计算机系列教材

C语言程序设计教程

高　枚　杨志强　许丽华　编著

同济大学出版社

图书在版编目(CIP)数据

C 语言程序设计教程 / 高枚等编著. —上海：同济大学出版社，2001.1
ISBN 7-5608-2241-X

Ⅰ.C... Ⅱ.高... Ⅲ.C 语言-程序设计-教材
Ⅳ.TP312

中国版本图书馆 CIP 数据核字(2000)第 85396 号

C 语言程序设计教程
作　者　高　枚　杨志强　许丽华
责任编辑　张智中　责任校对　徐春莲　装帧设计　潘向蓁

出　版
发　行　同济大学出版社
(上海四平路 1239 号　邮编 200092　电话 021-65985622)
经　销　全国各地新华书店
印　刷　崇明晨光印刷厂印刷
开　本　787mm×1092mm　1/16
印　张　16.75
字　数　428 800
版　次　2001 年 1 月第 1 版　2001 年 1 月第 1 次印刷
书　号　ISBN 7-5608-2241-X/TP・240
定　价　23.50 元

前　言

C 语言是一种模块化、结构化的通用程序设计语言，它同时还兼具了汇编语言对系统资源的操作能力及执行效率高的优点，既可以用来开发系统软件，又可以进行应用软件的设计。应用范围涵盖了操作系统、图像处理、数据库、网络、通讯等诸多领域。

Turbo C 是 Borland 公司推出的应用最为广泛的 C 语言版本，有着良好的开发环境。本书就是以 Turbo C2.0 为背景编写的，书中较全面地介绍了 C 语言的基本语法知识，内容主要包括基本数据类型、构造数据类型和指针类型的特点和使用；控制结构、函数结构的程序设计方法；文件的操作及 I/O 函数等等。

我们在多年的 C 语言教学实践中，对学生学习过程中遇到的困难有着较为深刻的体会，因而我们希望在本书中能够融入这些体会。内容上力求叙述准确，重点突出，选择例题尽量做到针对性强，为便于理解，在给出程序之前对问题进行尽可能准确的必要分析。对难点问题和易出错的地方通过许多思考题举一反三，反复加深印象。另外，每章后面都给出大量类型多样的习题，希望从不同角度、不同侧面去培养读者的程序设计能力。如果通过学习本书不仅可以掌握 C 语言本身的特点和用 C 语言设计程序的方法，而且还在某种程度上学会一些用计算机处理问题的思维方法，那便是我们的初衷。本书的编写过程中，郑志毅老师给予了许多的指导和帮助，在此深表感谢。

书中的例题程序已在 PC 机上 Turbo C2.0 环境下运行通过，若在其他计算机系统上使用这些程序，可能会有某些差别，稍加修改即可运行。

本书可作为高校非计算机专业程序设计课程教材，也可以作为广大学习者的自学参考书。为便于学生自学和教师备课，我们还编写了配套的《C 程序设计上机实验与解题指导》，即将由同济大学出版社出版。

由于水平有限，时间仓促，书中错误在所难免，恳请专家读者批评指正。

编者

2001 年 1 月

第一章　计算机语言及C语言概述

1.1　计算机语言

人们日常生活和工作的方方面面都离不开各种各样的程序，广义上讲，程序可以被理解成以时间为进程，为实现某一特定目标而进行的一系列活动的集合，而设计和编制这些程序的过程就是程序设计。我们通常提到的程序设计的对象都是特指计算机的，要实现让计算机替我们解题的目的就要使用计算机能够懂得的语言，这便是计算机语言。

1.1.1　计算机语言的发展过程

计算机语言是程序设计的最重要的工具，它是指计算机能够接受和处理的、具有一定格式的语言。从计算机诞生至今，计算机语言已经发展到了第四代。

机器语言是第一代计算机语言，它是由0和1两个代码组成的、能被机器直接理解、执行的指令集合。这种语言编程质量高，所占空间少，执行速度快，是机器唯一能够执行的语言。但机器语言不易学习和修改，且不同类型机器的机器语言不同，只适合专业人员使用。正因为如此，现在已经没有人用机器语言直接编程了。

第二代计算机语言是汇编语言，它采用一定的助记符来代替机器语言中的指令和数据，又称为符号语言。汇编语言一定程度上克服了机器语言难读难改的缺点，同时保持了其编程质量高，占用存储空间少，执行速度快的优点。故在程序设计中，对实时性要求较高的地方，如过程控制等，仍经常采用汇编语言。该语言也依赖于机器，不同的计算机一般也有着不同的汇编语言。

汇编语言再向自然语言方向靠近，便发展到了高级语言阶段，这就是第三代计算机语言。用高级语言编写的程序易学、易读、易修改，通用性好，不依赖于机器。但机器不能对用其编制的程序直接运行，必须经过语言处理程序的翻译后才可以被机器接受。高级语言的种类繁多，如面向过程的FORTRAN，PASCAL，C等，面向对象的C++，Java，Visual Basic等等。

第四代计算机语言是面向问题的语言，它是一种非过程化的语言。使用这种语言设计程序时，用户不必给出解题过程的描述，仅需要向计算机提出所要解决的问题即可。

无论是第二代的汇编语言、第三代的高级语言还是第四代的面向问题的语言，用它们编制的源程序都不能在计算机上直接运行，而需要借助于语言处理程序加工成目标程序后，才能够被机器执行。

1.1.2　语言处理程序

在所有的程序设计语言中，除了用机器语言编制的程序能够被计算机直接理解和执行外，用其他程序设计语言编写的程序都必须经过一个翻译过程才能转换为计算机所能识别的机器语言程序，实现这个翻译过程的工具便是语言处理程序。针对不同的程序设计语言编

写出的程序，语言处理程序也有不同的形式。

汇编程序是将汇编语言编制的程序（源程序）翻译加工成机器语言程序（目标程序）的工具。而编译程序则是将高级语言编写的程序（源程序）翻译成目标程序的工具。从高级语言程序到获得运行结果的一般过程见图 1.1。大部分高级语言都是采用编译程序进行翻译的，C 语言便是其中之一。还有一些高级语言则是采用另外一种翻译程序——解释程序进行处理的。解释程序直接对源代码中的语句进行解释执行，产生运行结果，但不产生目标代码。其优点是易于实现人机对话，能及时帮助用户发现错误和改正错误；但其效率低，耗时较多，如 BASIC 就是采用解释程序进行处理的。

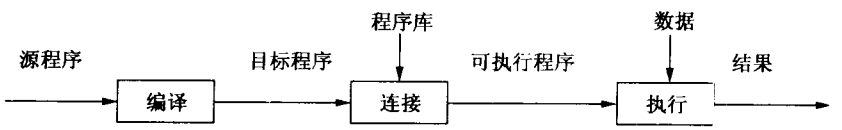

图 1.1　高级语言开发程序过程示意图

1.2　C 语言概述

C 语言出现之前的高级语言都存在一个共同的缺点，即它们无法对计算机硬件直接进行操作，这使得编制操作系统等系统程序的任务主要由汇编语言来实现，而汇编语言又存在着无法克服的可读性、可移植性差的问题，这在一定程度上限制了系统软件的开发。于是研制一种能够直接对硬件进行操作的新型的高级语言成为人们的迫切需要，C 语言就是在这样的背景下逐步开发出来的。它既具有汇编语言高效和对硬件进行操作的功能，又具有其他高级语言灵活、通用性好的优点，是目前最为流行的一种面向过程的语言。

1.2.1　C 语言的发展历史

C 语言是 1972 年由 D. M. Ritchie 在贝尔实验室的 PDP/11 机的 UNIX 操作系统上研制成功的，其原型是 ALGOL60 程序设计语言。1963 年伦敦和剑桥大学在 ALGOL60 基础上推出了 CPL 语言，它接近硬件一些，但规模大，难以实现。1967 年剑桥大学对 CPL 语言进行简化，推出了 BCPL 语言，但是这种语言只具有单一的数据类型，处理能力可想而知。以此为基础，贝尔实验室又于 1970 年研制出了简单且接近硬件的 B 语言，但 B 语言过于简单且功能有限。1972 年贝尔实验室的 D. M. Ritchie 以 B 语言为基础，引进多种数据类型，在 PDP/11 机上实现了 C 语言，并于次年用 C 语言重写了 UNIX 操作系统。从此 C 语言被用来代替汇编语言设计系统程序，大大促进了系统软件的开发。

1.2.2　C 语言的特点

（1）C 语言是一种理想的结构化程序设计语言，提供了三种形式的独立程序模块，即顺序模块、分支模块和循环模块，以保证其良好的结构化程序设计风格。

（2）程序具有较好的通用性和可移植性，基本不依赖于机器。在一个环境下运行的程序不加修改或稍加修改便可以在其他不同的环境下运行。

（3）生成的目标代码质量高，程序执行效率高，一般只比汇编程序生成的目标代码效率

低 10%～20%。

(4) 直接访问物理地址,能进行位操作,可直接对硬件进行操作。

(5) 表达能力强,书写形式简练。有丰富的数据类型和运算符,并提供了功能强大的库函数。

(6) 语法限制少,设计自由度大。这在一定程度上使C语言对编程者的要求高于其他高级语言。

C语言的这些特点,可以使我们方便地设计出灵活、高效的程序,另一方面在学习和编程过程中,也需要编程者抱着认真严谨的态度,尽量避免因C语言语法限制少而给程序排错带来的困难。

1.2.3 C程序举例

先来看几个用C语言编制的简单的程序示例,从中总结出C程序的结构要求。

【例 1.1】

```
#include  "stdio.h"
main(  )
{
printf("How are you! \n");
}
```

这是一个用C语言编写的最简单的源程序,该程序运行的结果是在屏幕上输出如下一行信息:

```
How are you!
```

C语言的程序是由函数组成的,任何一个程序都必须包含一个 main 主函数,也可以包含若干个其他函数。该程序只有一条语句,即 printf 函数调用语句,C语言中每条语句都以分号结尾。printf 是一个输出函数,是由系统提供的库函数。一般如果在程序中要调用某个库函数时,应在程序开始的地方用 include 命令将包括该函数说明的头文件作一说明,该程序的第一行就是这样一条编译预处理命令。对于库函数 printf,通常可以省去这条编译预处理命令,故该程序中也可将第一行去掉。一对花括号标志主函数 main 的开始和结束。

【例 1.2】 求半径为 r 的圆的面积。

```
#define  PI  3.14                    /* 定义一个符号常量 PI 为 3.14 */
main(  )
{
float  r, s;                         /* 定义变量 r 和 s */
printf("Input  r:\n");               /* 输出一条提示信息 */
scanf("%f", &r);                     /* 输入半径 */
s=PI*r*r;
printf("s=%f", s);                   /* 输出面积 */
}
```

该程序在 main 主函数中定义了两个实型变量 r 和 s，它们分别代表半径和面积，在 C 语言中，一般对变量的定义都应位于执行语句的前面，变量定义也要用分号结尾，相同类型的变量可放在一起定义，中间用逗号分隔。程序还调用了输入库函数 scanf，该函数可以接受从键盘读入的数据，并将数据转换后存放在相应的变量中，调用该函数时也可以省去编译预处理命令。程序中通过一条赋值语句(第 7 行)将计算出的圆的面积赋给变量 s。程序中以 /* …… */形式出现的部分是对程序的解释，该部分不属于可执行部分，只是为了增加程序可读性而书写的对某语句或某段程序代码的说明和解释。程序第一行以 #define 开头的代码，是一个宏定义命令，它是 C 语言的编译预处理命令之一。

程序的运行情况如下：

```
Input  r:
2  ↵
s==12.560000
```

其中，第二行数据“2”是用户输入的，↵代表回车键，其余的信息由机器自动输出。

【例 1.3】 求两个数的和。

```
main( )
{
 int  a, b, c;
 scanf("%d  %d", &a, &b);
 c=add(a, b);
 printf("sum = %d\n", c);
}
int  add(int  x, int  y)
{
 int  z;
 z=x+y;
 return  z;
}
```

程序执行情况如下：

```
12  13  ↵
sum=25
```

该程序由两个函数组成：主函数 main 和函数 add。从形式上看，它们是并列的关系，而不是嵌套的关系。在主函数的第四行是一个函数调用语句，当程序执行到该语句时，便转向执行被调用函数 add。add 函数是个用户自定义的函数，其功能是求两个数的和，它先完成实在参数 a，b 对形式参数 x，y 的值传递，然后执行函数 add 中的各个语句，当遇到 return 语句或函数结束后，add 函数又将控制权转给 main。带表达式的 return 语句将带回一个返回值给主调函数。一般使用一个自定义函数要从这样几方面入手：

(1) 函数的定义：用以描述函数的功能，位于主调函数的外部。

(2) 函数的调用:用在主调函数中,指出如何使用该函数。

对任何一个自定义的函数,上述两方面都是不可缺少的。另外,对于有些形式的函数,还需要在主调函数中加上对该自定义函数的说明,这在函数一章中将详细介绍。

通过对上面三个程序的讨论,我们将 C 语言源程序的构成要素、含义及书写形式归纳如下:

(1) 一个程序至少要包含一个主函数 main,并且无论这个函数的位置在哪里,程序都从它开始执行。

(2) 除了主函数之外,某些程序中还可能包含若干个其他的函数,包括系统提供的库函数和用户自定义的函数。在使用自定义函数时,要从函数的定义、调用和说明三部分加以考虑。

(3) 函数的定义用一对花括号作为开始和结尾的标志,包括说明和执行两部分:说明部分用来说明函数和定义变量,而执行部分则通过语句描述函数的功能。变量定义和语句都要以分号结尾。

(4) 程序的任何地方都可以加入以"/ * "和" * /"包围起来的解释部分,该部分的作用是为了增加程序的可读性,它不是程序的可执行部分,不产生程序代码。

(5) C 语言书写形式自由,一行内可以写多条语句,一条语句也可以分写在不同行上,当一条语句分写在几行上时,行尾需加续行符"\"。C 语言中大小写字母是有区别的,这一点在使用时要特别注意。

1.2.4 C 语言的基本语法单位

C 语言作为一种程序设计语言,也同自然语言一样,具有它自身的词汇和语法规则。编写程序时不能使用这些词汇以外的内容。

1. C 语言的基本符号

(1) 大小写英文字母各 26 个:a～z,A～Z;

(2) 0～9 共十个数字;

(3) 下划线_;

(4) 特殊符号:包括运算符在内的其他字符,暂不一一例举,详见运算符部分。

2. 标识符

标识符就是名字,它是一个字符序列,可以用来标识变量名、符号常量名、函数名、类型名、文件名等。在定义标识符的时候,C 语言对组成标识符的字符序列有一定的规定:首先,标识符只能由字母、数字和下划线三种字符组成,且必须以字母或下划线开头;其次,虽然 C 语言没有对标识符的长度进行统一的规定,但是具体的系统都有长度限制。有些系统只能识别前 8 个字符,对于这些系统,标识符 student1 和 student11 被认为是同一个标识符。Turbo C 能识别长达 31 个字符。C 语言中,标识符有三种类型:

(1) 关键字:系统已定义过的、有特定含义、不能挪作它用的标识符。C 语言共有 32 个关键字,例如,int,char,break,for 等。所有的关键字详见表 1.1。

(2) 预定义标识符:系统定义过的,用于编译预处理命令中的标识符,如 include,endif,define 等。建议这些标识符也不要作为用户自定义标识符。一旦用户自定义标识符与这些标识符同名,系统预定义的相应功能便会丢失。

(3) 自定义标识符：依据标识符的命名原则，由用户自己定义的标识符。其中自定义标识符不能与关键字重名，最好也不要和预定义标识符重名。除此之外，定义标识符时，最好能简洁且“见名知意”，以提高程序的可读性。

例如，下面的标识符都是合法的：

A2　　student　　area_of_circle　　num　　_dd

下面的标识符则不合法：

2A　　A－B　　area of circle(一个标识符)　　M.D

表 1.1　　C 语言中的关键字

auto	break	case	char	const
continue	default	do	double	else
enum	extern	float	for	goto
if	int	long	register	return
short	signed	sizeof	static	struct
switch	typedef	union	unsigned	void
volatile	while			

1.3　C 语言程序的上机步骤

从将一个编写好的 C 语言源程序输入计算机到获得程序的最终结果，要经过编辑、编译、连接、运行等步骤，以下我们在 Turbo C 集成开发环境下，对上述步骤的实现方法做一个简单的介绍。

1. 进入 Turbo C 集成环境

运行 Turbo C 的执行程序 TC.EXE，就进入了 Turbo C 集成环境，出现如图 1.2 所示的 TC 主屏。从图中可以看出，TC 主屏由四部分组成：主菜单、编辑窗口、信息窗口和功能键提示行。主菜单上的每个命令又都包含一个下拉子菜单(Edit 除外)，编辑窗口是用来装载或输入源程序的，而信息窗口则用于察看编译和调试程序的诊断信息。

在进行编辑程序之前应设置好包含文件、库文件、输出文件和 Turbo C 系统所在的目录。选择 Options 菜单中的 Directories 命令，弹出如图 1.3 所示的子窗口。若上述四类文件所在的目录分别为：C:\TC\INCLUDE，C:\TC\LIB，C:\TC 和 C:\TC(Turbo C 系统在 C:\TC 目录中)，则依次在“Include directories:”，“Library directories:”，“Output directory:”和“Turbo C directory:”后把它们键入。

2. 编辑

在编辑窗口中将编写好的程序输入计算机或对已经存在的源程序文件进行修改的过程，就是源程序的编辑过程。首先，选择 File/Load(表示 File 菜单中的 Load 命令，以下的表示法有类似的含义，不再解释)命令，输入一个要编辑的源程序名字，则编辑窗口中的 NONAME.C 被新输入的名字取代。如果被编辑的是一个新文件，就可以将编写好的程序

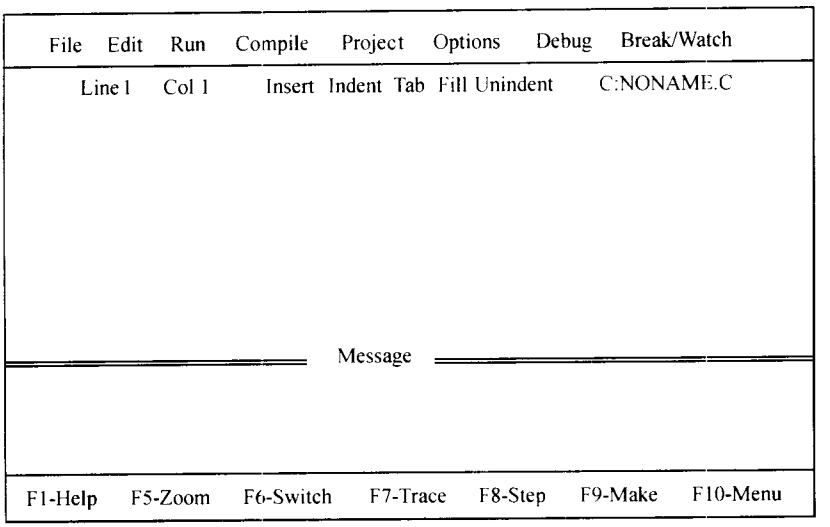

图 1.2 Turbo C 主屏

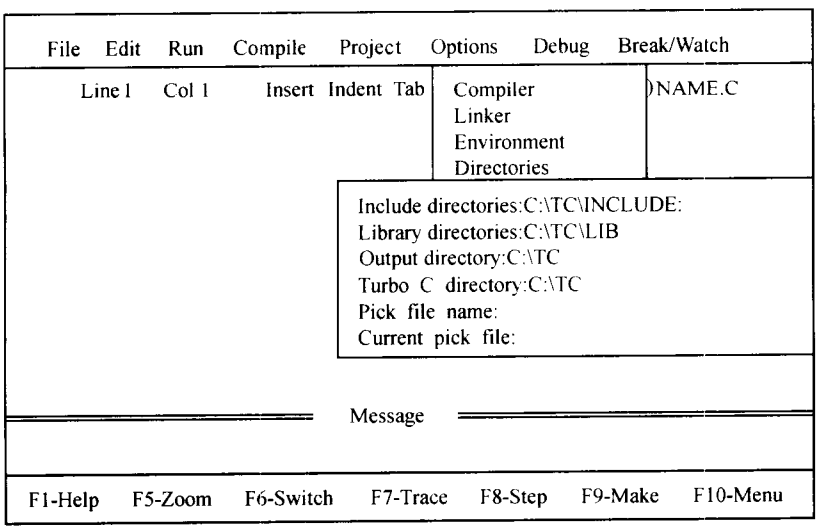

图 1.3 Turbo C 目录设置子窗口

输入到编辑窗口了；如果被编辑的是一个旧文件，这时则可以对其进行修改，输入或修改结束后，选择 File/Save 命令或按功能键 F2 保存文件。

3. 编译

编辑好的源程序是不能被机器识别的，必须要经过一个语言处理程序将其翻译成机器代码后才能被执行，在 C 语言中这个过程被称为编译，简而言之，编译是将源程序翻译成机器能识别的目标程序的过程。选择 Compile/Compile to OBJ 命令，对编辑窗口中的文件进行编译，编译的结果是在弹出的一个窗口中提示编译成功或发现的错误信息，按任意一个键结束本次编译。如果编译成功，按 F2 键，将文件保存一下后就可进入下一个过程；否则，若编译失败，则在信息窗口中将发现的错误和警告信息逐一列出。这时，需要回到编辑窗口对

源程序进行修改，之后，再重新编译，这一过程可能要重复若干次，直到编译成功为止。光标在信息窗口和编辑窗口间进行切换的热键是 F6。

4. 连接

这是一个将编译后生成的目标文件和库文件连接在一起生成可执行文件的过程。当编译成功后，选择 Compile/Link EXE file 命令，进入连接；如果出现错误，则一般要检查库文件和头文件的路径设置是否正确，修改后重新连接。

5. 运行

选择 Run/Run 命令或按热键 Ctrl+F9(两个键同时按下)，就可以执行程序。若不需要输入数据，则屏幕闪烁一下后，马上又回到集成环境的主屏。如果程序在运行过程中需要输入数据，则选择了 Run/Run 命令后，屏幕处于用户屏状态，等待用户输入所需的数据，数据输入结束，运行后重新回到集成环境的主屏。若要察看运行结果，则选择 Run/User Screen 命令或按下热键 Alt+F5，屏幕便切换到用户屏状态了。如果程序运行结果不正确，需要再回到编辑窗口修改源程序。

Turbo C 中，编译、连接和运行也可以通过选择 Run/Run 命令或按热键 Ctrl+F9 而一并完成。

6. 退出 Turbo C 环境

选择 File/OS shell 命令，可以暂时退出 Turbo C，转到 DOS 提示符下，此时，若再键入 EXIT，则又可以返回到 Turbo C 集成环境的主屏。当需要运行 DOS 命令而又不想退出 Turbo C 环境时，可以使用这种退出方式。

选择 File/Quit 命令，彻底退出 Turbo C，返回到 DOS 提示符下。之后若再想进入 Turbo C 环境，则需要重新运行可执行文件 TC.EXE。

7. 调试

一个程序编译成功，并不一定会得到正确的结果，因为编译程序只能将绝大部分语法错误提示出来，而程序中的错误除了编码上的，还有设计上的。只有将所有错误排除，程序才能得以正确运行。我们将发现和改正错误的过程称为程序的调试。如果不使用 Turbo C 开发环境提供给我们的工具，最直接的查找错误的方法就是在程序中的适当位置加入一些输出中间结果的语句，根据这些结果的正确与否便可以判断此输出语句之前是否存在错误。但 Turbo C 开发环境提供了许多便捷有效的工具来帮助我们发现错误。在此，仅介绍几个简单常用的工具供大家参考，至于想知道更多的内容，请参阅有关的书籍。

使用 Break/watch 菜单下的 Add watch 命令，将弹出一个 Add watch 窗口。在该窗口中，输入一个待观察的变量或表达式，若输入的内容有效，按回车键之后，就增加一个变量或表达式到监视窗口，并显示其当前值，见图 1.4。用这一方法能够动态地观察到重要数据的变化情况。

选择 Run/Go to cursor 命令，或按热键 F4 可以使程序运行到光标所在位置。若光标所在行不含可执行代码，则给出警告信息，提示按 ESC 键返回。使用该命令可以分段运行程序，通过观察监视窗口中变量或表达式的变化情况，判断错误所在的程序段。

选择 Run/Trace Into 命令，或按热键 F7，可以单步运行程序，即逐条地执行语句。该命令还可以跟踪到函数内部去单步运行。

选择 Run/Step Over 命令，或按热键 F8，也可以单步运行程序，但该命令不能跟踪到函

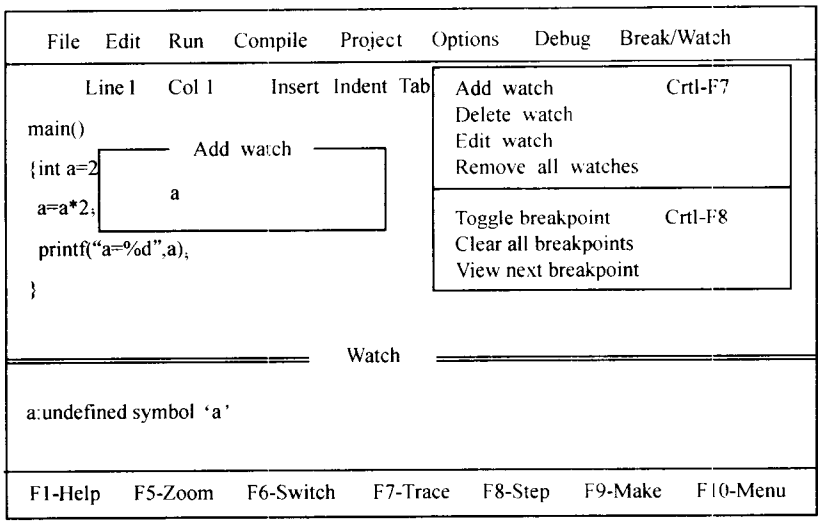

图 1.4 监视窗口设置子窗口

数内部去逐句执行。

无论选择了上述 Run 命令中的哪一种运行方式，都会在执行到的位置处出现一个高亮度显示条，若想消除高亮度显示条，选择命令 Run/Program reset 或按热键 Ctrl+F2。该命令释放用户程序占用的内存空间，终止程序调试。

习　题

1. 试述源程序、目标程序、编译程序、可执行程序的区别。
2. C 语言程序由哪几部分组成？
3. 判断下列符号中哪些不能作为 C 语言的标识符：
   abc，a ?b，3A，a－b，f(x)，A D，a. txt，D $，s _ 2，π
4. C 语言标识符有几类？各自的用途是什么？
5. 编写一个求梯形面积的程序。
   已知：上底 a＝5，下底 b＝8，高 h＝4。
6. 已知一个学生三门课的成绩分别为 65，80，79，编写一个程序求该学生的平均成绩。

第二章　基本数据类型、运算符和表达式

数据是程序操作的对象，对数据的说明是程序设计的前提和基础，一种程序设计语言中提供的数据类型的多少，也在一定程度上反映了其处理能力的强弱。C 语言不仅可以处理数值型数据，同时，也可以处理非数值型数据，其数据类型相当丰富。C 语言的数据类型可大致作如下分类：

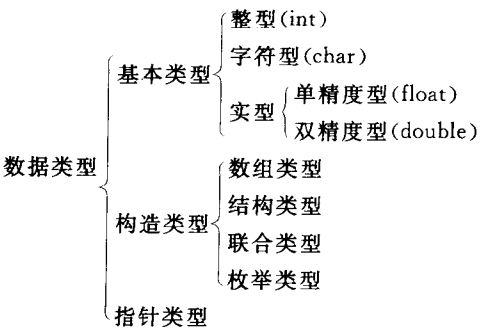

图 2.1　数据类型

其中，括号中的标识符为标识相应数据类型的关键字。基本类型又称为简单的数据类型，是不能再被细分的。而构造类型是由基本数据类型或构造类型按不同的方式组合成的复杂数据类型，指针类型是 C 语言中的一种很有特色的类型，构造类型和指针类型在后续章节中将陆续介绍。枚举类型是 ANSI C 中新增加的一种数据类型，它将该类型的所有取值逐一列出，本书略去不作介绍，有兴趣者请自行参考相关资料。

2.1　基本数据类型

C 语言提供了三种基本数据类型：整型、字符型和实型数据，其中每一种类型的数据又有常量和变量之分，下面将分别予以介绍。

2.1.1　常量

1. 值常量和符号常量

常量是指那些在程序运行过程中其值不能改变的量。C 语言的常量包括直接以具体值的形式出现的值常量和以符号标识的符号常量。如：

−25，3.0，2e−3，'A'　等都是值常量。

符号常量是用 define 命令定义的常量。如：

```
#define   PI   3.14
```

我们称标识符 PI 为符号常量，其值为 3.14，在程序中可以使用 PI 来代替常量 3.14。

define 是宏定义，它是一条编译预处理命令，更多的内容在后续章节中会作详细介绍，在此仅给出利用它定义符号常量的方法：

```
#define  标识符  常量值
```

这样定义过的标识符便可以代替常量值出现在程序中，使程序易读易改。

【例 2.1】 符号常量示意程序

```
#define  PI  3.14
main( )
{
float  r=3.0, s, l;
l=2 * PI * r;
s=PI * r * r;
printf("l=%f, s=%f", l, s);
}
```

该程序已知圆的半径，求圆的周长和面积。程序中两次使用了符号常量 PI 来代替 3.14，程序执行时 PI 将被 3.14 代替后参与运算。

当某个常量值在程序中被多次使用时，用符号常量来代替值常量，可以使程序更加简洁明了，并方便程序的修改。比如在上面的程序中，为了提高精度，需取圆周率的值为 3.1416，若使用符号常量，则只需要修改 define 中的常量值，程序的其他部分皆不需要改变，否则，就要到程序体中，将圆周率的值逐一改变。

在使用符号常量时要注意，虽然它是用标识符来标识的，但它本质上是常量，具有常量值不能改变的性质。例如，假设在一个程序中有如下两个定义：

```
#define  PI  3.14
#define  PI  3.1416
```

则是错误的。因此在一个程序中不能对同一个符号常量定义两次。

在程序体中也不能出现对符号常量的赋值语句。例如，下面是一个错误的赋值语句。

```
PI=3.1416;(假设此前已定义过 PI 为一符号常量)
```

另外，为了在形式上更明显地与变量进行区别，习惯上符号常量名用大写表示，而变量名则用小写表示。

以上介绍了常量的两种形式，即值常量和符号常量。下面就几种基本数据类型的常量逐一进行讨论。

2. 整型常量

同数学中整数是个无限集的概念不同，计算机中的整型数是有范围的，不同的机器系统，整型数据所占的字节数可能不同。但一般而言，用一个机器字来存放一个基本整型数据。考虑到特殊需要，除了基本整型之外，C 语言还提供了长整型(long)和短整型(short)的形式。一般来讲，短整型数据所占的字节数不应大于基本整型数据，而长整型数据所占的字节数不应小于基本整型数据。PC 机上，long 为 4 字节，short 为 2 字节。另外，根据最高位的不同含义，又可以将整型数据分为有符号整型和无符号整型(unsigned)。表 2.1 列出了 PC 机

上各类整型数据的数值范围。

在表示方法上，通常在整数后面加上字母“l”或“L”，表示它为一个长整型数据；若在某个整数后面加上字母“u”或“U”，则表示它是一个无符号整数。例如，125，26，40000L，37U，0L 等等，都是合法的整型常量。而 50000，－23u 则不是 C 语言中合法的整数。前者超出整数范围，而后者却在无符号数的前面加了负号“－”。

表 2.1　　PC 机上各类整型数据的数值范围

类　型	所占位数	数的范围
int	16	－32768～32767　　即 $-2^{15}\sim(2^{15}-1)$
short [int]	16	－32768～32767　　即 $-2^{15}\sim(2^{15}-1)$
long [int]	32	－2147483648～2147483647　　即 $-2^{31}\sim(2^{31}-1)$
unsigned [int]	16	0～65535　　即 $0\sim(2^{16}-1)$
unsigned short	16	0～65535　　即 $0\sim(2^{16}-1)$
unsigned long	32	0～4294967295　　即 $0\sim(2^{32}-1)$

以上讨论了 C 语言整型常量的类别，下面介绍整型常量的十进制、八进制和十六进制表示方法。

(1) 十进制形式

形式同数学中的整数，只是有个数据范围的限制。如 125，－21，0。

(2) 八进制形式

以数字 0 开头的、由 0～7 之间的数字组成的数据，如 056，－012。八进制数是以 8 为基数，每一位上的权为 8 的整数次幂。例如，$056=5*8^1+6*8^0=46$，即八进制的 56 相当于十进制的 46。

(3) 十六进制数

以 0x 或 0X 开头的，由数字 0～9 和字母 A～F 组成的数据。其中 6 个字母也可以小写，他们分别相当于十进制的 10～15，如 0x14，0X5A。十六进制数是以 16 为基数，各位上的权是 16 的整数次幂。例如：$0X5A=5*16^1+10*16^0=90$，即 16 进制的 5A 相当于十进制的 90。

整型常量的类别和进制表示可以结合在一起使用，如 0X41L 为十六进制长整型数。

3. 实型常量

同整型数据一样，机器中的实数也是有一定范围限制的，根据精度的不同，范围也有所不同。实型常量有两种表示方式：

(1) 十进制小数形式

由正负号、数字和小数点组成，其中小数点不能缺少，正数符号可省略。如 1.25，34.0，＋1.25，17. 和 .1。

(2) 指数形式

由尾数开头，加上指数部分共同构成，其中指数部分由指数符号 e(或 E)、正负号及整数组成。如 1.25e－5，＋1e10，－1.25e5，1e－6，3e2。

注意：e 前面不能没有数字，e 后面的数字必须为整数，也不能加圆括号。如 1E(－3)，E－5，1e2.1 都是错误的。

4. 字符型常量

C 语言除了可以处理数值型数据，还能够处理字符型数据。其提供的字符型常量包括字符常量和字符串常量。前者一般是用单引号引起的单个字符来表示，而后者则用双引号引起的若干个字符来表示。以下我们将着重讨论字符常量。

(1) 普通的字符常量：用单引号引起来的一个字符，如'A'、'b'、'0'、'?'等都是字符常量。对于可打印的字符都是用这种形式来表示的。

(2) 转义字符常量：它是以"\"(反斜杠)开头的用单引号引起来的字符序列。这类特殊字符常用来表示具有特定功能的非显示字符。如：'\n'代表换行符，输出时将光标移至下一行的行首；'\t'代表水平制表符，作用是输出时使光标横向跳到下一个输出区的开始位置。系统一般默认的一个"输出区"占 8 列，假设当前光标在第 1～8 列的任意一个位置上，则'\t'会使光标跳到第 9 列上，若当前光标在第 9～16 列的某列上，则'\t'会使光标跳到第 17 列上，依此类推。常用的转义字符见表 2.2。

表 2.2　　常用的转义字符

字符形式	功　能	字符形式	功　能
\n	换行	\\	反斜杠字符"\"
\t	横向跳格(即跳到下一个输出区)	\'	单引号(撇号)字符
\b	退格	\ddd	1 到 3 位 8 进制数所代表的字符
\r	回车(不换行，光标移到本行行首)	\xhh	1 到 2 位 16 进制数所代表的字符

其中，最后两种是用 ASCII 码表示一个字符，ASCII 码表见附录一。如"\x61"表示 ASCII 码为 97 的字符(x61 对应的十进制数是 97)，即字符'a'。又如，'\0'表示 ASCII 码为 0 的字符，它是一个空字符(不是空格字符)，常用在字符串中，作为字符串的结束标志。

字符型数据在内存中是以其 ASCII 码形式，而不是以其字符形式直接存放的。因此，整型数据和字符型数据可以在一定范围内通用，通过指定不同的格式，一个字符型数据既可以以字符形式输出，又可以以整数形式输出其 ASCII 码值。如数字字符'0'～'9'的 ASCII 码值是连续排列的，为 48～57；大小写字母的 ASCII 码值也是连续排列的，大写字母'A'～'Z'的值为 65～90；小写字母'a'～'z'的值为 97～122。

此外，在使用字符型数据的时候，应注意字符常量与字符串常量的区别。字符串是用双引号括起来的字符序列，如"This is a book"，"A"等都是字符串。"A"与'A'是不同的，前者是字符串，实际上由两个字符组成，即字符'A'和字符串结束标志'\0'，它也表示存放"A"的首地址，而后者是一个字符，仅由一个字符'A'组成，在存储器中存放的是'A'的 ASCII 码值。

2.1.2　变量

1. 变量的定义

相对于常量而言，变量是指在程序运行过程中其值可变的量。它有三个要素，即名字、类

型和值。每一个变量要用一个标识符来标识，称为变量名，其命名规则与标识符的命名规则相同。变量的定义方式如下：

[存储类别]　数据类型　变量名表；

其中“存储类别”是可以省略的。在此，我们先忽略存储类别，只介绍数据类型。关于存储类别，后续章节中将进行专门讨论。

例如：

```
int  a, b;        定义了两个整型变量 a 和 b。
float  d;         定义了一个单精度实型变量 d。
```

定义变量时，相同类型的变量可以放在一起同时定义，中间用逗号分隔；不同类型的变量要分别定义。每个变量定义的结尾都要以分号结束。

变量的数据类型限定了变量的取值范围和能参与的操作，不同机型上的基本数据类型所占字节数不完全相同。表 2.3 给出了 PC 机上基本数据类型的数据范围。

表 2.3　　PC 上基本数据类型的数值范围

类　型	所占字节	数值范围
char	1	−128～+127
unsigned char	1	0～255
short	2	−32768～32767
unsigned short	2	0～65535
int	2	−32768～32767
unsigned int	2	0～65535
long	4	−2147483648～2147483647
unsigned long	4	0～4294967295
float	4	−1.0e38～1.0e38(7 位精度)
double	8	−1.0e308～1.0e308(16 位精度)

2. 变量的赋值和初始化

在程序中，一个变量必须先有确定的值后才能参与各种相应的操作。变量可以通过赋值语句或输入语句获得一个值，也可以用初始化的方法获取一个值。

(1) 赋值语句

例如：

```
int  a;
a=3;
```

使整型变量 a 的值为 3。

(2) 初始化

在定义变量的同时对变量赋初值。如：

```
int  a=3;
```