

<<C和指针>>

图书基本信息

书名：<<C和指针>>

13位ISBN编号：9787115308498

10位ISBN编号：7115308497

出版时间：2013-2

出版时间：人民邮电出版社

作者：里克

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>

<<C和指针>>

内容概要

《C和指针(英文版)》提供与C语言编程相关的全面资源和深入讨论。

本书通过对指针的基础知识和高级特性的探讨，帮助程序员把指针的强大功能融入到自己的程序中去

。全书共18章，覆盖了数据、语句、操作符和表达式、指针、函数、数组、字符串、结构和联合等几乎所有重要的C编程话题。

书中给出了很多编程技巧和提示，每章后面有针对性很强的练习，附录部分则给出了部分练习的解答

。

<<C和指针>>

作者简介

Kenneth Reek是罗彻斯特理工大学计算机科学教授。他是一位经验丰富的C程序员，曾为多家公司担任过技术顾问。他讲授的课程有操作系统、数据通信、计算机网络、形式语言、算法分析和交换系统等。本书正是基于他9年的编程及教学的经验积累。

<<C和指针>>

书籍目录

第1章快速上手 1.1简介 1.1.1空白和注释 1.1.2预处理指令 1.1.3main函数 1.1.4read_column_numbers函数 1.1.5rearrange函数 1.2补充说明 1.3编译 1.4总结 1.5警告的总结 1.6编程提示的总结 1.7问题 1.8编程练习

第2章基本概念 2.1环境 2.1.1翻译 2.1.2执行 2.2词法规则 2.2.1字符 2.2.2注释 2.2.3自由形式的源代码 2.2.4标识符 2.2.5程序的形式 2.3程序风格 2.4总结 2.5警告的总结 2.6编程提示的总结 2.7问题 2.8编程练习 第3章数据 3.1基本数据类型 3.1.1整型家族 3.1.2浮点类型 3.1.3指针 3.2基本声明 3.2.1初始化 3.2.2声明简单数组 3.2.3声明指针 3.2.4隐式声明 3.3typedef 3.4常量 3.5作用域 3.5.1代码块作用域 3.5.2文件作用域 3.5.3原型作用域 3.5.4函数作用域 3.6链接属性 3.7存储类型 3.8static关键字 3.9作用域、存储类型示例 3.10总结 3.11警告的总结 3.12编程提示的总结 3.13问题 第4章语句 4.1空语句 4.2表达式语句 4.3代码块 4.4if语句 4.5while语句 4.5.1break和continue语句 4.5.2while语句的执行过程 4.6for语句 4.7do语句 4.8switch语句 4.8.1switch中的break语句 4.8.2default子句 4.8.3switch语句的执行过程 4.9goto语句 4.10总结 4.11警告的总结 4.12编程提示的总结 4.13问题 4.14编程练习 第5章操作符和表达式 5.1操作符 5.1.1算术操作符 5.1.2移位操作符 5.1.3位操作符 5.1.4赋值 5.1.5单目操作符 5.1.6关系操作符 5.1.7逻辑操作符 5.1.8条件操作符 5.1.9逗号操作符 5.1.10下标引用、函数调用和结构成员 5.2布尔值 5.3左值和右值 5.4表达式求值 5.4.1隐式类型转换 5.4.2算术转换 5.4.3操作符的属性 5.4.4优先级和求值的顺序 5.5总结 5.6警告的总结 5.7编程提示的总结 5.8问题 5.9编程练习 第6章指针 6.1内存和地址 6.2值和类型 6.3指针变量的内容 6.4间接访问操作符 6.5未初始化和非法的指针 6.6NULL指针 6.7指针、间接访问和左值 6.8指针、间接访问和变量 6.9指针常量 6.10指针的指针 6.11指针表达式 6.12实例 6.13指针运算 6.13.1算术运算 6.13.2关系运算 6.14总结 6.15警告的总结 6.16编程提示的总结 6.17问题 6.18编程练习 第7章函数 7.1函数定义 7.2函数声明 7.2.1原型 7.2.2函数的缺省认定 7.3函数的参数 7.4ADT和黑盒 7.5递归 7.5.1追踪递归函数 7.5.2递归与迭代 7.6可变参数列表 7.6.1stdarg宏 7.6.2可变参数的限制 7.7总结 7.8警告的总结 7.9编程提示的总结 7.10问题 7.11编程练习 第8章数组 8.1一维数组 8.1.1数组名 8.1.2下标引用 8.1.3指针与下标 8.1.4指针的效率 8.1.5数组和指针 8.1.6作为函数参数的数组名 8.1.7声明数组参数 8.1.8初始化 8.1.9不完整的初始化 8.1.10自动计算数组长度 8.1.11字符数组的初始化 8.2多维数组 8.2.1存储顺序 8.2.2数组名 8.2.3下标 8.2.4指向数组的指针 8.2.5作为函数参数的多维数组 8.2.6初始化 8.2.7数组长度自动计算 8.3指针数组 8.4总结 8.5警告的总结 8.6编程提示的总结 8.7问题 8.8编程练习 第9章字符串、字符和字节 9.1字符串基础 9.2字符串长度 9.3不受限制的字符串函数 9.3.1复制字符串 9.3.2连接字符串 9.3.3函数的返回值 9.3.4字符串比较 9.4长度受限的字符串函数 9.5字符串查找基础 9.5.1查找一个字符 9.5.2查找任何几个字符 9.5.3查找一个子串 9.6高级字符串查找 9.6.1查找一个字符串前缀 9.6.2查找标记 9.7错误信息 9.8字符操作 9.8.1字符分类 9.8.2字符转换 9.9内存操作 9.10总结 9.11警告的总结 9.12编程提示的总结 9.13问题 9.14编程练习 第10章结构和联合 10.1结构基础知识 10.1.1结构声明 10.1.2结构成员 10.1.3结构成员的直接访问 10.1.4结构成员的间接访问 10.1.5结构的自引用 10.1.6不完整的声明 10.1.7结构的初始化 10.2结构、指针和成员 10.2.1访问指针 10.2.2访问结构 10.2.3访问结构成员 10.2.4访问嵌套的结构 10.2.5访问指针成员 10.3结构的存储分配 10.4作为函数参数的结构 10.5位段 10.6联合 10.6.1变体记录 10.6.2联合的初始化 10.7总结 10.8警告的总结 10.9编程提示的总结 10.10问题 10.11编程练习 第11章动态内存分配 11.1为什么使用动态内存分配 11.2malloc和free 11.3calloc和realloc 11.4使用动态分配的内存 11.5常见的动态内存错误 11.6内存分配实例 11.7总结 11.8警告的总结 11.9编程提示的总结 11.10问题 11.11编程练习 第12章使用结构和指针 12.1链表 12.2单链表 12.2.1在单链表中插入 12.2.2其他链表操作 12.3双链表 12.3.1在双链表中插入 12.3.2其他链表操作 12.4总结 12.5警告的总结 12.6编程提示的总结 12.7问题 12.8编程练习 第13章高级指针话题 13.1进一步探讨指向指针的指针 13.2高级声明 13.3函数指针 13.3.1回调函数 13.3.2转移表 13.4命令行参数 13.4.1传递命令行参数 13.4.2处理命令行参数 13.5字符串常量 13.6总结 13.7警告的总结 13.8编程提示的总结 13.9问题 13.10编程练习 第14章预处理器 14.1预定义符号 14.2#define 14.2.1宏 14.2.2#define替换 14.2.3宏与函数 14.2.4带副作用的宏参数 14.2.5命名约定 14.2.6#undef 14.2.7命令行定义 14.3条件编译 14.3.1是否被定义 14.3.2嵌套指令 14.4文件包含 14.4.1函数库文件包含 14.4.2本地文件包含 14.4.3嵌套文件包含 14.5其他指令 14.6总结 14.7警告的总结 14.8编程提示的总结 14.9问题 14.10编程练习 第15章输入 / 输出函数 15.1错误报告 15.2终止执行 15.3标准I/O函数库 15.4ANSI I/O概念 15.4.1流 15.4.2文件 15.4.3标准I/O常量 15.5流I/O总览 15.6打

<<C和指针>>

开流 15.7关闭流 15.8字符I/O 15.8.1字符I/O宏 15.8.2撤销字符I/O 15.9未格式化的行I/O 15.10格式化的行I/O 15.10.1scanf家族 15.10.2scanf格式代码 15.10.3printf家族 15.10.4printf格式代码 15.11二进制I/O 15.12刷新和定位函数 15.13改变缓冲方式 15.14流错误函数 15.15临时文件 15.16文件操纵函数 15.17总结 15.18警告的总结 15.19编程提示的总结 15.20问题 15.21编程练习 第16章标准函数库 16.1整型函数 16.1.1算术 16.1.2随机数 16.1.3字符串转换 16.2浮点型函数 16.2.1三角函数 16.2.2双曲函数 16.2.3对数和指数函数 16.2.4浮点表示形式 16.2.5幂 16.2.6底数、顶数、绝对值和余数 16.2.7字符串转换 16.3日期和时间函数 16.3.1处理器时间 16.3.2当天时间 16.4非本地跳转 16.4.1实例 16.4.2何时使用非本地跳转 16.5信号 16.5.1信号名 16.5.2处理信号 16.5.3信号处理函数 16.6打印可变参数列表 16.7执行环境 16.7.1终止执行 16.7.2断言 16.7.3环境 16.7.4执行系统命令 16.7.5排序和查找 16.8locale 16.8.1数值和货币格式 16.8.2字符串和locale 16.8.3改变locale的效果 16.9总结 16.10警告的总结 16.11编程提示的总结 16.12问题 16.13编程练习 第17章经典抽象数据类型 17.1内存分配 17.2堆栈 17.2.1堆栈接口 17.2.2实现堆栈 17.3队列 17.3.1队列接口 17.3.2实现队列 17.4树 17.4.1在二叉搜索树中插入 17.4.2从二叉搜索树删除节点 17.4.3在二叉搜索树中查找 17.4.4树的遍历 17.4.5二叉搜索树接口 17.4.6实现二叉搜索树 17.5实现的改进 17.5.1拥有超过一个的堆栈 17.5.2拥有超过一种的类型 17.5.3名字冲突 17.5.4标准函数库的ADT 17.6总结 17.7警告的总结 17.8编程提示的总结 17.9问题 17.10编程练习 第18章运行时环境 18.1判断运行时环境 18.1.1测试程序 18.1.2静态变量和初始化 18.1.3堆栈帧 18.1.4寄存器变量 18.1.5外部标识符的长度 18.1.6判断堆栈帧布局 18.1.7表达式的副作用 18.2C和汇编语言的接口 18.3运行时效率 18.4总结 18.5警告的总结 18.6编程提示的总结 18.7问题 18.8编程练习 附录部分问题答案

章节摘录

版权页：插图：3.7 Storage Class The storage class of a variable refers to the type of memory in which the variable's value is stored. The storage class of a variable determines when it is created and destroyed and how long it will retain its value. There are three possible places to store variables: in ordinary memory, on the runtime stack, and in hardware registers. Variables stored in these three places will have different characteristics. The default storage class for a variable depends on where it is declared. Variables declared outside of any blocks are always stored in static memory, that is, in memory that is not part of the stack. There is no way to specify any other storage class for these variables. Static variables are created before the program begins to run and exist throughout its entire execution. They retain whatever value they were assigned until a different value is assigned or until the program completes. The default storage class for variables declared within a block is automatic, that is, on the stack. There is a keyword `auto`, but it is rarely used because it doesn't change the default. Automatic variables are created just before the program execution enters the block in which they were declared, and they are discarded just as execution leaves that block. If the block is executed several times, as in the case of a function that is called repeatedly, new copies of the automatic variables are created each time. These new variables may or may not occupy the same memory locations on the stack as the previous instances of the same variables, and even if they do there is no guarantee that the memory was not used for some other purpose in the meantime. We therefore say that automatic variables disappear at the end of a block; they generally will not have their previous values the next time the block is entered. On variables declared within a block, the keyword `static` changes the storage class from automatic to static. Variables with static storage class exist for the entire duration of the program, rather than just the duration of the block in which it is declared. Note that changing the storage class of a variable does not change its scope; it is still accessible by name only from within the block. The formal parameters to a function cannot be declared static, because arguments are always passed on the stack to support recursion.

<<C和指针>>

编辑推荐

《C和指针(英文版)》提供与C语言编程相关的全面资源和深入讨论，通过对指针的基础知识和高级特性的探讨，帮助程序员把指针的强大功能融入到自己的程序中去。

《C和指针(英文版)》适合C语言初学者和初级C程序员阅读，也可作为计算机专业学生学习C语言的参考。

<<C和指针>>

名人推荐

我竭尽全力地推荐这本我所见过的最好的C编程入门图书。

作者深知读者所需，并为他们打下良好基础。

如果你已经开始学习C语言但始终不得要领，不妨试一试《C和指针(英文版)》。

——Francis Glassborow，ACCU主席

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>