C 辅导:C 的效率浅析 PDF转换可能丢失图片或格式，建议阅读原文
https://www.100test.com/kao_ti2020/291/2021_2022_C___E8_BE_85_E5_AF_BC__c67_291935.htm 自从七十年代C语言诞生以来，一直以其灵活性、高效率和可移植性为软件开发人员所钟爱，成为系统软件开发的首选工具。而C 作为C语言的继承和发展，不仅保留了C语言的高度灵活、高效率和易于理解等诸多优点，还包含了几乎所有面向对象的特征，成为新一代软件系统构建的利器。 相对来说，C语言是一种简洁的语言，所涉及的概念和元素比较少，主要是：宏(macro)、指针(pointer)、结构(struct)、函数(function)和数组(array)，比较容易掌握和理解。而C 不仅包含了上面所提到的元素，还提供了私有成员(private members)、公有成员(public members)、函数重载(function overloading)、缺省参数(default parameters)、构造函数、析构函数、对象的引用(references)、操作符重载(operator overloading)、友元(friends)、模板(templates)、异常处理(exceptions)等诸多的要素，给程序员提供了更大的设计空间，同时也增加了软件设计的难度。 C语言之所以能被广泛的应用，其高效率是一个不可忽略的原因，C语言的效率能达到汇编语言的80%以上，对于一种高级语言来说，C语言的高效率就不言而喻了。那么，C 相对于C来说，其效率如何呢？实际上，C 的设计者stroustrup要求C 效率必须至少维持在与C相差5%以内，所以，经过精心设计和实现的C 同样有很高的效率，但并非所有C 程序具有当然的高效率，由于C的特殊性，一些不好的设计和实现习惯依然会对系统的效率造成较大的影响。同时，也由于有一部分程序员对C 的一些

底层实现机制不够了解，就不能从原理上理解如何提高软件系统的效率。 本文主要讨论两个方面的问题：第一，对比C的函数调用和C函数调用，解析C 的函数调用机制；第二，例举一些C 程序员不太注意的技术细节，解释如何提高C 的效率。为方便起见，本文的讨论以下面所描述的单一继承为例(多重继承有其特殊性，另作讨论)。 class X { public: virtual ~X(). file://析构函数 virtual void VirtualFunc(). file://虚函数 inline int InlineFunc() { return m_iMember}. file://内联函数 void NormalFunc(). file://普通成员函数 static void StaticFunc(). file://静态函数 private: int m_iMember. }. class XX: public X { public: XX(). virtual ~XX(). virtual void VirtualFunc(). private: String m_strName. int m_iMember2. }. C 的的函数分为四种：内联函数(inline member function)、静态成员函数(static member function)、虚函数(virtual member function)和普通成员函数。内联函数类似于C语言中的宏定义函数调用，C 编译器将内联函数的函数体扩展在函数调用的位置，使内联函数看起来象函数，却不需要承受函数调用的开销，对于一些函数体比较简单的内联函数来说，可以大大提高内联函数的调用效率。但内联函数并非没有代价，如果内联函数体比较大，内联函数的扩展将大大增加目标文件和可运行文件的大小；另外，inline关键字对编译器只是一种提示，并非一个强制指令，也就是说，编译器可能会忽略某些inline关键字，如果被忽略，内联函数将被当作普通的函数调用，编译器一般会忽略一些复杂的内联函数，如函数体中有复杂语句，包括循环语句、递归调用等。所以，内联函数的函数体定义要简单，否则在效率上会得不偿失。 静态函数的调用，如下面的几种方式

：X obj. X* ptr = amp.obj. obj.NormalFunc(). ptr->NormalFunc(). 将被被编译器转化为如下的C函数调用形式，如同这样。 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com