科海培训中心系列教材

C语言

高级实用教程

尹彦芝 编著

清华大学出版社

北京科海培训中心系列教材

C语言高级实用教程

尹彦芝 编著

清华大学出版社

内 容 简 介

本书从比较深入的角度详细地阐述了一般C语言的参考书中很少提及，而实际应用中又很重要的C语言编程的一些难题，如中断处理程序、插入程序、pop__up图形窗口、文本窗口及其编辑、菜单及其编辑、内存动态分配、程序管理、基指针的使用等等。在每一个问题的阐述中都给出了一个以上的实例，这些实例完整而详尽，又都是开发于目前最流行的Turbo C、MS C、Turbo C Tools等编译工具上，极方便于读者的调试与学习。

本书适宜于计算机专业及相关专业的本科高年级学生和研究生，以及C语言程序开发人员，他们已掌握了C语言的基本知识，进而想在PC机的DOS操作系统下开发更高级的软件，或者想更为深入地了解C语言的一些细节和技巧。

(京) 新登字 158 号

JS518/13

C语言高级实用教程

尹彦芝 主编

☆

清华大学出版社出版

北京 清华园

北京市朝阳区经伟印刷厂印刷

新华书店总店科技发行所发行

☆

开本:787×1092 1/16 印张: 29.75 字数: 757 千字

1991 年 10 月第 1 版 1992 年 10 月第 1 次印刷

印数:00001～10000 册

ISBN 7-302-01100-1/TP · 412

定价: 19.50 元

前　言

C 语言是当前最流行的程序设计语言。本书是针对这样的读者而编写的：他们已熟悉和掌握了 C 语言的基本编程技术，现在想在 PC 机 DOS 操作系统下用 C 语言开发更完美的软件，或者想更为深入地了解 C 语言的一些细节和使用技巧。

为此，我们编写了这本《C 语言高级实用教程》。

本书既不是仅仅简单地罗列 C 语言的语法和函数，也不是仅仅提供一些可以整块移植的实用程序，而是将重点放在对一些一般的 C 语言的教材和参考书中极少提及、而实际应用中又极为重要的 C 语言编程的技术难点上（如中断处理程序，插入程序，Pop-up 图形窗口，文本窗口及其编辑，菜单及其编辑，内存动态分配，程序管理，基指针的使用等等）。为了加深读者的理解并给读者编程提供更多的方便，本书亦提供了许多演示程序和 9 个工具包及工具包的应用实例。这些工具包所提供的是一些针对某一类型的任务很有用的工具函数，这可作为对 C 语言函数的重要补充。因此称本书为“实用教程”。

同时，本书亦是一本“高级实用教程”。浏览一下本书的目录便可知道，本书不是按一般 C 语言教科书的“数据类型”、“表达式”、“控制语句”（假设读者已具备了这些基础知识）这样的方式来编写的，而是按照一般编程实践中所要处理的实际问题来划分章节。全书共 15 章，除第 1 章是阐述一些共同性的难点外，其余各章都是一章一个专题，依次是：编译模式和内存的组织，鼠标输入，文本屏幕输出和文本窗口，键盘输入、菜单及编辑，文件处理基本技术，字符串的处理，动态通用串的处理，文件处理高级技术，内存和程序管理，Microsoft C 6.0 中的基指针技术，与 DOS 和 BIOS 的接口，中断服务程序，图形处理，混合模式与混合语言编程。

相对于其他语言来说，C 语言有一个很重要的特点，这就是标准的 C 语言学起来并不困难，初看上去甚至很容易，但是，C 语言程序的真正实现都是基于其丰富的函数的，而这些函数随 C 编译系统的不同而有所区别，这使得要想真正掌握 C 语言的编程比起仅仅掌握 C 语言的语法本身来说要困难得多。Turbo C 2.0 大约有 440 多个函数，Microsoft C 6.0 提供的函数更多，更不用说最新推出的面向 Windows 3.0 编程的 Borland C++ 3.0 和 Microsoft C／C++ 7.0 了。掌握与查询这些函数的最好办法莫过于正确地分类。Turbo C 2.0 将其库函数分为 16 类。本书也采取分类的办法，但分类的原则与 Turbo C 的手册略有不同。当然，由于函数太多，本书也只能涉及其中的一部分。对于较易理解的函数，如数学函数等，本书根本未予提及，因为只要查阅手册便清楚了。而对于一些较重要而又较难掌握的函数，本书花的篇幅较多一些。书中除“动态通用串的处理”、“文件处理高级技术”这几个专题外，其余各章均从相应的 C 函数开始详细讲解这些函数的功能与限制，并以此为基础，提供一些演示程序或小实例或扩展出一些新的函数。

对国内为数众多的微型计算机用户来说，Turbo C 2.0 和 Microsoft C 6.0 是大家目前较为熟悉的编译系统。本书的讲解大多以 Turbo C 2.0 为线索，而在某些章节，如键盘输

入、窗口输出、菜单的处理、域编辑、中断服务程序和插入程序等，Turbo C Tools 系统中有了明显的改进或补充，本书将花较多的篇幅予以介绍。全书仅在第 11 章“MS_C6.0 的基指针技术”中的例子要求使用 Microsoft C 6.0，因只在这一编译系统中才提供了这种技术。美国 Blaise Computing 公司 1989 年推出的 Turbo C 工具库 Turbo C Tools 6.0 适用于 Turbo C 的 1.0、1.5 和 2.0 版本。这个工具库提供了丰富的库函数，如字符串转换、屏幕操作、窗口、菜单、鼠标器、键盘、文件、打印机、内存管理、中断服务、插入程序、编辑器、在线帮助等函数资源，充分利用好这些函数，可使用户很方便地设计出更为精美的用户界面，使得用户编写出的应用程序在外观上和操作上达到一个新的境界。

更深入地掌握 C 语言，更“优美”地编写 C 程序，这正是我们希望读者通过本书的学习所能达到的目标。但由于作者水平有限，书中所介绍的又都是近年来不断发展变化着的新技术，在全面与准确地把握各种 C 编译的系统与环境方面，难免有疏漏，欢迎读者来函指正。

如果读者对一些常用的实用程序和算法感兴趣，可参考由清华大学出版社出版的《C 语言常用算法和子程序》一书。

作　者

1992 年 6 月

目　录

第 1 章　几个重要问题

1.1　数据类型转换

1.1.1　各类整数之间的转换

C 语言中的数分 8 位、16 位和 32 位三种。属于 8 位数的有：带符号字符 char，无符号字符 unsigned char。属于 16 位数的有：带符号整数 int，无符号整数 unsigned int(或简写为 unsigned)，近指针。属于 32 位数的有：带符号长整数 long，无符号长整数 unsigned long，远指针。

IBM PC 是 16 位机，基本运算是 16 位的运算，所以，当 8 位数和 16 位数进行比较或其它运算时，都是首先把 8 位数转换成 16 位数。为了便于按 2 的补码法则进行运算，有符号 8 位数在转换为 16 位时是在左边添加 8 个符号位，无符号 8 位数则是在左边添加 8 个 0。当由 16 位转换成 8 位时，无论什么情况一律只是简单地裁取低 8 位，抛掉高 8 位。没有 char 或 unsigned char 常数。字符常数，像'C'，是转换为 int 以后存储的。当字符转换为其它 16 位数（如近指针）时，是首先把字符转换为 int，然后再进行转换。

16 位数与 32 位数之间的转换也遵守同样的规则。

注意，Turbo C 中的输入／输出函数对其参数中的 int 和 unsigned int 不加区分。例如，在 printf 函数中如果格式说明是%d，则对这两种类型的参数一律按 2 的补码(即按有符号数)进行解释，然后以十进制形式输出。如果格式说明是%u、%o、%x、%X，则对这两种类型的参数一律按二进制(即按无符号数)进行解释，然后以相应形式输出。在 scanf 函数中，仅当输入的字符串中含有负号时，才按 2 的补码对输入数进行解释。

还应注意，对于常数，如果不加 L，则 Turbo C 一般按 int 型处理。例如，语句 printf("%08lx", -1L)，则会输出 ffffffff。如果省略 l，则输出常数的低字，即 ffff。如果省略 L，则仍会去找 1 个双字，这个双字的低字就是 int 常数-1，高字内容是不确定的，输出效果将是在 4 个乱七八糟的字符之后再跟 ffff。

在 Turbo C 的头文件 value.h 中，相应于 3 个带符号数的最大值，定义了 3 个符号常数:

```
#define  MAXSHORT   0X7FFF
#define  MAXINT     0X7FFF
#define  MAXLONG    0X7FFFFFFFL
```

在 Turbo C Tools 中，包括 3 对宏，分别把 8 位拆成高 4 位和低 4 位，把 16 位拆成高 8 位和低 8 位，把 32 位拆成高 16 位和低 16 位。

```
uthinyb(char value)      utlonyb(char value)
uthibyte(int value)      utlobyte(int value)
uthiword(long value)     utloword(long value)
```

在 Turbo C Tools 中，也包括相反的 3 个宏，它们把两个 4 位组成一个 8 位，把两个 8 位组成一个 16 位，把两个 16 位组成一个 32 位。

```
utnybbyt (HiNyb, LoNyb)
utwdlong (HiWord, Loword)
utbyword (HiByte, LoByte)
```

1.1.2 实数与整数之间的转换

Turbo C 中提供了两种实数：float 和 double。float 由 32 位组成，由高到低依次是：1 个尾数符号位，8 个偏码表示的指数位(偏值＝127)，23 个尾数位。double 由 64 位组成，由高到低依次是：1 个尾数符号位，11 个偏码表示的指数位(偏值＝1023)，52 个尾数位。通过下列公式，可以由尾数和指数计算出所代表的实数值：

$X = \pm 1.\text{尾数} * 2^{(\text{指数}-\text{偏值})}$

下列几种情况下，此公式不成立：

指数＝000...0 且尾数 ＝00...0，则 X ＝± 0

指数＝000...0 且尾数!＝00...0，则 $X = \pm 0.\text{尾数} * 2^{(1-\text{偏值})}$

指数＝11....1 且尾数 ＝00...0，则 X ＝±∞

指数＝11....1 且尾数!＝00...0，则 X 是一个无效数

在 Turbo C 的头文件 value.h 中，相应于实数所能达到的最大最小值，定义了如下几个符号常数：

```
#define  MAXFLOAT     3.37E+38
#define  MINFLOAT     8.43E-37
#define  MAXDOUBLE    1.797693E+308
#define  MINDOUBLE    2.225074E-308
```

实常数是按 double 格式存放的，如果想按 float 格式存放，则必须加后缀 F，如：1.23E+4F。

当把实数强制转换为整数时，采取的是“向零靠拢的算法”，如：

float值:	65432.6	−65432.6
转换为long:	65432	−65432
转换为unsigued:	65432	104

如果不希望“向零靠拢”，而希望“四舍五入”，则必须由程序员自己处理。一种办法是先加上(符号位／2)，然后再进行类型转换。应该注意的是：如果被转换的实数值超过了目标类型的表达范围，则会产生错误。例如上面的 float 值−65432.6 转换为 unsigned int 值时，由于超过了目标范围，所产生的 104 就是错误的。在 65432.6 转换为 unsigned int 的 65432 以后，可以用 printf 的%u 格式输出，如果用%d 格式输出，则会得到错误的结果。

1.1.3 指针之间的转换

关于各类指针之间的转换请见下一节以及第 11 章。

1.2 指　　针

1.2.1 指针说明

指针是包含另一变量的地址变量。它的一般说明形式，如 int　* fd，其 fd 是一个指向整型变量的指针。比较复杂的指针说明，如 * (* pfpi)( )，可按如下几个原则来理解：以标识符为中心，一对方括号一般表示数组，一对圆括号一般表示函数或强调某一优先顺序，方括号对和圆括号对为同一优先级，方括号和圆括号比 * 号优先级高。以下几例解释了这些原则的应用。

int　* fip( )，因圆括号优先级高，故 fip 先与圆括号结合，说明 fip 是一个函数，这个函数返回一个指向整数的指针。

int　(* pfi)( )，因两对圆括号为同一优先级，故从左到右，pfi 是一个指针，这个指针指向一个函数，这个函数返回一个整数。

int　* par[ ]，因方括号比 * 号优先级高，故 par 是一个数组，这个数组的每一个元素是指向整数的指针。

int　(* ptr)[ ]，因方括号与圆括号为同一优先级，故 ptr 是一个指针，这个指针指向一个数组，这个数组的每一个元素是一个整数。

int　* (* pfpi) ( )，pfpi 是一个指针，这个指针指向一个函数，这个函数返回一个指向整数的指针。

1.2.2 指针与地址

指针在使用之前必须初始化，给指针赋地址的方法一般有如下几种：

第一种很容易理解，通过取地址操作符取变量(包括结构变量)的地址，如：

```
char__c="a",        * ptr__char;
ptr__char=&c;
```

第二种是数组，因为不带方括号的数组名等效于数组中第一个元素的地址，即数组名也可看作是一个指针，所以有两种办法。如：

```
char myname[31],    * ptr;
ptr=myname;
```

或

```
ptr=&myname[0];
```

第三种是动态分配的一块内存，这时往往带有类型强制转换，但应注意当内存不够时，可能返回一个空(NULL)指针。如：