高等学校教学用书

冶金反应问题的计算机求解与软件开发

方 觉 编著

GAODENG XUEXIAO JIAOXUE YONGSHU

冶金工业出版社

冶金反应问题的计算机求解与软件开发

东北大学　方　觉　编著

北　京
冶 金 工 业 出 版 社
1 9 9 9

内 容 简 介

本书重点针对冶金学科的本科生教学，也可作为其他化工类本科生或研究生的教学参考书。教学内容以四个方面为主线。其一是有关C语言的知识，通过本课程的学习应能达到熟练使用TURBO C的程度。其二是编程技巧，书中给出了大量程序实例并进行了详细讲解，例如汉字使用、鼠标使用、界面开发、数据管理等。其三是计算机的应用与专业知识的结合，书中详细讨论了解决冶金反应热力学和动力学问题的原则和手段，并给出了实用性的例程。其四是综合型软件的开发，书中例程的选材有意识地围绕“冶金热力学数据手册”的需要进行，以这个软件的开发为例，使学生在实践中掌握大型软件的开发技巧。

图书在版编目(CIP)数据

冶金反应问题的计算机求解与软件开发/方觉编著．—北京：冶金工业出版社，1999.8
高等学校教学用书
ISBN 7-5024-2331-1

Ⅰ．冶… Ⅱ．方… Ⅲ．冶金-反应-计算机模拟-软件开发-高等学校-教学参考资料 Ⅳ．TF1

中国版本图书馆CIP数据核字(1999)第20958号

出版人　卿启云(北京沙滩嵩祝院北巷39号，邮编100009)
责任编辑　宋　良　责任校对 侯　瑢
北京华威冶金印刷厂印刷；冶金工业出版社发行；各地新华书店经销
1999年8月第1版，1999年8月第1次印刷
787mm×1092mm　1/16；14.75印张；342千字；225页；1-1200册
21.00元
冶金工业出版社发行部　电话：(010)64044283　传真：(010)64013877
冶金书店　地址：北京东四西大街46号(100711)　电话：(010)65289081

前　言

在过去几年里，计算机应用得到了迅猛的发展。计算机已经普及到冶金生产和科研的各个角落，应用水平日新月异。形势迫人，冶金学科的计算机教学如果没有一个大幅度的提高，已经不能满足社会的需要。

计算机应用是受高校学生偏爱的一门课程，利用课余时间学习计算机的大有人在。遗憾的是，真正能够达到较高水平的仍然是少数。显然，造成这种状况的责任不在学生，而在于缺乏系统的讲授和正确的引导。在计算机学习中有两种倾向值得重视。其一是将精力过分倾注于学习商品软件的使用方法。这种学习固然是必要的，但若过分则会影响水平的提高。其二是计算机应用与本专业脱节。计算机软件只有依附于一个处理对象才能存在。我们对这个对象的了解越深入，编写出高水平软件的可能性就越大。对于冶金专业学生来讲，将冶金问题作为处理对象是顺理成章的事。有些学生计算机用得很熟练，但解决实际问题或编写软件的能力却不高。没能处理好专业与计算机的关系，是造成这种状况的主要原因之一。

本书的内容设置，在很大程度上是基于以上考虑。教学应围绕两个中心进行：一是将提高计算机应用水平与提高专业水平结合起来，培养学生应用计算机解决专业问题的兴趣。二是使学生的编程水平逐步达到能够编写大型复杂软件的程度。

全书要点如下：

第一章主要讲述有关 TURBO C 的必要知识。没有这方面的知识，后续课程将无法进行。如果学生已经学习过 C 语言，本章内容可少讲或不讲。但其中汉字应用函数部分不应舍弃。

第二章内容是数据管理，实质是指导学生开发出一个数据库软件。其中涉及到大量有价值的函数开发工作，例如鼠标控制函数、数据输入输出函数、图形生成及调用函数等。应当尽量使学生掌握这些函数的原理，主要程序则可更多地交与学生自己去理解。

第三章以热力学计算为主。通过四个具有实用价值的例程开发讲述冶金热力学的计算原理、方法及其在计算机上的实现。

第四章的主要内容是冶金动力学问题的特点和研究方法。通过两个实例具体讲述了冶金动力学问题的模型建立和计算机解题过程。

第五章讲述综合型软件的设计和开发。例程是一个集菜单程序管理、数据管理、热力学计算、图形展示、图形打印、文件管理等功能于一体的实用软件“冶金热力学数据手册”。

本书例程均经过编者的精心调试和试运行，很多函数或程序具有较广泛的使用价值，最后形成的综合软件“冶金热力学数据手册”，除可起到一个加强型无机物热力学数据表的作用外，还可随时提供任意反应 $\Delta G^{\ominus}$ 与 T 及 $\ln K^{\ominus}$ 与 $1/T$ 的线性关系式和图形。

东北大学杜鹤桂教授、车荫昌教授和北京科技大学苍大强教授对本书进行了审查，并提供了建设性的修改意见。东北大学教务处，特别是陈文娇女士，对本书的写作给予了大力的支持和指导。笔者对此再次表示衷心的感谢。

方　觉

1998 年 6 月

目　录

1 TURBO C 基础知识

1.1 概述

编程的目的是为了解决某个或某些问题。这些特定的问题就是处理对象。能否编好一个程序，取决于编程者两方面的知识。其一是专业知识，即对对象的了解。对于我们来说就是对冶金学理论和过程的掌握。只有深入地了解冶金，才能使用计算机有效地模拟冶金过程，得到预期的结果。其二是对计算机的掌握。对计算机了解越深入，越有希望编写出简明实用或功能强大的程序。

我们不能指望计算机专家对各行各业都有深入的了解，特别是冶金这样的学科。但冶金行业也确实需要大量的专门软件来解决生产和科研中的问题。大幅度提高冶金专业人员的计算机使用水平，进而培养出一些本专业的计算机专家，可能是解决上述矛盾的最现实的措施。这就是本课程的基本出发点。

计算机技能中，语言的使用占有重要的地位。哪怕要编写最简单的程序，也起码要会使用一种计算机语言。计算机语言形形色色，从汇编那样的低级语言到 BASIC 和 PASCAL 那样的高级语言，都可供我们选择。一般说来，只要很好地掌握一种高级语言就可以解决范围很大的实际问题。但这并不是说我们可以随便选择一种语言来进行我们的编程工作。事实上，各种语言都有其特点，可能特别适合于解决某类问题，对解决其他一些问题则可能显得软弱无力。例如，汇编语言可以深入到机器内部，也可以直接访问外围硬件。而且它的编译和运行效率也较高级语言高得多。但要使用汇编语言解决大量的数学运算问题则会事倍功半。再例如，使用 BASIC 语言可以方便地解决各类数学运算问题，用其编程，程序具有极强的可读性，这一点对初学者特别有利。但若想使用 BASIC 编写出复杂的大型软件，显然是不合适的。因此，我们应当根据对象的特点来决定采用的语言。

从另一个角度讲，差不多每个编程的人都有一种偏爱的语言。他对这种语言下的功夫较多，掌握较好，编程时也就不太可能根据需要变换语言。如果这个受偏爱的语言正好是一种不适合编写大型复杂软件的语言的话，那么就很难指望这个编程者的水平有较大程度的提高。遗憾的是，这种情况在非专业的程序员中并不少见。如果我们能够在一种适当的语言上多下些功夫，养成使用它的习惯，则可望形成一种良性循环，将我们的编程水平提高一个档次。

我们选择 TURBO C 作为本课程的编程语言，在很大程度上是基于以上的理由。C 语言与汇编语言相似，允许程序直接与机器内部打交道，直至字节和位。它也支持程序访问和控制外设寄存器。因此，C 语言被称为高级汇编语言。但 C 语言的功能实际上要比高级汇编语言强得多，它的层次介于低级语言和高级语言之间，因此也被称为中级语言。C 语言的特点可归纳为以下几点：

1)C 语言是一种结构化程序设计语言，适合于编写大型软件；

2)拥有丰富的数据类型和运算符；

3)具有直接访问硬件的能力，编译和运行效率可与汇编语言媲美；

4)程序可读性强,易于编程和维护;

5)为用户提供丰富的库函数,功能不弱于高级语言;

6)库函数与C语言相对独立,C语言不含依赖于硬件的语句,因此可移植性强。

由于以上特点,C语言被广泛地应用于操作系统、图形处理、数据库管理、多媒体等软件的开发。目前大型和复杂软件系统多采用C语言。

TURBO C是Borland公司于1987年推出的一种专业水平的C系统。TURBO C完全符合ANSI C标准。它支持普通C编程和MS-DOS下80X86的C编程。它为用户提供了一个集编辑、编译、连结和调试于一体的集成开发环境。它的库函数丰富多彩,也允许用户自己定义数据类型和库函数。

本章主要讲述编程前必备的TURBO C知识,这些知识的应用及其他有关内容将在编程用到时再详细介绍。

1.2 内存

1.2.1 DOS内存管理

我们的编程工作主要在DOS环境下进行,TURBO C的内存使用离不开DOS的大环境。因此我们应当首先对DOS的内存管理有一个初步的了解。

内存是内部存储器的简称,即memory。程序或数据必须首先装入内存才能被执行或使用。存储器可大致分为两种。其一是只读存储器ROM(Read Only Memory)。这种存储器是一种固化的软件,不会因为断电而消失。其中的程序和数据只能被读取,不能被修改或者删除。其二是随机存取存储器RAM(Random Access Memory)。这种存储器是用来装入用户程序及数据的部件,其中的内容可随时读出或改写。一旦计算机断电,RAM中的内容就会丢失。计算机内存的大小就是以RAM的配置来衡量的。

内存的基本单位是位,即bit。一位内存可储存一位二进制数表达的信息量1或0。计算机内存以8位为一个单元,称为字节,即byte,简称B。一个字节或8位二进制数可用两位十六进制数表示。当我们查看内存时,一般会得到以十六进制数表示的信息。

字节以上单位间的进位关系是1024:

1KB(千字节)=1024B

1MB(兆字节)=1024KB

1GB(吉字节)=1024MB

1TB(太字节)=1024GB

内存的访问是依据其地址进行的,没有地址的内存毫无用处。因此,计算机允许配置的最大内存量由其寻址能力决定。

目前,微机的主流是以IBM PC系列机为中心形成的。第一代IBM PC机以8088为处理器。8088的地址总线为20根,它的地址是20位的,范围为00000H至FFFFFH。数字后的字符H表示该数字是十六进制的。这个范围就是8088的地址空间,它的寻址能力是1MB。这1MB的地址空间被分为两个部分。640KB以上的部分作为保留区。保留区的高端,即地址较高的一端被ROM占用。低端被视频存储器VIDEO RAM占用。VIDEO RAM是装在监视器适配卡上的。中间一段未被占用。保留区的管理用不着DOS,DOS只需要管理余下的640KB内存,也就是地址为00000H至9FFFFH的低端部分。

当时谁也没有料到 IBM PC 会有后来那样的迅猛发展，因此 DOS 按 8088 的需要被设计成仅能管理 640KB 内存的基本结构。很多软件就在这种条件下被开发出来并得到广泛的应用。软件的丰富程度在很大的程度上决定了微机市场。因此在以后的 DOS 版本和微处理器的升级过程中都要照顾到那些正在流行的软件。目前微机内存已得到了爆炸性的发展，但由于上述原因，DOS 直接管理内存的能力仍然局限在 640KB。

8088 那样 1MB 的内存称为常规内存。其中低端 640KB，即 DOS 可直接控制的部分称为基本内存。习惯上将 640KB 以外的 RAM 称为额外内存。DOS 要使用额外内存必须借助特殊软件才行。

通常使用两个规范来使用额外内存。其一是 Microsoft、Intel、Lotus 和 AST 推出的 XMS (eXtended Memory Specification)。其二是 Lotus、Intel 和 Microsoft 推出的 EMS (Expanded Memory Specification)。

XMS 一般称为扩展内存规范。DOS 通过一个名为 HIMEM. SYS 的驱动程序来实现对 XMS 的管理。这个程序应通过系统配置文件 CONFIG. SYS 在开机时调入内存。图 1-1 是 XMS 管理下的内存配置与常规内存配置的对比。

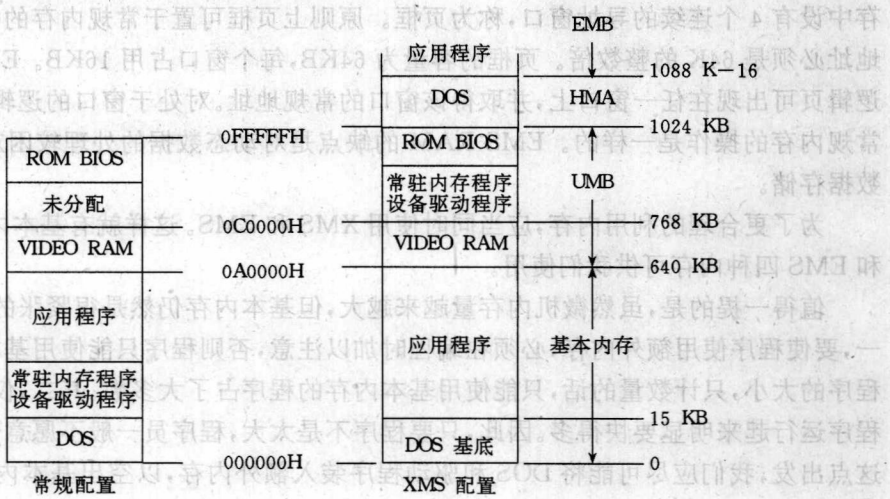

图 1-1　内存配置示意图

XMS 为 80X86 定义了一个软件接口，实现了以下三种内存的存取：

(1)上内存块 UMB

UMB 就是保留区。保留区中，在 VIDEO RAM 与 ROM 之间有一段未被占用的地址，通过 XMS 驱动程序可以使用这一段地址来配置 RAM。UMB 可用于装入一些常规配置中必须装入基本内存的常驻程序，特别是设备驱动程序。当使用 EMS 驱动程序时，可将 64KB 的扩充内存页框(Page Frame)装入 UMB。

(2)高内存区 HMA

HMA 是 XMS 管理下的一块特殊区域，它是通过启用第 21 条地址线 A20 实现的。存储器的物理地址等于段地址乘以 16 再加上偏移量。当 A20 处于 off 状态时，如果该值超过 20 位，则将高位截掉。在这种情况下，0000H：0000H 至 FFFFH：000FH 生成的物理地址

范围是00000H至FFFFFH。覆盖了1MB的地址空间。这时FFFFH：0010H至FFFFH：FFFFH与0000H：0000H至0000H：FFFFH是等价的。如果A20处于on状态，则FFFFH：0010H生成的物理地址是100000H，FFFFH：FFFFH生成的物理地址是10FFEF，覆盖了65520字节的地址空间。这一段内存就是HMA。HMA只能作为一个整体供一个程序使用。所以应当尽量用于存放长度与HMA相当的程序，一般用于装载除基底以外的DOS(约45KB)。

(3)扩展内存块EMB

EMB分配于HMA以上的区域中，无法在实方式下访问。

EMS是一个软硬件混合的规范，称为扩充内存规范。实际的扩充内存是通过扩充卡加入计算机的，卡上配备专用的管理硬件和软件。EMS实质上是一个软件通讯协议，因此可用软件仿真。目前一般使用DOS提供的驱动程序EMM386.EXE来仿真扩充内存，支持1MB以上内存区的使用。这个驱动程序可作为DOS外部命令使用，也可通过CONFIG.SYS装入。

EMS RAM的存储实体是16KB的逻辑页。因此，EMS内存也被称为扩页内存。常规内存中设有4个连续的寻址窗口，称为页框。原则上页框可置于常规内存的任意部位，但起始地址必须是64K的整数倍。页框的容量为64KB，每个窗口占用16KB。EMS RAM的任一逻辑页可出现在任一窗口上，并取得该窗口的常规地址。对处于窗口的逻辑页进行存取与对常规内存的操作是一样的。EMS RAM的缺点是对动态数据的处理较困难，最好用于静态数据存储。

为了更合理的利用内存，应当同时使用XMS和EMS。这样就有基本内存、UMB、HMA和EMS四种内存可供我们使用。

值得一提的是，虽然微机内存量越来越大，但基本内存仍然是很紧张的。原因有二。其一，要使程序使用额外内存，必须在编程时加以注意，否则程序只能使用基本内存。如果不论程序的大小，只计数量的话，只能使用基本内存的程序占了大多数。其二，仅使用基本内存的程序运行起来明显要快得多。因此，只要程序不是太大，程序员一般不愿意使用额外内存。从这点出发，我们应尽可能将DOS和驱动程序装入额外内存，以空出基本内存供应用程序使用。

1.2.2 TURBO C的存储模式

80X86的存储器采用分段结构，以64KB为一段。DOS支持两种可执行文件，COM文件和EXE文件。COM文件在装入内存时仅占用一个段，寻址是在段内进行的。这样，段地址就是一个定值，寻址仅靠偏移量进行。这种特征使程序长度不可能超过64KB，但也使运行速度大为提高。TURBO C生成的是EXE型文件。这种类型的程序可占用多段内存，适合解决大型复杂的问题，但装入和执行速度则较慢。TURBO C以段的使用情况为标准提供了六种存储模式供编程者选择：

(1)Tiny

Tiny模式即微型模式。在这种模式下TURBO C将程序的代码和数据全部编译到同一个64KB的段内。也就是说，采用这种模式工作时产生的EXE文件在执行时只占用一个段，满足生成COM文件的条件。因此，这种文件可使用DOS命令EXE2BIN转化为COM文件。这种模式的内存使用情况见图1-2。

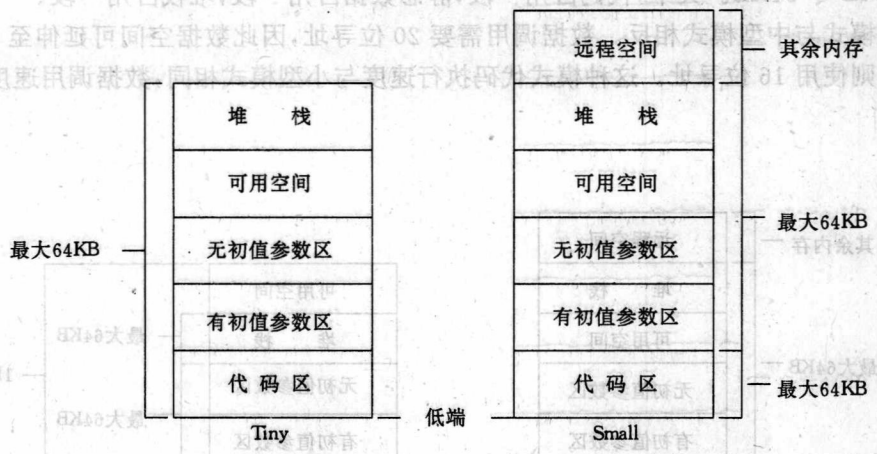

图 1-2　微型和小型存储模式内存使用示意

(2)Small

Small 模式为小型模式，图 1-2 给出了该模式的内存使用与微型模式的对比。以代码区和堆栈为界，内存中划分出一个区域。这个区域就是 TURBO C 控制的内存空间。这一点两种模式是一样的。在这一区域中，由低地址内存开始依次是代码区、有初值参数区和无初值参数区。堆栈则设置在该区域的最高端。处于这两个部分之间的是归 TURBO C 掌握的自由内存，可在程序运行中使用。这一块自由内存称为可用空间。

两个参数区，堆栈和可用空间都是用来储存数据的，统称数据区。其中两个参数区所储存的是静态数据。与微型模式不同的是，小型模式划归 TURBO C 的内存区域扩展到了两个段。代码占用一段，最长可至 64KB。数据占用另一段，最长也是 64KB。此外，TURBO C 控制区之外的自由内存形成了所谓远程空间。远程空间也可供程序使用。不过，TURBO C 无权管理远程空间，程序要使用远程空间时必须向 DOS 申请。

小型模式是 TURBO C 的默认模式。实际上该模式已经能够满足很大范围的编程需要。在小型模式中，代码和数据限制在各自的段内，分别采用 16 位寻址，运行速度接近微型模式。

(3)Medium

Medium 模式为中型模式，它与小型模式的内存配置差不多。区别是中型模式可设置多个代码模块，形成一个模组，每个模块最大为 64KB。这些模块可使用 LINK 命令连接成一个最大可至 1MB 的 EXE 文件。这种模式是为满足代码较长而数据空间需要量不大的情况而设置的，可控制 1MB 零 64KB 的内存空间。

在这种模式下，代码占用多段内存，要使用 20 位寻址结构。因此执行代码的速度较小型模式慢得多。数据则仍限制在一段之内，可采用 16 位寻址，因此数据调用速度与小型模式相同。

(4)Compact

Compact 模式为紧凑型模式，它与 Medium 模式互补，适用于代码不是很长但数据区较

大的程序。这两种模式的内存配置情况见图 1-3。在紧凑型模式下，TURBO C 可控制的内存空间也是 1MB 零 64KB。其中，代码占用一段，静态数据占用一段，堆栈占用一段。

紧凑型模式与中型模式相反。数据调用需要 20 位寻址，因此数据空间可延伸至 1MB。而代码调用则使用 16 位寻址。这种模式代码执行速度与小型模式相同，数据调用速度则慢得多。

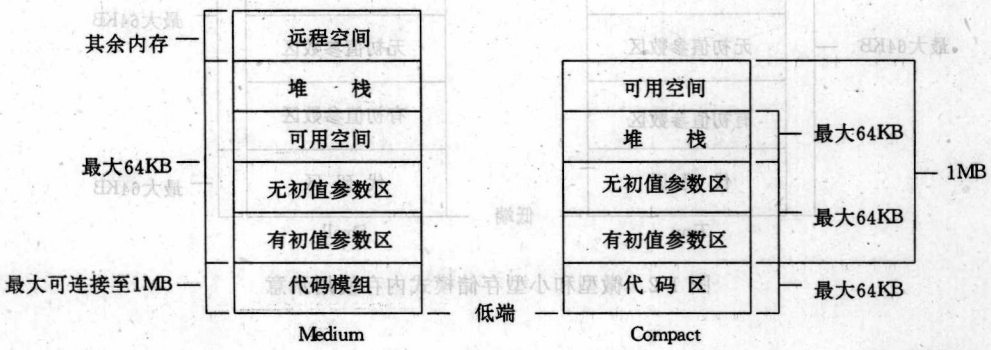

图 1-3　中型和紧凑型存储模式内存使用示意

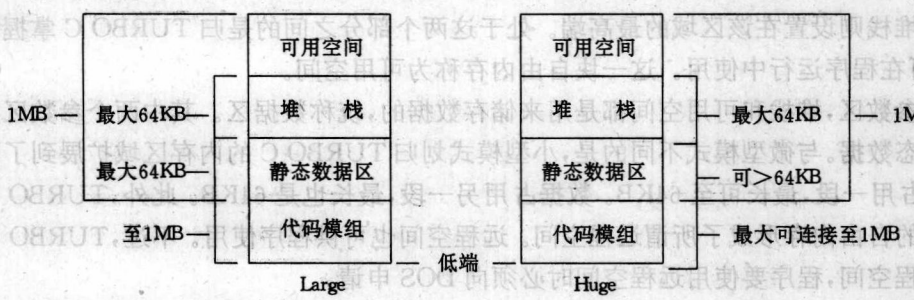

图 1-4　大型和巨型存储模式内存使用示意

(5)Large

Large 为大型模式。它的代码空间与中型模式相同，每个模块最大为 64KB，总长可连接至 1MB，需使用 20 位寻址。数据空间与紧凑型相同，使用 20 位寻址，也可延伸至 1MB。因此，大型模式可用的总内存达 2MB。它的代码执行和数据调用均很慢，总速度较前四种模式要慢得多。但它能够满足代码和数据都很长的编程需要。这种模式的内存配置见图 1-4。

(6)Huge

Huge 模式为巨型模式，内存配置见图 1-4。这种模式是运行速度最慢的一种。它与大型模式的区别是静态数据区扩展至多段，不再局限于 64KB。

下表将各种模式的内存配置情况做了一个简明扼要的对比。

表 1-1 TURBO C 最大内存占用

模 式	代 码	数 据		
		静态数据	堆 栈	总空间
微 型	64K			
小 型	64K	64K		
中 型	1M	64K		
紧凑型	64K	64K	64K	1M
大 型	1M	64K	64K	1M
巨 型	1M	>64K	64K	1M

模式越大，运行速度越低。编程时应优先选择较小的模式。特别是大型和巨型模式的运行速度下降更为明显，如非确有必要，一般不要选择这两种模式。

1.3 数据类型

C 语言有五种基本数据类型：

char，即字符型，用于存储 ASCII 字符，长度为 1 个字节。

int，即整型，用于存储整数，长度为 2 个字节。

float，即实型，用于存储实数，长度为 4 个字节。

double，即双精度实型，也用于存储实数，但长度为 8 个字节。

void，即无值形，长度为 0。

无值型 void 是一种特殊类型。用 void 来说明函数时表明该函数不返回任何值，用来说明指针时表明该指针可以和任何类型的指针相容。

此外，TURBO C 还提供了四种数据类型说明符，使数据类型大为丰富。这些说明符分别是有符号的 signed、无符号的 unsigned、长的 long 和短的 short。表 1-2 给出了与说明符结合后 TURBO C 中所有的数据类型。

表 1-2 TURBO C 数据类型

类 型	名 称	长度，字节	值 域
char	字符型	1	−128 至 127
unsigned char	无符号字符型	1	0 至 255
signed char	有符号字符形	1	−128 至 127
int	整型	2	−32768 至 32767
unsigned int	无符号整型	2	0 至 65535
signed int	有符号整型	2	−32768 至 32767
short int	短整型	2	−32768 至 32767
unsigned short int	无符号短整型	2	0 至 65535
signed short int	有符号短整型	2	−32768 至 32767
long int	长整型	4	−2147483648 至 2147483647
unsigned long int	无符号长整型	4	0 至 4294967295
signed long int	有符号长整型	4	−2147483648 至 2147483647
float	实型	4	3.4E−38 至 3.4E+38
double	双精度实型	8	1.7E−308 至 1.7E+308
long double	长双精度实型	8	1.7E−308 至 1.7E+308
void	无值型	0	无

事实上，表中各类型的说明符并不都是有用的。某些说明符 TURBO C 虽然也能接受，但加上该说明符并不起任何作用，因此可以省略。这些说明符有：

说明字符型数据的 signed

说明整型、短整型和长整型的 signed

说明整型、无符号整型和有符号整型的 short

说明双精度实型的 long

除去上面这些重复定义的类型后，TURBO C 实际有 9 种数据类型：char、unsigned char、int、unsignedint、longint、unsignedlongint、float、double 和 void。

有修饰符的整型可将 int 省略：

```
unsigned=unsigned int
long=long int
unsigned long=unsigned long int
```

此外，用户还可根据需要使用 typedef 自行定义数据类型。例如，下例定义了一个数据类型 A：

```
typedef unsigned A;
```

以后，如果用 A 来说明一个变量时，该变量就成为 A 类型，也就是 unsigned 类型的变量。应当注意的是，TURBO C 与大多数其他语言不同，它是区分大小写字母的。例如，在 TURBO C 中 A 与 a 并不等同，他们可同时使用，并代表两个不同的变量。

1.4 标识符与变量

标识符是变量、函数及标号的名称，它们由 1 至 32 个字符组成。其中第一个字符必须是英文字母或下划线，其余可以是字母、数字或下划线。此外还应注意不要将关键字用于标识符。例如 Time_of_Reduction、Tg、_buffer、T1、P2、V3 等都是合法的标识符。

TURBO C 要求所有的变量在使用前都要加以说明。说明的主要目的是为变量确定类型，以便为其分配内存。含有未说明变量的程序无法通过编译。变量说明的一般格式为：

类型　变量标识符 1，变量标识符 2，……，变量标识符 n；

变量说明可在程序内的三个地方进行，但位置必须在第一次使用这个变量的语句之前。

第一个地方在所有函数的外部，其中包括主函数 main()。这样的变量称为全局变量，对整个程序有效。例如：

```
float T=273.15,V,P,t,v,m;
double Hsum,Msum;
main()
  {... ...}
F1()
  {... ...}
F2()
  {... ...}
.....
```

第二个地方在函数内部。这样的变量称为局部变量，仅对该函数有效。说明格式与全局变量相同，区别仅在位置。下例说明了 main 函数的局部变量：

```
main()
  {
    int i=1,j=1;
    double Mvalue;
    char *str;
    ……
  }
```

第三个地方在函数名后面的括号内。这样的变量称为形式参数(简称形参)，仅对该函数有效，作用是接收调用者传递过来的参数值。例如：

```
……
int Tcalc(int n1,int n2,float V,float P)
  {… …}
……
```

在说明变量的同时允许为其赋初值。这样的变量称为有初值变量或初始化变量，例如前例中的 T、i 和 j。在说明时未经赋值的变量称为无初值变量或非初始化变量，例如前式中的 Hsum 和 Mvalue。应当注意，各函数所使用的形式参数和局部变量可以重复，但不能和全局变量重复。

1.5 运算符

1.5.1 算术运算符

TURBO C 支持七个算术运算符：

＋　加，含义同其他高级语言。

－　减或取负，含义同其他高级语言。

*　乘，含义同其他高级语言。

/　除，含义同其他高级语言。用于整型或字符型变量时，余数将被截掉。

％　取模，即进行整数除法运算后，将余数作为结果。例如，10％3＝1。

＋＋加 1，x＋＋或＋＋x 相当于 x＝x＋1。

－－减 1，x－－或－－x 相当于 x＝x－1。

如上所示，＋＋和－－的不同位置并不影响 x 的值。但在稍复杂的表达式中，这种位置的差别是有影响的。例如：

y＝＋＋x；相当于 x＝x＋1；y＝x

y＝x＋＋；相当于 y＝x；x＝x＋1

1.5.2 关系运算符和逻辑运算符

TURBO C 支持六个关系运算符和三个逻辑运算符。

关系运算符为：

＞　大于，左方的值大于右方的值为真，否则为假。

＜　小于，左方的值小于右方的值为真，否则为假。

>= 大于等于,左方的值大于或等于右方的值为真,否则为假。

<= 小于等于,左方的值小于或等于右方的值为真,否则为假。

== 等于,左方的值等于右方的值为真,否则为假。

! = 不等于,左方的值不等于右方的值为真,否则为假。

逻辑运算符为:

&& 与,左右双方均为真时结果为真,否则为假。

|| 或,左右双方只要有一方为真时,结果即为真。双方均为假时,结果才为假。

! 非,表达式为真时结果为假。表达式为假时结果为真。

运用这些运算符可组成简单的或复杂的表达式。如果表达式正确,则返回真(true),即一个不为 0 的数。如果表达式不正确,则返回假(false),即 0。

这样的表达式常用于程序的分支结构,根据表达式的真假决定程序的不同进程。TURBO C 的 if、if else、while、do while 和 for 语句中经常可以见到关系运算符和逻辑运算符。

1.5.3 按位运算符

TURBO C 支持六个按位运算符:

& 按位与,按位进行与运算。

| 按位或,按位进行或运算。

^ 按位异或,两操作位都为真或都为假时,结果位为假。否则结果位为真。

~ 按位非,按位进行非运算。

>> 右移,将变量的每一位都向右移动指定的位数。

<< 左移,将变量的每一位都向左移动指定的位数。

按位运算符是 TURBO C 完成汇编语言功能的有力工具。当需要与硬件通讯时,常使用这些运算符处理低层的位操作。

位逻辑运算是按位进行的,因此 &、|、^ 和~的运算结果对每一位来讲是 1 或 0,但总结果却可能是任何数。

>>和<<运算表达式中应同时给出操作数和需要移动的位数。每左移一位相当于乘以 2,但最左一位会丢失。右移一位相当于除以 2,但最右一位会丢失。一般格式为:

变量名>>移动位数;

变量名<<移动位数;

1.5.4 其他运算符

TURBO C 的? 运算符可完成判断和使程序分支的功能。一般格式是:

表达式 1? 表达式 2:表达式 3;例如:y=n>=5? 0:1;

上例中,y 值取决于 n 值。当 n>=5 为真时,y=0,否则 y=1。这相当于:

```
if(n>=5)y=0;else y=1;
```

& 和 * 是 TURBO C 一对互补的运算符。& 是取某变量的地址,* 是取某地址的变量。这两个运算符的符号与按位与和乘号相同,但功能完全不同。例如,我们有一个变量 p,它的地址是 2000,值为 3.14。

```
A=&p;
```

这个表达式将 p 的地址赋于 A,这时 A 的值为 2000。