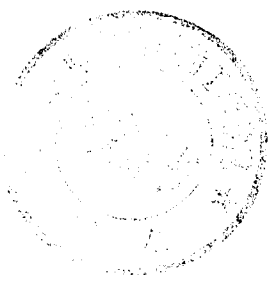
Turbo C 实务入门

魏献旺　赖荣枢
魏宾甫　林志峰　编著

北京航空航天大学出版社

Turbo C
实务入门

魏献旺　赖荣枢
魏宾甫　林志峰　编著



北京航空航天大学出版社

内 容 简 介

Turbo C 提供了一个优良的环境让程序员开发程序。因此，对于想学习 C 语言的人来说，Turbo C 是最合适的。本书讲述了 Turbo C 的各种程序编写技巧。它的主要内容是：内存与 I/O；C 的初步；流程的控制；数组与指针；宏与函数；变量与层次；自定义数据与类型；文件处理及 Turbo C 的公用程序 CPP.EXE 的使用。本书由浅入深，内容详细，适合初学 C 语言者阅读。

图书在版编目(CIP)数据

Turbo C 实务入门/魏献旺等编著. —北京：北京航空航天大学出版社，1995.7

ISBN 7-81012-575-3

Ⅰ.T… Ⅱ.魏… Ⅲ.C 语言-程序设计 Ⅳ.TP312C

中国版本图书馆 CIP 数据核字（95）第 04674 号

Turbo C 实务入门

Turbo C SHIWU RUMEN

魏献旺 赖荣枢 魏宾甫 林志峰 编著

责任编辑：许振伍

*

北京航空航天大学出版社出版

（北京市学院路 37 号；邮编 100083；发行科电话 2015720）

北京朝阳科普印刷厂印刷

新华书店总店科技发行所发行 全国书店销售

*

787×1092 1/16 印张：16 字数：403 千字 印数：5000 册

1995 年 12 月第 1 版 1995 年 12 月第 1 次印刷

ISBN 7-81012-575-3/TP·163 定价：25.00 元

版权号 图字：01-95-702 号

序　言

C 语言是目前世界上最著名的系统程序语言。近年来，被广泛地使用在个人电脑上，也是目前最为适合个人电脑使用的系统程序设计语言。其原因是 C 十分的娇小，在有限的内存中却能任凭设计员的想像力来发挥它的功能。不仅如此，C 语言又兼具了低级语言的能力，也就难怪它会如此的受到欢迎了。

在 C 语言中以 Turbo C 最适合初学者与深入研究的程序设计员，这是因为 Turbo C 提供一个集成环境让程序设计员来开发程序，所以书中将以 Turbo C 为主体来讲述 C 语言的各种程序设计技巧。除此之外还包含变量、结构、自定义类型、函数指针、程序库管理、内存管理、文件处理等的详细的解释。

过去，笔者在学习 C 语言过程中，常常觉得找不到一本适合初学者学习的参考书。因为书店买得到的书籍不是写的太深奥，就是太过于粗浅，除了学会一些基本的函数外，根本接触不到 C 语言的精髓。在有了这些年的学习经验后，希望能借这本书的出版，给想接触 C 语言的同行们一本经过慎重的安排，由浅入深、内容详细的参考书籍，希望能够帮您在程序设计的领域中一展所长。相信您会认同，这本书将会是您在学习 C 语言的过程中最有力的帮手。

魏献旺负责第三、五、六、九章
魏宾甫负责第一、四章
赖荣枢负责第二章
林志峰负责第七、八章

出版者的话

虑及读者的经济承受能力，我们没有将录有本书所有程序的软盘附在书后。如果读者需要，请与本社发行科邮购组魏兰英联系（地址：北京市学院路 37 号，邮编：100083；电话：2015720）。软盘一张定价为 25 元。如邮寄，另加邮购费 8 元。

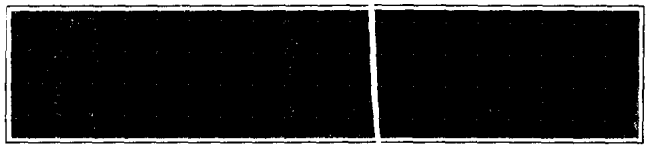
原编辑部序

“系统编辑”是我们的编辑方针，我们所提供给您的，绝不只是一本书，而是关于这门学问的所有知识，它们由浅入深，循序渐进。

C语言具有速度快、轻巧、应用广泛等功能，所以一推出，就受到程序设计员的青睐。书店中有关C语言的书籍，比比皆是，但真正能将自己的心得编写成书，却寥寥无几，本书即是其中之一。内容除了各种程序设计技巧之外，还对函数有深入的介绍，看完此书，可以深知C语言的精髓，并成为C语言设计的高手。

同时，为了能有系统且循序渐进学习相关方面的丛书，我们以流程图的方式，列出各有关图书的阅读顺序，以减少学习这门学问的探索时间，并能对这门学问有完整的知识。如果在这方面有任何问题，欢迎来函联系，我们将竭诚服务。

目　录



内存通常所指的就是 ROM(Read Only Memory)，RAM(Random Access Memory)等。在整部计算机中都占着极重要的地位。不论是程序的输入、执行，还是数据的处理，都需要通过内存。若对内存处理不当，就容易产生系统数据的混乱或者死机的结果。由此可知内存的重要性。

在 C 语言中对数据的处理非常多，尤其是 C 语言中的指针，更是需要正确的内存概念才能运用自如。为了以后各章易于介绍程序的原理，先说明内存的基本知识，当做读者学习 C 语言的学前入门。

0-0-0 分片式内存管理

所谓分片式内存管理就是将实际上是连续的内存分成一片一片。其优点是易于管理，且使内存空间有效的被使用。这也是 DOS 一直所使用的内存管理方式。

Segment 是段位址，offset 就是偏移量(如图 0.0)。段地址与偏移量都是是由二个字节所储存。所以若段位址不改变则偏移量可达到 64K 字节，也就是说一个 Segment 可控制 64K 字节的内存空间。

在地址的表示法中通常用 16 进制来表示其地址，图 0.1 中数字之前有个“0x”表示这个数字是 16 进制。地址的表示法有二种，即图 0.1 左、右二种。

其实，图 0.1 左、右二边的表示法的意义是完全相同的，也就是说 0x10001 与 0x1000:0001 代表的是同一内存地址。很奇怪吧！其实两者之间是经过一个固定的转换方式来换算的，其转换方法就是将段地址左移一位(即左移 4 bites)，然后与偏移量相加。如此一来转换就完成了(参考图 0.2)。由此可知段地址和偏移量的配合，可使系统使用 1M 字节的内存空间，这就是 8086/8088 模式，也就是所谓的实模式(Real Mode)的管理方式。

0-0-1 数据在内存的储存方式

在 C 中的数据类型有许多种，如字符类型、整数类型、浮点数、结构等。但它们在内存中的存储方式大都相同，因为常见的数据类型不外乎是以字符与字符串的形态存放在内存中的，这是简单的部分，而要注意的是一个完整的内存地址是如何存放在内存中的。因为这涉及到 C 语言的指针的概念，所以是学习 C 语言必须具备的知识。

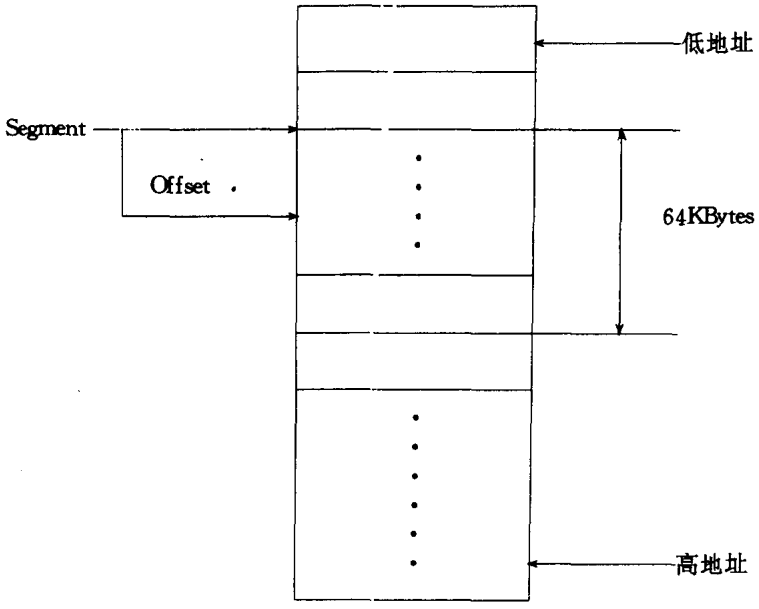

低地址
Segment
Offset
64KBytes
高地址

图 0.0

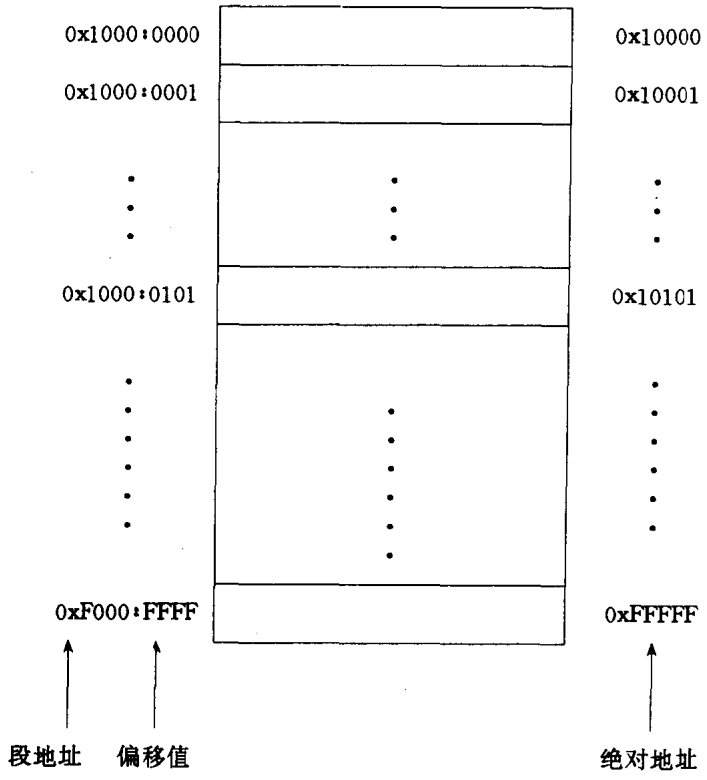

0x1000:0000
0x10000
0x1000:0001
0x10001
0x1000:0101
0x10101
0xF000:FFFF
0xFFFFF
段地址
偏移值
绝对地址

图 0.1

在此就以 Turbo C 的字节(1Byte)、整数(2Bytes)、长整数(双字,4Bytes)来说明其存储方式。为了说明方便,用 16 进制(即数字前加上 0x)来表示存放在内存中的数值。

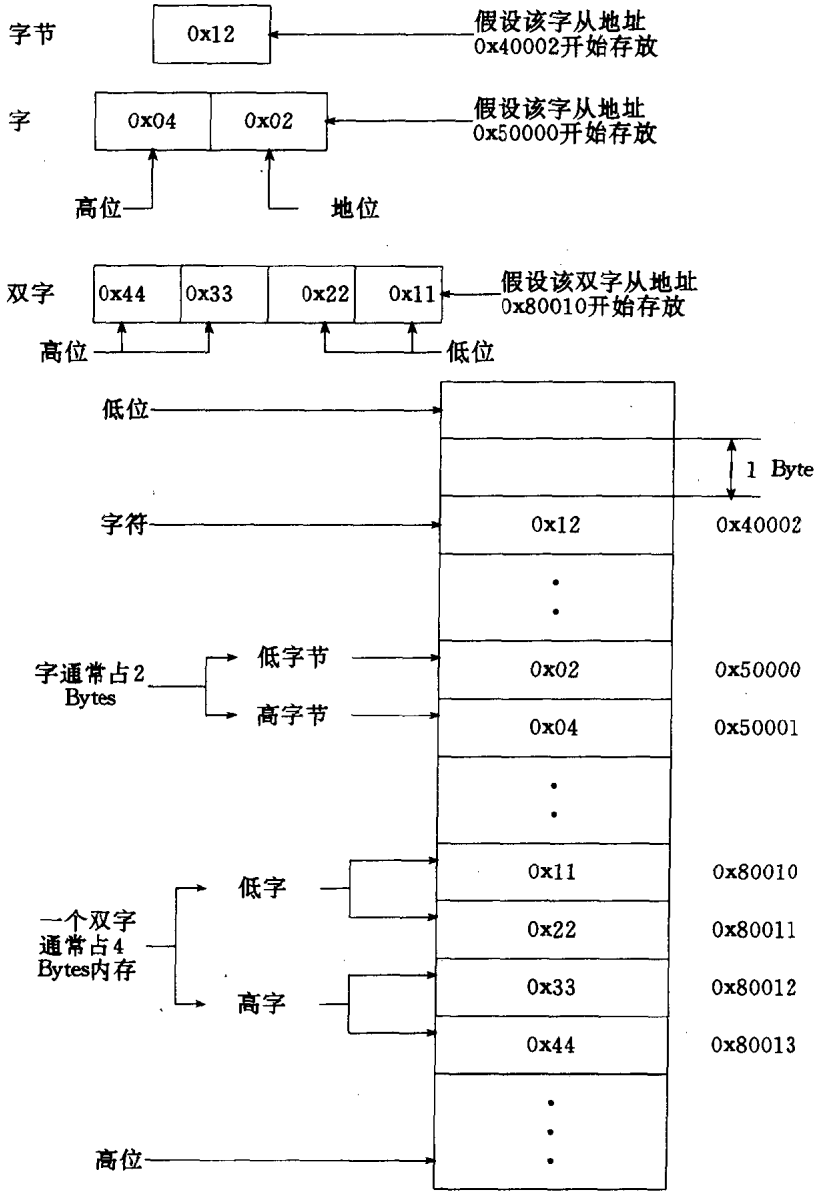


图 0.3

由图 0.3 可知,不论是字或双字其存放规则就是"高者在高,低者在低"。就以字为例,一个字可分成高位字节和低位字节,低位字节放于低地址,高位字节放于高地址。而双字分成高字与低字,其中每一个字再分成高位字节和低位字节再按"高者在高,低者在低"的方法来存放,所以原理与字是完全相同的。而绝对地址的存放方式就等于存放一个双字的数据在内

存的方法(因为绝对地址也须要用 4 Bytes 来存放)。

0-0-2 扩展与扩充内存

DOS 5.0 已具备使用扩展内存(Extended Memory)与扩充内存(Expanded Memory)的能力。这是DOS 的一大改进。本书将简单说明其管理的方式。当然主要是希望读者能建立起较明确的关于内存的概念。

其实,在 8086/8088 时代之所以受限于传统内存(即所谓的 640KBytes),除了因为 8086/8088 本身只具备 20 位地址线外,DOS 的功能不支持也是原因之一。而 80286/80386/80486 等因为地址线皆在 20 位以上,所以它们可使用的地址范围皆在 1M Bytes 以上,加上 DOS 已提供管理扩充内存和扩展内存的驱动程序,所以,了解其管理方式也就成了必要的课题。

所谓扩展内存与扩充内存就是如图 0.4 所标示的区域。DOS 提供了与 XMS(Extended Memory Specification)兼容的驱动程序,即 Himem.sys。如此一来不但 DOS 本身可载入高端内存,许多公用程序也可通过驱动程序载入高端内存。扩充式内存的管理是通过 emm386.sys,如果机器本身未含有扩充内存,也可使用扩展内存来模拟扩充式内存。

UMA(Upper Memory Area)是常规内存(640M Bytes)与 1M 内存间的内存区,一般而言这个区域是分配给各种显示卡和 BIOS(BASIC INPUT/OUTPUT SYSTEM)来使用的,但通常会有许多小的块(一般称的为 Upperd Memory Block)未被任何显示卡所占用,而 DOS 就使用该块,用映射的技巧将未使用的扩展内存或显示卡上的扩充内存通过驱动程序映射到 UMB 上,以达到使用更多内存的目的。

图 0.4 所中示的 HMA(High Memory Area)是扩充内存的最低地址区(即 0xFFFF:000F～0xFFFF:FFFF 共约 64KBytes),这是 himem.sys 故意将扩充内存最低的 64K Bytes 保留下来做为 HMA 的,并且是 80286/80386/80486 才有的区域。

读者或许很奇怪,既然 80286/80386/80486 可定址到 1MBytes 以上,那为何还需要那些驱动程序呢? 原因是 DOS 5.0 的是设计在 8086/8088 模式(真实模式)下的操作系统,因此就算是 80286/80386/80486 的机器在 MS-DOS 下仍被设定为实模式,这就是为什么在 8086/8088 能执行的软件也可在 80286/80386/80486 上执行的原因(简单的说就是为了兼容)。但为了不浪费系统资源,于是就设计了一些驱动程序,而通过这些驱动程序去使用或管理扩展内存或扩充内存。

至于其他与内存相关的驱动程序,如虚拟磁盘(ramdrive.sys)、磁盘 I/O 高速缓存程序(smartdrv.sys)等,请自行查阅 DOS 的使用手册,在此就不再细说了。

一般而言,I/O 就是输出与输入,如键盘、显示、打印机等等。而 C 可以说是没有 I/O 的。很奇怪吧! 这句话并没有写错,完全是观点的问题。以下本书将介绍 C 语言的所谓无 I/O 的问题。

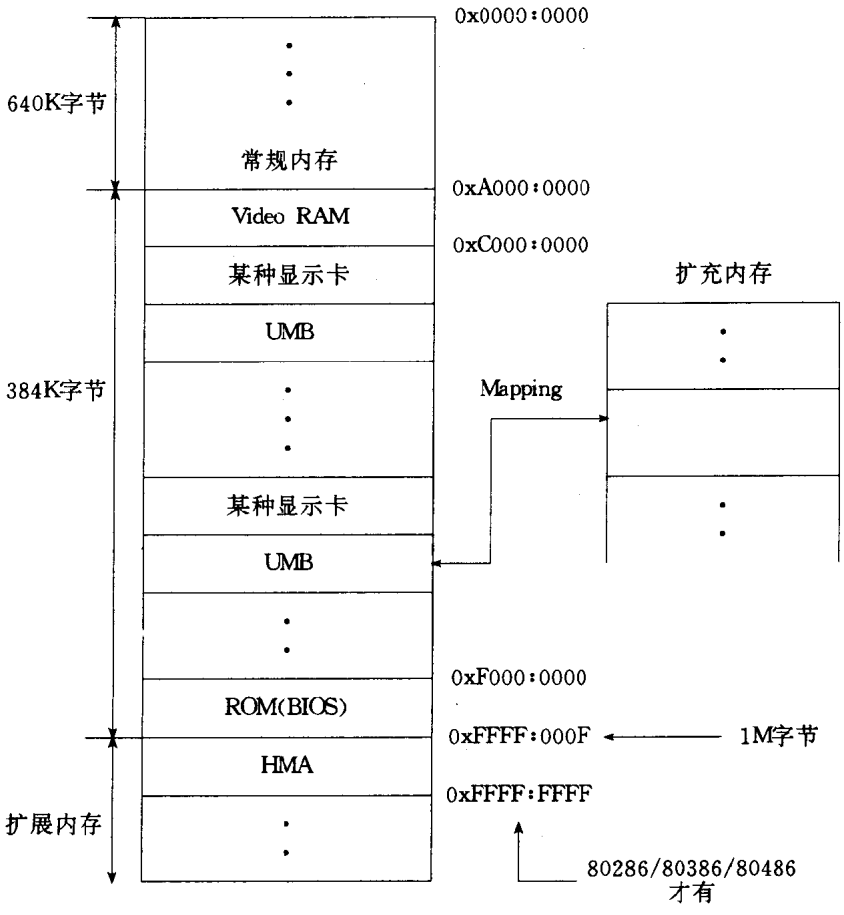


图 0.4

0-1-0 高级 I/O 与低级 I/O

真正的低级 I/O 就是直接的控制硬件。当然,这是不容易的,不管是对专业的程序员或一般的用户,都是一个非常大的负担与困扰。不过那些以能控制硬体为傲的人不在此范围之内。所以有 BIOS 的 I/O 和通过 DOS 的 I/O,这就是为什么需要操作系统的原因。

而 C 中也并不是真的没有 I/O,只是 C 的 I/O 是由函数来完成的,本身并不具备 I/O 的能力。因此 C 语言有没有 I/O 可以说是因人而异,不过这反而造就了 C 语言在通用性上的成就。

C 语言的函数(如 scanf(),printf()等)就是高级 I/O 的实例。那何为高级 I/O 呢? 讲简单一点就是程序员只需通过不同的函数就可以达到一些 I/O 的目的,而不需了解函数中的结构,只需知道函数的输入与输出,当给予函数适当的输入时就可以得到正确的输出。

读者应该听过低级语言比高级语言难学吧,其实这不是没有道理的。在低级语言中,以 Assembly 为例,所有输入、输出的工作都需由程序员一手包办。而在高级语言中一行程序就可能是 Assembly 中的十几行,甚至数十行程序。所以读者应该知道,为什么要有高级语言、

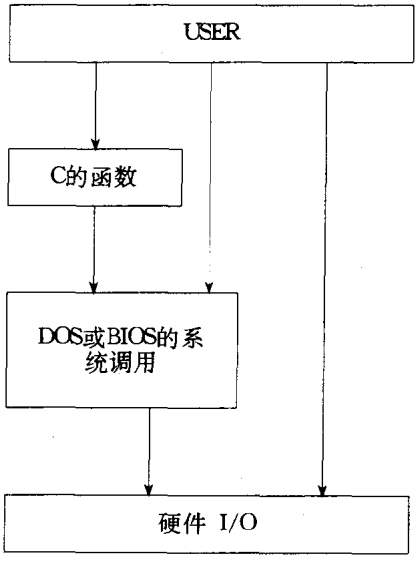


图 0.5

高级 I/O。

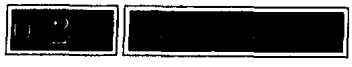
0-1-1 数据流(Stream)

数据流(Stream),从字意上就可以知道是数据所形成的流,当然这是比较抽象的形容方式。其实,可以将其解释为数据的来源或数据的去向。就以文件为例,若将数据放入数据流就如同将数据写进文件,反之,若将数据从数据流取出就如同从文件中读入数据。

在 DOS 系统下,DOS 将外部设备(如显示器、打印机、键盘等)都当成文件去处理,故外部设备也可视数据流。既然数据流与文件是相等的,那何必要有数据流呢?其实不然,如果程序员直接使用文件方式管理外部设备,则文件的相关数据(如文件指针、文件名称、文件编号等)必须由程序员自行记录。而如果使用数据流的方式,只要声明函数所需的数据流变量,则以后只要将此变量传给函数,就可使函数正常工作,而当文件有变动时函数也会自动的更改数据流变量的内容,这样一来使程序员省了不少功夫。

C 语言就是利用这种数据流的方式来编制函数,当然这只是一种理论,而程序在做 I/O 时,内部依然是通过 BIOS 或 BOS 来进行的。所谓的数据流变量其实就是把文件在使用时所需要的数据以结构的方式组合在一起,也就是使用一个文件所需要的结构体变量,来存储所有有关该文件的数据。当在对文件做 I/O 时就不再需要输入一大堆数据给函数,只要传入结构体变量即可。

或许,读者阅读到此深感迷惑,这是正常的,这些相关的理论(如结构体)将于后面各章详述。在读完本章之后,读者只需有个大概的认识,这样在后面各章介绍时将可事半功倍。



本书在以后各章中将以条状图(如图 0.1)的方式来讲述与内存有关的各项操作,读者

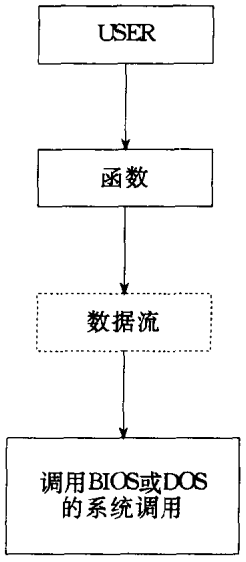


图 0.6

务必具备本章的基本知识,以便日后在阅读各章时不会有太大的困扰。