行，迭代器只能指向容器。这就说明了迭代器和指针其实是完全不一样的概念来的。指针是一种特殊的变量,它专门用来存放另一变量的地址，而迭代器只是参考了指针的特性进行设计的一种STL接口。

笔者曾在网上看到这样一种说法：迭代器是广义指针，而指针满足所有迭代器要求。迭代器是STL算法的接口，而指针是迭代器，因此STL算法可以使用指针来对基于指针的非STL容器进行操作。

笔者觉得上面说法也有几分道理，但是到底正不正确就留给看官自己判断了。但是有一点希望大家注意的是：千万不要把指针和迭代器搞混了。也许某些编译器使用指针来实现迭代器以至于有些人会误以为指针和迭代器是一个概念来的。

**4、dynamic\_cast<type T>(expression)强制类型转换**

是RTTI机制里面的一种方法，他提供了安全的将基类指针或引用转换为派生类指针或引用的功能。前提是基类必须带有一个或多个虚函数。

RTTI机制是虚函数机制的一种补充，两者都是实现通过基类指针或引用获得派生类行为的方法。