<<C语言深度解剖>>

图书基本信息

书名：<<C语言深度解剖>>

13位ISBN编号：9787512401440

10位ISBN编号：7512401442

出版时间：2010-7

出版时间：北京航空航天大学出版社

作者：陈正冲　编著

页数：165

字数：252000

前言

我面试过很多人，包括应届本科、硕士和工作多年的程序员，在问到C语言相关问题的时候，总是没几个人能完全答上我的问题。
甚至一些工作多年，简历上写着“最得意的语言是C语言”、“对C有很深的研究”、“精通C语言”的人也不完全能答对我的问题，更有甚者我问的问题一个都答不上。
我也给很多程序员和计算机系毕业的学生讲解过《高级C语言程序设计》。
每期开课前，我总会问学生：你感觉C语言学得怎么样？
难吗？
指针明白吗？
数组呢？
内存管理呢？
往往学生回答说：感觉还可以，C语言不难，指针很明白，数组很简单，内存管理也不难。
一般我会再问一个问题：通过这个班的学习，你想达到什么程度？
很多学生回答：精通C语言。
我告诉他们：我很无奈，也很无语，因为我完全在和一群业余者或者是C语言爱好者在对话。
你们浪费了大学学习计算机的时间，念了几年大学，连C语言的门都没摸着。
现在大多数学校计算机系都开了C、C++、Java、C#等语言，好像什么都学了，但是什么都不会，更可悲的是有些大学居然取消了C语言课程，认为其过时了。
我个人的观点是“十鸟在林，不如一鸟在手”，真正把C语言整明白了再学别的语言也很简单，如果C语言都没整明白，别的语言学得再好也是花架子，因为你并不了解底层是怎么回事。
当然我也从来不认为一个没学过汇编的人能真正掌握C语言的真谛。
我个人一直认为，普通人用C语言在3年之下，一般来说，还没掌握C语言；5年之下，一般来说还没熟悉C语言；10年之下，谈不上精通。
所以，我告诉我的学生：听完我的课，远达不到精通的目标，熟悉也达不到，掌握也达不到。

<<C语言深度解剖>>

内容概要

本书由作者结合自身多年嵌入式C语言开发经验和平时讲解C语言的心得体会整理而成，其中有很多作者独特的见解或看法。
由于并不是从头到尾讲解C语言的基础知识，所以本书并不适用于C语言零基础的读者，其内容要比一般的C语言图书深得多、细致得多，其中有很多问题是各大公司的面试或笔试题。
本书适合广大计算机系学生、初级程序员参考学习，也适合计算机系教师、中高级程序员参考使用。

作者简介

陈正冲，湖南沅江人，毕业于长春光学精密机械学院数学系。
具有丰富的嵌入式软件开发与管理经验，曾多次举办各种技术和管理方面的讲座和培训。
讲课深入、透彻、幽默，深受学员好评。
目前从事与CMMI相关的流程管理方面的工作。

<<C语言深度解剖>>

书籍目录

第1章 关键字 1.1 最宽宏大量的关键字——auto 1.2 最快的关键字——register 1.2.1 皇帝身边的小太监——寄存器 1.2.2 使用register修饰符的注意点 1.3 最名不符实的关键字——static 1.3.1 修饰变量 1.3.2 修饰函数 1.4 基本数据类型——short、int、long、char、float、double 1.4.1 数据类型与“模子” 1.4.2 变量的命名规则 1.5 最冤枉的关键字——sizeof 1.5.1 常年被人误认为函数 1.5.2 sizeof（int）*p表示什么意思 1.6 signed、unsigned关键字 1.7 if、else组合 1.7.1 bool变量与“零值”进行比较 1.7.2 float变量与“零值”进行比较 1.7.3 指针变量与“零值”进行比较 1.7.4 else到底与哪个if配对呢 1.7.5 if语句后面的分号 1.7.6 使用if语句的其他注意事项 1.8 switch、case组合 1.8.1 不要拿青龙偃月刀去削苹果 1.8.2 case关键字后面的值有什么要求吗 1.8.3 case语句的排列顺序 1.8.4 使用case语句的其他注意事项 1.9 do、while、for关键字 1.9.1 break与continue的区别 1.9.2 循环语句的注意点 1.10 goto关键字 1.11 void关键字 1.11.1 void a 1.11.2 void修饰函数返回值和参数 1.11.3 void指针 1.11.4 void不能代表一个真实的变量 1.12 return关键字 1.13 const关键字也许该被替换为readonly 1.13.1 const修饰的只读变量 1.13.2 节省空间，避免不必要的内存分配，同时提高效率 1.13.3 修饰一般变量 1.13.4 修饰数组 1.13.5 修饰指针 1.13.6 修饰函数的参数 1.13.7 修饰函数的返回值 1.14 最易变的关键字——volatile 1.15 最会带帽子的关键字——extern 1.16 struct关键字 1.16.1 空结构体多大 1.16.2 柔性数组 1.16.3 struct与class的区别 1.17 union关键字 1.17.1 大小端模式对union类型数据的影响 1.17.2 如何用程序确认当前系统的存储模式 1.18 enum关键字 1.18.1 枚举类型的使用方法 1.18.2 枚举与#define宏的区别 1.19 伟大的缝纫师——typedef关键字 1.19.1 关于马甲的笑话 1.19.2 历史的误会——也许应该是typerename 1.19.3 typedef与#define的区别 1.19.4 #define a int\[10\]与typedef int a\[10\]第2章 符号 2.1 注释符号 2.1.1 几个似非而是的注释问题 2.1.2 y = x/*p 2.1.3 怎样才能写出出色的注释 2.2 接续符和转义符 2.3 单引号、双引号 2.4 逻辑运算符 2.5 位运算符 2.5.1 左移和右移 2.5.2 0x01[[2+3的值为多少 2.6 花括号 2.7 ++、--操作符 2.7.1 ++i+++i+++i 2.7.2 贪心法 2.8 2 /（-2）的值是多少 2.9 运算符的优先级 2.9.1 运算符的优先级表 2.9.2 一些容易出错的优先级问题第3章 预处理 3.1 宏定义 3.1.1 数值宏常量 3.1.2 字符串宏常量 3.1.3 用define宏定义注释符号“？” 3.1.4 用define宏定义表达式 3.1.5 宏定义中的空格 3.1.6 #undef 3.2 条件编译 3.3 文件包含 3.4 #error预处理 3.5 #line预处理 3.6 #pragma预处理 3.6.1 #pragma message 3.6.2 #pragma code_seg 3.6.3 #pragma once 3.6.4 #pragma hdrstop 3.6.5 #pragma resource 3.6.6 #pragma warning 3.6.7 #pragma comment 3.6.8 #pragma pack 3.7“#”运算符 3.8“##”预算符第4章 指针和数组 4.1 指针 4.1.1 指针的内存布局 4.1.2“*”与防盗门的钥匙 4.1.3 int*p=NULL和*p=NULL有什么区别 4.1.4 如何将数值存储到指定的内存地址 4.1.5 编译器的bug 4.1.6 如何达到手中无剑、胸中也无剑的境界 4.2 数组 4.2.1 数组的内存布局 4.2.2 省政府和市政府的区别——&a\[0\]和&a的区别 4.2.3 数组名a作为左值和右值的区别 4.3 指针和数组之间的恩恩怨怨 4.3.1 以指针的形式访问和以下标的形式访问 4.3.2 a和&a的区别 4.3.3 指针和数组的定义与声明 4.3.4 指针和数组的对比 4.4 指针数组和数组指针 4.4.1 指针数组和数组指针的内存布局 4.4.2 int （*）\[10\] p2——也许应该这么定义数组指针 4.4.3 再论a和&a之间的区别 4.4.4 地址的强制转换 4.5 多维数组和多级指针 4.5.1 二维数组 4.5.2 二级指针 4.6 数组参数和指针参数 4.6.1 一维数组参数 4.6.2 一级指针参数 4.6.3 二维数组参数和二级指针参数 4.7 函数指针 4.7.1 函数指针的定义 4.7.2 函数指针的使用 4.7.3 （*（void（*） （））0） （）——这是什么 4.7.4 函数指针数组 4.7.5 函数指针数组指针第5章 内存管理 5.1 什么是野指针 5.2 栈、堆和静态区 5.3 常见的内存错误及对策 5.3.1 指针没有指向一块合法的内存 5.3.2 为指针分配的内存太小 5.3.3 内存分配成功，但并未初始化 5.3.4 内存越界 5.3.5 内存泄漏 5.3.6 内存已经被释放了，但是继续通过指针来使用第6章 函数 6.1 函数的由来与好处 6.2 编码风格 6.3 函数设计的一般原则和技巧 6.4 函数递归 6.4.1 一个简单但易出错的递归例子 6.4.2 不使用任何变量编写strlen函数第7章 文件结构 7.1 文件内容的一般规则 7.2 文件名命名的规则 7.3 文件目录的规则第8章 关于面试的秘密 8.1 外表形象 8.1.1 学生就是学生，穿着符合自己身份就行了 8.1.2 不要一身异味，熏晕考官对你没好处 8.1.3 女生不要带2个以上耳环，不要涂指甲 8.2 内在表现 8.2.1 谈吐要符合自己身份，切忌不懂装懂、满嘴胡咧咧 8.2.2 态度是一种习惯，习惯决定一切 8.2.3 要学会尊敬别人和懂礼貌 8.3 如何写一份让考官眼前一亮的简历 8.3.1

个人信息怎写　8.3.2 求职意向和个人的技能、获奖或荣誉情况怎么突出　8.3.3 成绩表是应届生必须要准备的附录1 C语言基础测试题附录2 C语言基础测试题答案后记参考文献

章节摘录

插图：其实在汇编语言阶段，函数这个概念还是比较模糊的。
汇编语言的代码往往就是从入口开始一条一条执行，直到遇到跳转指令（比如ARM指令B、BL、BX、BLX之类）然后才跳转到目的指令处执行。
这个时候所有的代码仅仅是按其将要执行的顺序排列而已。
后来人们发现这样写代码非常费劲，容易出错，也不方便。
于是想出一个办法，把一些功能相对来说能成为一个整体的代码放到一起打包，通过一些数据接口和外界通信。
这就是函数的由来。
那函数能给我们带来什么好处呢？
简单来说可以概括为以下几点：①降低复杂性：使用函数最首要的原因是为了降低程序的复杂性，可以使用函数来隐含信息，从而使你不必再考虑这些信息。
②避免重复代码段：如果在两个不同函数中的代码很相似，这往往意味着分解工作有误。
这时，应该把两个函数中重复的代码都取出来，把公共代码放入一个新的通用函数中，然后再让这两个函数调用新的通用函数。
通过使公共代码只出现一次，可以节约许多空间，因为只要在一个地方改动代码就可以了。
这时代码也更可靠了。
③限制改动带来的影响：在独立区域进行改动，由此带来的影响也只限于一个或最多几个区域中。
④隐含顺序：如果程序通常先从用户那里读取数据，然后再从一个文件中读取辅助数据，那么在设计系统时编写一个函数，隐含那个首先执行的信息。
⑤改进性能：把代码段放入函数也使得用更快的算法或执行更快的语言（如汇编）来改进这段代码的工作变得容易些。

后记

写书不容易，写一本好书更不容易，写一本满足所有读者的好书更是几乎没有可能。
本书的初稿挂在CSDN网站之后，3天内下载量冲到周排行榜第一名，2个月单链接下载量达4 000以上，至于各个网站转载后的下载量更是无从统计了。
目前，仅百度文库的下载量已实破3万次。
从网友的反馈来看，绝大多数还是觉得本书非常不错，但仍然还是有极个别网友觉得本书满足不了他们的要求。
比如有人提出，本书没有从汇编的角度来解剖C语言，是个遗憾。
其实，我个人并非没有考虑过深入到汇编层次，但最终没有这么做，原因有以下几点。
第一，C语言和汇编语言本来就是两种语言，既然本书的定位是讲解C语言，那就尽量在C语言的层次上解决问题。
况且，很多人没学过或是对汇编语言不太懂，如果一下子就深入到汇编语言，可能会加大理解本书的难度，得不偿失。
因为本书的一个显著特点就是深入浅出，将难以理解的问题通过各种方式来表达，从而降低学习的难度。
第二，从汇编语言的层次来解读C语言，这个事已经有人做了。
姚新颜先生花了好几年时间写的《C语言：标准与实现》，就是从汇编层面来解读C语言，已经给读者献上了一份厚礼。
我深感学识水平远不如姚先生，所以未敢班门弄斧。
第三，相对于很多读者所学的x86汇编，我个人更熟悉ARM汇编一些。
如果要从汇编的层面来写书，我可能没有太多时间学习x86汇编，而有可能以ARM汇编为基础，这样同样有可能增加读者的学习难度和降低读者学习C语言的兴趣。
第四，汇编语言目前的确用得很少了，哪怕是在嵌入式开发方面，绝大多数情况下用C语言也可以解决问题，偶尔会内嵌几句汇编代码，很少使用纯汇编写代码。
我对汇编的看法是，要懂它，但不要花过多的精力。

媒体关注与评论

《C语言深度解剖》从另一个层面来让你更深地了解c语言的精华所在，很好的一本书。
——网友，zuoshaobokzcj4这是一部经典的c语言讲解教案，你可以在其中学习到你从没思考过的关键知识！
——网友，C1989如果是学习过c语言的人，那么看几页就会觉得很不一般，大力推荐学过或正在学习C语言的人阅读此书，绝对受益匪浅。
——网友，caliow偶然在网上看到这本书，读毕，获益颇多。
看过不少c语言的书籍，貌似很少有哪本书提过哪些东西应该写在.h头文件里，哪些东西应该写在.cpp里。
——网友，惊羽九天强烈推荐，两位老师20年编程工作的经验总结，一位资深嵌入式工程师的呕心沥血之作！
堪称同类资料中之经典中的经典。
——网友，HXW718059156这是一篇c语言高手的箴言，将c语言的精髓展示出来。
——网友，lingzhimeng很好，对有些问题的见解很有深度，提出了很多人没有想到的问题，其中对关键字的解释很有创意，诸如static等，用了很多很好的例子，值得一看。
——网友，FengHui

<<C语言深度解剖>>

编辑推荐

《C语言深度解剖:解开程序员面试笔试的秘密》：博客藏经阁丛书

<<C语言深度解剖>>

版权说明