

<<征服C指针>>

图书基本信息

书名：<<征服C指针>>

13位ISBN编号：9787115301215

10位ISBN编号：7115301212

出版时间：2013-2

出版时间：人民邮电出版社

作者：前桥和弥

译者：吴雅明

版权说明：本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问：<http://www.tushu007.com>

## <<征服C指针>>

### 内容概要

《图灵程序设计丛书:征服C指针》被称为日本最有营养的C参考书。作者是日本著名的“毒舌程序员”，其言辞犀利，观点鲜明，往往能让读者迅速领悟要领。书中结合了作者多年的编程经验和感悟，从C语言指针的概念讲起，通过实验一步一步地为我们解释了指针和数组、内存、数据结构的关系，展现了指针的常见用法，揭示了各种使用技巧。另外，还通过独特的方式教会我们怎样解读C语言那些让人“纠结”的声明语法，如何绕过C指针的陷阱。

## <<征服C指针>>

### 作者简介

前桥和弥 (Maebashi Kazuya) 1969年出生, 著有《彻底掌握C语言》、《Java之谜和陷阱》、《自己设计编程语言》等。

其一针见血的"毒舌"文风和对编程语言深刻的见地受到广大读者的欢迎。

作者主页: <http://kmaebashi.com/>。

译者简介: 吴雅明 13年编程经验。

其中7年专注于研发基于JavaEE和.NET的框架和代码生成工具。

目前主要关注的方向有: Hadoop、HTML5、智能手机应用开发等。

## &lt;&lt;征服C指针&gt;&gt;

## 书籍目录

第0章本书的目标与结构——引言 0.1本书的目标 0.2目标读者和内容结构 第1章从基础开始——预备知识和复习 1.1C是什么样的语言 1.1.1比喻 1.1.2C的发展历程 1.1.3不完备和不统一的语法 1.1.4ANSI C 1.1.5C的宝典——K&R 1.1.6C的理念 1.1.7C的主体 1.1.8C是只能使用标量的语言 1.2关于指针 1.2.1恶名昭著的指针究竟是什么 1.2.2和指针的第一次亲密接触 1.2.3指针和地址之间的微妙关系 1.2.4指针运算 1.2.5什么是空指针 1.2.6实践——swap函数 1.3关于数组 1.3.1运用数组 1.3.2数组和指针的微妙关系 1.3.3下标运算符[]和数组是没有关系的 1.3.4为什么存在奇怪的指针运算 1.3.5不要滥用指针运算 1.3.6试图将数组作为函数的参数进行传递 1.3.7声明函数形参的方法 第2章做个实验见分晓——C是怎么使用内存的 2.1虚拟地址 2.2C的内存的使用方法 2.2.1C的变量的种类 2.2.2输出地址 2.3函数和字符串常量 2.3.1只读内存区域 2.3.2指向函数的指针 2.4静态变量 2.4.1什么是静态变量 2.4.2分割编译和连接 2.5自动变量（栈） 2.5.1内存区域的“重复使用” 2.5.2函数调用究竟发生了什么 2.5.3可变长参数 2.5.4递归调用 2.6利用malloc（）来进行动态内存分配（堆） 2.6.1malloc（）的基础 2.6.2malloc（）是“系统调用”吗 2.6.3malloc（）中发生了什么 2.6.4free（）之后，对应的内存区域会怎样 2.6.5碎片化 2.6.6malloc（）以外的动态内存分配函数 2.7内存布局对齐 2.8字节排序 2.9关于开发语言的标准和实现——对不起，前面的内容都是忽悠的 第3章揭秘C的语法——它到底是怎么回事 3.1解读C的声明 3.1.1用英语来阅读 3.1.2解读C的声明 3.1.3类型名 3.2C的数据类型的模型 3.2.1基本类型和派生类型 3.2.2指针类型派生 3.2.3数组类型派生 3.2.4什么是指向数组的指针 3.2.5C语言中不存在多维数组！ 3.2.6函数类型派生 3.2.7计算类型的大小 3.2.8基本类型 3.2.9结构体和共用体 3.2.10不完全类型 3.3表达式 3.3.1表达式和数据类型 3.3.2“左值”是什么——变量的两张面孔 3.3.3将数组解读成指针 3.3.4数组和指针相关的运算符 3.3.5多维数组 3.4解读C的声明（续） 3.4.1const修饰符 3.4.2如何使用const？可以使用到什么程度？ 3.4.3typedef 3.5其他 3.5.1函数的形参的声明 3.5.2关于空的下标运算符[] 3.5.3字符串常量 3.5.4关于指向函数的指针引起的混乱 3.5.5强制类型转换 3.5.6练习——挑战那些复杂的声明 3.6应该记住：数组和指针是不同的事物 3.6.1为什么会引起混乱 3.6.2表达式之中 3.6.3声明 第4章数组和指针的常用方法 4.1基本的使用方法 4.1.1以函数返回值之外的方式来返回值 4.1.2将数组作为函数的参数传递 4.1.3可变长数组 4.2组合使用 4.2.1可变长数组的数组 4.2.2可变长数组的可变长数组 4.2.3命令行参数 4.2.4通过参数返回指针 4.2.5将多维数组作为函数的参数传递 4.2.6数组的可变长数组 4.2.7纠结于“可变”之前，不妨考虑使用结构体 4.3违反标准的技巧 4.3.1可变长结构体 4.3.2从1开始的数组 第5章数据结构——真正的指针的使用方法 5.1案例学习1：计算单词的出现频率 5.1.1案例的需求 5.1.2设计 5.1.3数组版 5.1.4链表版 5.1.5追加检索功能 5.1.6其他的数据结构 5.2案例学习2：绘图工具的数据结构 5.2.1案例的需求 5.2.2实现各种图形的数据模型 5.2.3Shape型 5.2.4讨论——还有别的方法吗 5.2.5图形的组合 5.2.6继承和多态之道 5.2.7对指针的恐惧 5.2.8说到底，指针究竟是什么 第6章其他——拾遗 6.1陷阱 6.1.1关于strncpy 6.1.2如果在早期的C中使用float类型的参数 6.1.3printf（）和scanf 6.1.4原型声明的光和影 6.2惯用句法 6.2.1结构体声明 6.2.2自引用型结构体 6.2.3结构体的相互引用 6.2.4结构体的嵌套 6.2.5共用体 6.2.6数组的初始化 6.2.7char数组的初始化 6.2.8指向char的指针的数组的初始化 6.2.9结构体的初始化 6.2.10共用体的初始化 6.2.11全局变量的声明

## &lt;&lt;征服C指针&gt;&gt;

## 章节摘录

版权页：插图：空指针确保它和任何非空指针进行比较都不会相等，因此经常作为函数发生异常时的返回值使用。

另外，对于第5章的链表来说，也经常在数据的末尾放上一个空指针来提示：“请注意，后面已经没有元素了哦。

”在如今的操作系统下，应用程序一旦试图通过空指针引用对象，就会马上招致一个异常并且当前应用程序会被操作系统强制终止。

因此，如果每次都使用NULL来初始化指针变量，在错误地使用了无效（未初始化）的指针时，我们就可以马上发现潜在的bug。

通常，我们可以根据指针指向的数据类型来明确地区别指针的类型。

如果将“指向int的指针”赋给“指向double的指针”，如今的编译器会报出前面提到的警告。

但是，只有NULL，无论对方指向什么类型的变量，都可以被赋值和比较。

偶尔会见到先将空指针强制转型，然后进行赋值、比较操作的程序，这不但是徒劳的，甚至还会让程序变得难以阅读。

## <<征服C指针>>

### 编辑推荐

书中结合开发人员多年的编程经验和感悟，介绍了指针在数组中的应用、在函数中的应用、指向指针的指针、数据结构中指针的应用，指向文件类型的指针、指针在c语言算法中的应用，覆盖了所有重要的C编程话题，并给出了很多编程技巧和提示。

<<征服C指针>>

版权说明

本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问:<http://www.tushu007.com>