

C 语言 绘 图 教 程

陆润民 编著

清 华 大 学 出 版 社

(京)新登字 158 号

内 容 简 介

《计算机绘图》和《C 语言》，是目前计算机应用类课程中两门重要的课程。作者根据多年来的教学体会，将该两门课程结合起来，编写了这本《C 语言绘图教程》。

全书介绍了图形数据结构，图形变换，绘图程序设计方法，曲线，二、三维图形的各种算法，以及 C 语言的基本内容和图形功能。并附有许多练习题及图形例程序。读者可通过本书，既学习计算机绘图，又学习 C 语言。

本书可作为大专院校开设《计算机绘图》课的教材，也可作为从事计算机辅助设计工作的工程技术人员和广大电脑爱好者的自学参考书。

版权所有，翻印必究。

本书封面贴有清华大学出版社激光防伪标签，无标签者不得销售。

书 名：C 语言绘图教程

作 者：陆润民 编著

出版者：清华大学出版社（北京清华大学校内，邮编 100084）

印刷者：北京联华印刷厂

发行者：新华书店总店北京科技发行所

开 本：787× 1092 1/16 印张：16.25 字数：401 千字

版 次：1996 年 4 月第 1 版 1996 年 4 月第 1 次印刷

书 号：ISBN 7-302-02085-X/TP·972

印 数：0001—8000

定 价：16.00 元

前 言

结合 C 语言,编写一本“计算机绘图”教材,是作者多年来的愿望,也是历届同学在选择该课程后提出的希望。现在,在清华大学出版社的大力支持下,终使好事成真。

当前,在计算机应用类课程中,有两门课长热不衰,这就是《计算机绘图》和《C 语言》。“计算机绘图”是计算机辅助设计和科学计算可视化两大领域内的核心内容,有着极其重要的地位。并且,随着科学技术的不断发展,人们对应用计算机图形的要求,必将会在广度和深度上不断提高。为了适应这种形势的发展和满足广大读者的需要,作者编写了这本《C 语言绘图教程》。该书具有以下几个特点:

内容跨度大 本书从基本的 C 程序、简单图形的绘制,直到二维图形运算和三维真实感图形的生成,内容跨度大,能使不同层次的读者均可从中获益。

适合于自学 本书的内容组织,由浅入深;讲解深入浅出。书中附有大量的 C 源程序,这些绘图程序均在 PC 机上用 Turbo C 2.0 调试通过,并绘出正确图形。适合于自学,能做到“无师自通”。也为广大电脑爱好者开辟了又一个电脑应用天地。

可一举两得 为使没有学过 C 语言的读者也能利用本书学习计算机绘图的内容,故本书以一定的篇幅系统介绍了 C 语言的基本内容,以及 C 语言中的图形功能。因此,利用本书,既可以学习计算机绘图,又学会了 C 语言。

实行作业总分制 结合教学内容,准备了充足的不同层次的练习题,供同学凭自己的条件和兴趣选做。每个习题后均标有分值,教师可根据学校的具体情况提出总分标准。该方法在作者的多年试行中,受到广泛欢迎。

在本书的编写过程中,始终得到王菱同志和陆晨同学的具体帮助和大力支持;作者历年来上千名学生中的许多同学对课程内容、教学方法、课外作业等教学环节,提出了不少富有建设性的意见,且有部分优秀作业被选入本书。在此,作者一并向他们致以真挚的谢意。

限于作者的水平,书中难免有不当之处,希望得到广大读者的批评指正。

陆润民

1995 年 8 月于清华园

目 录

第 1 章 C 语言基本知识	1	2.2 图形函数及其用法	35
1.1 C 程序的结构及特点	1	2.2.1 图形系统管理	35
1.1.1 简单的 C 程序	1	2.2.2 屏幕管理	37
1.1.2 C 程序的结构和特点	2	2.2.3 绘图函数	38
1.1.3 C 程序的基本结构	4	2.2.4 图形属性控制	43
1.2 数据类型	5	2.2.5 充填	46
1.2.1 常量与变量	5	2.2.6 图形方式下的文本	48
1.2.2 整型数据	6	2.3 绘图程序实例	49
1.2.3 实型数据	8	2.3.1 直线类图形	49
1.2.4 字符型数据	8	2.3.2 圆组成的图案	52
1.2.5 数据间的混合运算	11	2.3.3 曲线类图形	55
1.3 输出输入函数	11	2.3.4 图形文字	57
1.3.1 printf 函数	11	2.4 上机过程	58
1.3.2 putchar 函数	13	2.4.1 TC 的集成界面	58
1.3.3 scanf 函数	14	2.4.2 菜单命令	59
1.3.4 getchar 函数	16	2.4.3 编辑操作	61
1.4 运算符和表达式	17	2.4.4 快速功能键(热键)说明	63
1.4.1 算术运算符和算术表达式 ...	17	2.4.5 调试程序	64
1.4.2 赋值运算符和赋值表达式 ...	18	第 3 章 图形数据结构	66
1.4.3 关系运算符和关系表达式 ...	19	3.1 图形数据结构概述	66
1.4.4 逻辑运算符和逻辑表达式 ...	20	3.2 线性列表结构	68
1.5 控制语句	21	3.2.1 线性表	68
1.5.1 if 语句	21	3.2.2 线性表的运算	68
1.5.2 switch 语句	23	3.2.3 线性表的运算示例	69
1.5.3 while 语句	24	3.2.4 栈结构	70
1.5.4 do-while 语句	25	3.3 C 语言中的数组	70
1.5.5 for 语句	25	3.3.1 一维数组	71
1.5.6 其他控制语句	27	3.3.2 二维数组	72
第 2 章 C 程序绘图基础	29	3.3.3 字符数组	73
2.1 Turbo C 2.0 简介	29	3.4 链表结构	74
2.1.1 Turbo C 2.0 的特点	29	3.4.1 单向链表	75
2.1.2 Turbo C 2.0 的内容	30	3.4.2 循环链表	78
2.1.3 Turbo C 2.0 的配置 与安装	32	3.4.3 多重链表	78
2.1.4 Turbo C 2.0 对环境的 支持能力	33	3.5 树形结构简介	78
		3.5.1 树的基本概念	79
		3.5.2 二叉树	80

3.5.3	二叉排序树	82	5.3.1	基本图形变换法	134
3.6	排序及查找	83	5.3.2	递归法	138
3.6.1	排序的基本概念	83	5.4	典型绘图方法	139
3.6.2	排序的方法	84	5.4.1	参数法绘图	140
3.6.3	查找	86	5.4.2	子图形法绘图	143
3.7	数据文件	87	5.5	剖面线算法程序	147
3.7.1	文件概述	87	5.5.1	剖面线的特点	147
3.7.2	文件的打开与关闭	88	5.5.2	剖面线算法步骤	147
3.7.3	文件的读写	89	5.5.3	剖面线源程序	152
3.7.4	文件的定位	92			
第 4 章	图形变换	93	第 6 章	曲线	155
4.1	图形变换的基本原理	93	6.1	常见二次曲线的绘制	155
4.2	二维图形的变换	95	6.1.1	绘制曲线的基本方法	155
4.2.1	二维基本变换	95	6.1.2	椭圆绘图程序分析	156
4.2.2	二维组合变换	99	6.2	抛物样条曲线	159
4.3	三维图形的变换	102	6.2.1	曲线生成的原理	159
4.3.1	三维变换矩阵	102	6.2.2	曲线的讨论	163
4.3.2	三维基本变换	102	6.2.3	绘图程序	165
4.3.3	三维组合变换	106	6.3	三次参数样条曲线	167
4.4	三维图形的生成	109	6.3.1	曲线生成的原理	167
4.4.1	三视图	109	6.3.2	连续三次参数样条曲线的 表达式	170
4.4.2	正轴测投影图	110	6.3.3	解题过程	172
4.4.3	透视投影图	112	6.4	贝塞尔曲线和 B 样条曲线	174
4.5	视向变换	116	6.4.1	贝塞尔曲线	174
4.5.1	世界坐标系和观察坐标 系	116	6.4.2	B 样条曲线	177
4.5.2	视向变换矩阵	117	6.5	实验曲线的绘制方法	183
4.6	任意视点的透视变换	119	6.5.1	最小二乘法	183
4.6.1	透视原理	119	6.5.2	用最小二乘法拟合直线	184
4.6.2	坐标变换	120	6.5.3	用最小二乘法拟合二次以上 多项式曲线	185
4.6.3	规格化坐标	120			
第 5 章	绘图程序设计方法	122	第 7 章	二维图形的运算	188
5.1	C 语言中的函数	122	7.1	基本运算方法	188
5.1.1	函数的定义	122	7.1.1	交点计算	188
5.1.2	函数的调用	124	7.1.2	关系判别	181
5.1.3	有关函数的其它概念	127	7.2	窗口运算	194
5.2	通用绘图程序的设计方法	130	7.2.1	窗口和视图区定义	194
5.2.1	构造功能模块的基本 原则	130	7.2.2	窗口—视图区转换	195
5.2.2	正多边形绘图函数	131	7.2.3	裁剪和覆盖	196
5.3	图案程序设计方法	134	7.3	直线段的裁剪算法	197
			7.3.1	线段和窗口的关系	197
			7.3.2	代码裁剪算法	198

7.3.3	矢量裁剪算法	199	8.3.1	算法基本思想及程序 流程	217
7.3.4	中点再分裁剪算法	201	8.3.2	隐藏线的消去过程	217
7.4	多边形的裁剪算法	202	8.3.3	曲面体的消隐方法	220
7.4.1	多边形的裁剪特点	202	8.4	隐藏面算法	221
7.4.2	单边裁剪算法	202	8.4.1	深度缓冲器算法	221
7.4.3	边界裁剪算法	204	8.4.2	扫描线算法	222
7.5	多边形之间的运算	205	8.4.3	面积相关算法	224
7.5.1	多边形的覆盖	205	8.5	光照效应	225
7.5.2	多边形的布尔运算	206	8.5.1	明暗模型	225
第 8 章	三维真实感图形	210	8.5.2	多面体的明暗模型	227
8.1	概述	210	8.5.3	阴影产生	229
8.1.1	真实感图形的概念	210	第 9 章	上机作业及指导	231
8.1.2	基本计算方法	211	9.1	作业说明	231
8.1.3	描述立体的数据结构	213	9.2	习题集	231
8.2	凸面体的消隐方法	215	附录 1	常用字符与 ASCII 代码对照表	239
8.2.1	平面体表面法向量与可见 性的关系	215	附录 2	常用库函数	240
8.2.2	凸多面体隐线的消去 方法	216	附录 3	Turbo C 2.0 的图形函数	244
8.3	任意平面体的消隐算法	217	参考文献	251

第 1 章 C 语言基本知识

C 语言是目前最为流行的一种计算机高级语言。尽管它以一种高级语言的形式出现,却带有低级语言的功能(比如:允许直接访问物理地址,能进行位(bit)运算,可以直接对硬件进行操作,能实现汇编语言的大部分功能)。因此,也有把 C 语言称作中级语言的。但是,这并无贬意,完全是因为它把高级语言的成份同汇编语言的功能结合起来的缘故。

C 语言的突出优点是大家所公认的:

C 语言是结构化的语言,易于进行程序设计,且使得程序清晰明了,易于维护。

C 语言简洁紧凑,它仅有 32 个关键字(IBM PC 的 BASIC 包含有 159 个关键字)。

C 语言的程序效率高(几乎接近于汇编语言的程序),可移植性强。

用 C 语言可以实现汇编语言的功能,宜用于进行系统程序设计。

由于篇幅所限,本章只介绍 C 语言中的最基础内容。

1.1 C 程序的结构及特点

下面通过两个简单的 C 语言程序的例子,来说明 C 程序的结构及特点。

1.1.1 简单的 C 程序

[例程序 1.1]

```
/* 650597 Lurunmin exp1.c IN-OUT * /  
# include < stdio.h>  
main()  
{  
    char a;  
    printf( Input a character );  
    a= getchar();  
    putchar(a);  
}
```

该程序的功能是:请求用户输入一个字符,用户在键盘上敲入一个字符后,计算机把该输入的字符又回显在屏幕上。

程序运行的情况为:

```
Input a character    (提示)  
m                   (输入)  
m                   (输出)
```

[例程序 1.2]

```
/* 650597 Lurunmin exp2.c AREA * /
```

```

# include < math. h>
main()
{
    float a, b, c, s, area;
    printf( Input a, b, c );
    scanf( % f, % f, % f , &a, &b, &c);
    s= 1.0/2* (a+ b+ c);
    area= sqrt(s* (s- a)* (s- b)* (s- c));
    printf( area= % 7.2f ,area);
}

```

该程序的功能是：根据三角形的三条边长，计算三角形的面积。程序运行的情况是，首先在屏幕上显示一行字符 Input a, b, c，请求用户输入三角形的三条边长，随之经过计算，在屏幕上显示出三角形的面积值。如：

```

Input a, b, c           (提示)
3, 4, 5                (输入)
area= 30.00            (输出)

```

1.1.2 C 程序的结构和特点

1.1.2.1 C 程序的组成

从上面的例程序 1.1 和例程序 1.2, 可以看到, 一个 C 程序一般包括三部分: 注释行(如两个例程序中的第一行)、文件包含命令(如两个例程序中的第二行)和函数(两个例程序中 main() 及以下的部分)。其程序的组成如下所示(指简单的 C 程序):

```

          文件包含命令
          函数说明
C 程序  函数  函数体  说明语句
          函数体  执行语句
          注释行

```

1.1.2.2 函数

C 程序的主要组成部分是函数(C 的独立子程序)。一个 C 源程序至少包含有一个函数(main 函数), 也可以包含一个 main 函数和若干个其它函数, 函数是 C 程序完成程序功能的基本构件。在这些函数中, 可以是系统提供的库函数, 也可以是用户根据需要自己设计编制的函数(自定义函数)。

一个函数由两部分组成:

函数的说明部分。包括函数名、函数类型、函数形参表和形参类型。

一个函数名后面必须跟一对圆括弧, 但里面可以没有函数参数, 如 main()。

函数体。即为函数说明部分下面的最外层一对花括弧{.....}内的部分。函数体一般包括一些说明语句和执行语句。

关于函数的更详细内容, 将在后面介绍。

1.1.2.3 文件包含命令

文件包含命令用来实现 C 语言提供的文件包含功能,这是 C 提供的预处理功能之一。

所谓“文件包含”是指一个 C 程序源文件可以将另外一个源文件的全部内容包含进本程序中,然后在编译中将这两部分内容作为一个源文件单位进行编译,并得到一个目标文件。C 语言用 `# include` 命令来实现文件包含的功能。其一般形式为:

```
# include 文件名
```

其中,引号中的文件名指的是被包含文件(也称为“头文件”)的文件名。

文件包含命令的使用要注意以下几点:

在 `# include` 命令中,头文件名可以用双引号括起来,也可以用尖括弧括起来。如例程序 1.2 中,用

```
# include < math.h >
```

或

```
# include math.h
```

都是合法的。但其中也有区别:用双引号括起来的,系统先在引用该头文件的源文件所在目录中寻找该头文件;若找不到,则再按系统指定的标准方式检索其它目录。而用尖括弧括起来的,则不检索源文件所在的目录,直接按系统标准方式检索其它目录。

一个 `# include` 命令只能指定一个被包含文件。所以,如果一个程序要包含 n 个文件,就要用 n 个 `# include` 命令。文件包含命令应写在文件的开头部分。

文件包含命令的末尾不能有标点符号。

由于 C 语言的库函数分别属于不同的头文件,所以在编写一个 C 程序时,要根据在程序中所用库函数的多少和类别,引用若干条文件包含命令。

例如,例程序 1.1 中,用了文件包含命令:`# include < stdio.h >`

这是因为在例程序 1.1 中,用到了两个输入输出函数: `getchar` 和 `putchar`。而 C 语言中的输入输出函数是在头文件“ `stdio.h` ”中的。

其它函数和不同头文件的关系,可参看书后附录。

1.1.2.4 注释行

C 程序中的注释部分用 `/* */` 表示。注释可以加在程序中的任何地方,注释部分可以是任何内容。注释只是给人看的,对编译和运行不起任何作用。一个好的、有使用价值的源程序,都应当在必要的地方加上注释,以增加程序的易读性和易维护性。一个无注释的程序和使用非专业符号是程序员的严重错误之一,也是使一些阅读者恼火的重要原因。

本书中由于所举的例程序均比较简单,加上主要突出授课内容,所以忽略了这部分内容。

1.1.2.5 C 程序的特点

归纳起来,C 程序有如下特点:

程序的全部工作都是由函数来完成的,函数是用来实现特定功能的模块,所以 C 程序是一个模块化的程序。

C 的函数库十分丰富, 标准 C 提供了 100 多个库函数, 而 Turbo C 2.0 则提供了 400 多个库函数。

一个 C 程序总是从 main 函数开始执行的, 而不管 main 函数在整个程序中的位置在什么地方。而 main 函数和程序中的其他函数的位置关系是很灵活的。

C 程序中, 每个语句必须以分号(;) 结束, 分号是 C 语句的必要组成部分。

C 程序中, 一行内可以写多个语句, 同样, 一个语句也可以分写在多行上。但每个语句的结束符分号不能缺少, C 是根据有无分号来判断语句是否结束的。

1.1.3 C 程序的基本结构

为了使程序的结构清晰, 易读性强和易于维护, 以提高程序设计的质量和效率, 广泛采用了结构化程序的设计方法。所谓结构化程序, 是指由若干个基本结构组成的程序, 每个基本结构可以包含若干个语句。一般说来, 有三种基本结构:

顺序结构。顺序结构的特点是分步操作, 顺序执行。如图 1.1 中的(a): 先执行 A 操作, 然后再执行 B 操作。

选择结构(也称条件结构)。选择结构的特点在于“选择”, 即根据所给定条件的是否成立在两个操作之中选择一个操作执行。见图 1.1 中的(b)。图中的 P 代表一个给定的条件, 当条件 P 成立时执行 A 操作; 不成立则执行 B 操作。

循环结构(也称重复结构)。循环结构的特点是在某个条件下重复执行一个操作。其中有两种类型: 当型循环, 见图 1.1 中的(c)。当条件 P 成立时, 反复执行 A 操作, 直到条件 P 不成立为止。直到型循环, 见图 1.1 中的(d)。先执行 A 操作, 再判断条件 P 是否成立, 如

图 1.1 程序的三种基本结构

若成立,则再执行 A 操作,如此重复,直到条件 P 不成立为止。

关于这三种基本结构的程序设计方法,在学习到后面的内容时再具体介绍。

1.2 数据类型

数据是程序处理的对象,所以在—个程序中,应包含有对数据的描述和对数据进行的操作。C 语言中提供了多种类型的数据,这些数据类型如下:

		整 型
		字符型
基本类型		实 型(浮点型)
		枚举型
		数组类型
数据类型	构造类型	结构体类型
		共用体类型
	指针类型	
	空类型	

1.2.1 常量与变量

C 语言中的数据有常量与变量之分,它们分别属于不同的类型。用来标识变量名、符号常量名、函数名、数组名、类型名和文件名的有效字符序列称为标识符。标识符就相当于是一个名字。C 语言中规定,标识符只能由字母、数字和下划线三种字符组成,且第一个字符不能为数字。大写字母和小写字母被认为是两个不同的字符。标识符的长度没有统一的规定,但在使用中—般不要超过 8 个。

1.2.1.1 常量与符号常量

在程序运行的整个过程中,其值不能被改变的量称为常量。常量区分为不同的类型,如 123, 0, - 456 为整型常量; 12.3, 0., - 4.56 为实型常量;而 ' a ', ' b ', ' c ' 则为字符常量。

也可以用一个标识符来代表一个常量。如:

[例程序 1.3]

```
/* 650597 Lurunmin exp3.c DEFPI* /
# define PI 3.1415926
main()
{
    float area, r;
    r= 10.0;
    area= PI* r* r;
    printf( area= %f , area);
}
```

程序中用 # define 命令行定义了 PI 代表常量 3.1415926,此后凡在此程序中出现 PI 的地方,这个 PI 都代表了 3.1415926,可以用它和常量—样进行运算。

这种用一个标识符代表的一个常量,称为符号常量。习惯上,符号常量名用大写字母表示。

符号常量和变量是不同的,变量代表的是一个值,而符号常量代表的是一个字符串。定义符号常量的一般形式为:

```
# define 标识符 字符串
```

如例程序 1.3 中的

```
# define PI 3.1415926
```

它的作用是指定用标识符“PI”来代替“3.1415926”这个字符串。在编译预处理时,把程序中在该命令以后的所有“PI”都用“3.1415926”来代替。所以,符号常量不能在程序中用赋值语句来进行赋值,它一旦定义,则它的值在其作用域内就不能被改变。

“# define.....”是一个命令,不是一个语句,所以末尾不能有分号,这和文件包含命令“# include.....”是一样的。

1.2.1.2 变量

在程序运行的过程中,其值可以改变的量称为变量。每一个变量有一个相应的名字(标识符,称为变量名),在内存中占据一定的存储单元,在该存储单元中存放变量的值。习惯上,变量名用小写字母表示。在选择变量名(或其它标识符)时,最好选用有相应含意的英文单词(或缩写)或者汉语拼音,尽可能做到“见名知意”,也可以使用专业符号,以增加程序的可读性。

在 C 程序中,要求做到对所有用到的变量在使用之前先定义。变量的定义包括两方面的内容:即说明变量所属的确定数据类型和变量的名字,系统在编译时就为其分配存储单元。如果在程序中所用到的一个变量在这之前没有被预先定义,则程序在被编译时会检查出“变量未经定义”的错误。这一点和 FORTRAN 语言有明显的差别。用惯 FORTRAN 语言的读者应特别加以注意。

1.2.2 整型数据

1.2.2.1 整型常量

整型常量即整常数。在 C 语言中,整常数可以用如下三种形式来表示:

十进制整数。如 518, - 19, 0 等。

八进制整数。八进制整数的表示是以 0 开头,如 0123 表示八进制数的 123,等于十进制数中的 83。即 $(123)_8 = 1 \times 8^2 + 2 \times 8^1 + 3 \times 8^0 = 83$ 。- 012 等于十进制数的 - 10。

十六进制整数。十六进制数是以 0x 开头的,如 0x123,表示十六进制数的 123,即 $(123)_{16}$,它等于十进制数中的 291,即 $(123)_{16} = 1 \times 16^2 + 2 \times 16^1 + 3 \times 16^0 = 291$ 。- 0x11 等于十进制数的 - 17。

1.2.2.2 整型变量

整型变量的分类

整型变量可分为以下 4 种:

- 基本型, 以 int 表示。
- 短整型, 以 short int(或 short)表示。
- 长整型, 以 long int(或 long)表示。
- 无符号型, 无符号型变量只能存放不带符号的整数, 故不能存放负数。无符号型中又分为无符号整型(unsigned int)、无符号短整型(unsigned short)和无符号长整型(unsigned long)。

C 标准没有具体规定以上各类数据为存储所需的字节数, 各种机器在处理上有所不同。以 IBM PC 为例, 各类整数的存储长度和范围如表 1.1 所示。

表 1.1

类型	存储位	数的范围
int	16	- 32768 ~ 32767
short	16	- 32768 ~ 32767
long	32	- 2147483648 ~ 2147483647
unsigned int	16	0 ~ 65535
unsigned short	16	0 ~ 65535
unsigned long	32	0 ~ 4294967295

整型变量的定义和赋初值

因为 C 语言规定, 在程序中所有用到的变量都必须在使用之前先定义。所以对于变量的定义, 一般均是放在一个函数的开头部分。如:

[例程序 1.4]

```
/* 650597 Lurunmin exp4.c DEFV* /
main()
{
    int a, t;
    long vt, v0;
    a= 2; t= 35; v0= 10;
    vt= v0+ a* t;
    printf( vt= %d , vt);
}
```

在以上的例程序 1.4 中, 函数的开头部分:

```
int a, t;    (指定变量 a、t 为整型)
long vt, v0; (指定变量 vt、v0 为长整型)
```

在程序中, 常需要对一些变量预先设置初值。在 C 程序中, 可以在定义变量的同时对变量赋初值。如:

```
int a= 2; (指定变量 a 为整型, 且初值为 2)
```

它相当于:

```
int a;    (指定变量 a 为整型)
```

a= 2; (赋值语句, 将值 2 赋于 a)

也可以使被定义的变量中的一部分赋以初值。如:

```
int x, y, z= 10;
```

它相当于:

```
int x, y, z;
```

```
z= 10;
```

如果要对多个变量赋以同一个初值, 则可以写成:

```
int x= y= z= 10;
```

1.2.3 实型数据

1.2.3.1 实型常量

实型数在 C 语言中又可称为浮点数。它有两种表示形式:

十进制数形式。它和我们通常所用的小数一样, 由数字和小数点组成, 并且必须有小数点。如: 0.518, .518, 518.0, 518., 0.0 等都是用十进制数形式表示的实型数。

指数形式。如 5.18e2 或 5.18E2 都表示了数 5.18×10^2 , 即 518.。在用指数形式表示一个实型数时, 要注意字母 e(或 E)之前必须有数字, 且 e 后面的指数必须为整型数。所以, 如 e2, 5.18e2.5 等都是错误的表示方法。

1.2.3.2 实型变量

实型变量分为单精度(float)型和双精度(double)型两类。对每一个实型变量也都应该在使用之前加以定义。如:

```
float x, y;           (指定变量 x, y 为单精度实型)
```

```
double u, v;         (指定变量 u, v 为双精度实型)
```

在一般的计算机系统中, 一个单精度实型数据在存储器中占 4 个字节(32 位), 一个双精度实型数据占 8 个字节(64 位)。单精度实型数据提供 7 位有效数字, 双精度实型数一般提供 16 位有效数字。其数值的范围随机器系统而不同, 在 IBM PC MS-C 中, 单精度实型数的数值范围约为 $-10^{38} \sim 10^{38}$ 之间, 双精度实型数的范围约为 $-10^{308} \sim 10^{308}$ 之间。

对于实型变量, 同样可以在定义的同时给变量赋初值, 如:

```
float x, y= 51.8;
```

其规则和前面所述的整型变量赋初值的规则相同。

1.2.4 字符型数据

1.2.4.1 字符型常量

C 语言中的字符常量是用单引号括起来的一个字符, 如 'a', 'm', 'A', '?', '#' 等都是字符常量。要注意, 字符常量只包含有一个字符, 另外大写和小写(如 'a' 和 'A')是不同的字符常量。

除了以上形式表示的字符常量外, C 还允许用一种特殊形式表示的字符常量, 它是以一个 "\ " (反斜杠) 开头的字符序列, 这种字符称为“转义字符”, 意思是将反斜杠(\)后面的字符

转变成另外一种意思,而不是它本身原来的意思了。如‘\n’中的n不再代表字母n而作为“换行”符。常用的转义字符见表1.2所列。

表 1.2

字符形式	功 能
\n	换行
\t	横向跳格
\v	竖向跳格
\b	退格
\r	回车
\f	走纸换页
\\	反斜杠字符“\”
\	单引号(撇号)字符
\ddd	1 到 3 位 8 进制数所代表的字符
\xhh	1 到 2 位 16 进制数所代表的字符

表 1.2 中的最后第二行是用 ASCII 码(八进制数)表示一个字符的方法。例如‘\101’代表字符‘A’;‘\012’代表“换行”;‘\0’或‘\000’代表 ASCII 码为 0 的控制字符,即“空操作”字符等。

1.2.4.2 字符型变量

字符型变量用来存放字符常量。一个字符变量只能存放一个字符,不要误认为一个字符变量中可以存放一个字符串(包括多个字符)。

字符型变量的定义格式如下:

```
char c1, c2;    (指定变量 c1 和 c2 为字符型)
```

一般,一个字符变量在存储器中占据一个字节,即用一个字节来存放一个字符。

用和整型变量、实型变量同样的方法,可以给字符型变量赋初值。如:

```
char c1= m , c2= n ;
```

这等效于

```
char c1, c2;
```

```
c1= m ; c2= n ;
```

1.2.4.3 字符型数据和整型数据的关系

任何数据在计算机内部都是以二进制的形式表示的,所以从本质上来说,它们都是一个数。对于一个字符来说也是一样,我们在存储一个字符的时候,不是存储“字符”本身,而是存储了和该字符相对应的 ASCII 代码,而 ASCII 代码是以整数的形式表示的。例如字符“m”的 ASCII 代码值是 109,字符“n”的 ASCII 代码值是 110。所以字符型数据和整型数据的存储具有相同的形式,因而它们之间是可以通用的。这意味着,一个字符型数据既可以以字符

形式输出,也可以以整数形式输出。以字符形式输出时,需要先将存储单元中的 ASCII 码转换成相应的字符,然后输出。以整型数形式输出时,就直接将相应的 ASCII 码作为整数输出。并且字符型数据和整型数据之间可以互相赋值。例如:

[例程序 1.5]

```
/* 650597 Lurunmin exp5.c C-I* /
main()
{
    char c1, c2;
    int i, j;
    c1= 109; c2= 110;
    i= m ; j= n ;
    printf( % c, % d\ n , c1, c2);
    printf( % d, % c\ n , i, j);
}
```

输出结果为:

```
m, 110
109, n
```

因为字符型数据具有整型数据的存储形式,所以也可以对字符型数据进行算术运算,此时相当于对它的 ASCII 码值进行算术运算。例如:

[例程序 1.6]

```
/* 650597 Lurunmin exp6.c CUL* /
main()
{
    char c1, c2;
    c1= m ; c2= n ;
    c1= c1- 32; c2= c2- 32;
    printf( % c, % c , c1, c2);
}
```

输出结果为:

```
M, N
```

它的功能是将两个小写字母转换成两个大写字母。同一个英文字母大写和小写之间的 ASCII 码值相差 32。

1.2.4.4 字符串常量

字符串常量是用一对双引号括起来的字符序列。如“ Hello! ”,“ m ”,“ n= 110 ”等都是字符串常量。

可以输出一个字符串常量,如:

```
printf( How are you. );
```

不要将字符常量和字符串常量相混淆。‘ m ’是字符常量,而“ m ”则是字符串常量,二者