第九章：

一般应该把内联定义放在头文件里。当编译器看到这个定义时，它把函数类型（函数 名+返回值）和函数体放到符号表里。当使用函数时，

编译器检查以确保调用是正确的且 返回值被正确使用，然后将函数调用替换为函数体，因而消除了开销。内联代码的确占用 空间，但假如

函数较小，这实际上比为了一个普通函数调用而产生的代码（参数压栈和执 行CALL）占用的空间还少。

在头文件中，内联函数处于一种特殊状态，因为在头文件中声明该函数，所以必须包 含头文件和该函数的定义，这些定义在每个用到该函

数的文件中，但是不会出现产生多个定义错误的情况（不过，在任何使用内联函数地方该内联函数的定义都必须是相同的）。

能够隐藏类的底层实现是关键的，因为在以后有可能想修改这一实现。我们可能为了 效率这样做，或为了对问题有更好的理解，或因为有

些新类变得可用而想在实现里使用这些新类。任何危害实现隐蔽性的东西都会减少语言的灵活性。这样，内联函数就显得非常重要，因为它实

际上消除了预处理器宏和伴随的问题。通过用内联函数方式，成员函数可以和预处理器宏一样有效。

虚基类背后的实现机制：

假设类Bottom继承类Left和Right，而Left和Right均继承自Top，那么如果Top中有成员变量或虚函数，它们可能会被Bottom继承两次，引入二义性。

为了解决菱形继承问题，引入了虚基类的概念。通过将A定义为虚基类，只会有一个Top的实例被共享，在类Bottom中只有一个Top子对象。

每个类都有一个虚表，其中存储了该类的虚函数的指针。虚基类中的虚函数也会在虚表中有相应的位置。对于虚基类，编译器会创建一个虚继承表，用于跟踪虚基类在派生类中的偏移量。这个表确保不同派生类中共享的虚基类只有一个实例。派生类的构造函数和析构函数只会调用一次虚基类的析构函数。

因为虚基类引起共享子对象，共享发生之前它们就应该存在才有意义。所以子对象的初始化顺序按如下的规则递归地进行：

1、所有虚基类子对象，按照它们在类定义中出现的位置，从上到下、从左到右初始化。

2、非虚基类按通常顺序初始化。

3、所有的成员对象按声明的顺序初始化。

4、完整的对象的构造函数执行。