函授教育专用

C 语 言

C 语 言

范亚军

教材编写组选编

目　录

第一章　基本数据类型

在程序员日常所用有词汇中，“数据”一词大概是用得最多的一类。如果程序员之间谈业务问题而不提“数据”，那几乎是不可能的事。

在本书中，“数据”一词具有特定的意义，数据可以是常数和变量。常数具有固定不变的值，而变量则是可变的。正如我们已经知道的，变量通常用具有某种意义的名字来命名，变量在不同的时候可以有不同的值。

在C语言中，有两种基本不同的数据类型：整型和浮点型。从这两种类型，又可以得到另外两种基本数据类型：字符型和双精度型。一共有四种基本数据类型。

在C语言程序中，我们所用的数据绝大多数都是整型或者是字符型。另外两种基本类型用得比较少。

看起来，只有四种基本类型似乎是太少了。如果仅有这么四种，那的确是太少。但幸好还可以从这些基本类型推出许多导出类型，以后将要讨论的指针、数组、结构和联合等就属于这种导出类型。事实上可以导出和定义的类型数目在理论上是没有限制的。C语言的程序员可以发明他自己需要的任何数据类型。这一点，可以说是C语言的模块化结构体系的又一例子，即是说，通过用少数的规则，一些较小而简单的单元就可以合在一起以构成一些精巧而有用的结构。

§1.1 整型

整型变量是指没有分数部分的所有的数，如 1、2、3、－1、0 等，整型变量可以有正的或负的值。

在整型的表示符号 int 之前，还可以加上一个或两个以下的符号：short 、long、unsigned。加上这些符号后，又可以在整型(int)之外构成短整型(short int)、长整形(long、int)，无符号整型(unsigned int)、无符号短整型(unsigned short int)和无符号长整型(unsigned long int)，一共有六类。根据所有相同的表示(如整型和短整型等)。下面是说明不同整型变量的几个说明语句(其中的 int 可省略)：

```
short int x;
long int y;
unsigned int z;
```

整型值可以用来表示整数、数组下标、逻辑值，二进制字符串等。它的取值范围视所用的机器和编译程序而定。int 通常反映了所用机器的最"自然的"字长，而编译程序可自由解释适合自己硬件的 long 和 short。

整形变量的典型取值范围为－32768～＋32767；无符号整型变量的典型取值范围为 0～65535。表 1.1 是在不同机器上实现的几个例子。

表 1.1

	DEC PDP—11 ASCII	Honeywell 6000 ASCII	IBM 370 EBCDIC	Interdate 8/32 ASCII
char	8 bits	9 bits	8 bits	8 bits
int	16	36	32	32
short	16	36	16	16
long	32	36	32	32
float	32	36	32	32
double	64	72	64	64

整型常数在使用之前是不需要加以说明的，而整型变量在使用前则必须加以说明。说明语句通常放在用花括号括起来的一个程序块的上面部分，如：

```
{
  int a;
  int b;
  int c;
  ⋮
}
```

也可以在一行内对几个变量同时加以说明，如：

```
{
  int a,b,c;
  ⋮
}
```

说明整型变量的关键字是 int。说明以后，就可以用赋值运算符＝来对它们进行赋值了，如：

```
{
  int a,b,c;
```

```
    ⋮
    a=123;
    b=45;
    c=678;
}
```

这里，还要提醒读者注意，赋值运算符与数字中等号是一样的，但意义却不一样。诚然，在上述的语句中，我们已把变量a赋为“等于”123，把b赋为“等于”45等，但它只是一个把常数赋值给变量的操作，完全不是相等语句。例如，表达式

```
x=x+1
```

是完全合法的，它把x的值加1，但要认为它是“等于”则就不对了（显然x不等于x+1）。由此，我们可以体会出它们的差别。

变量说明的次序可以是任意的，具体地说，说明的次序并不需要与该程序块中变量出现的次序保持一致。

§1.2 字 符 型

与整型一样，字符既可是常数，也可以是变量。通常字符型可以用来表示一个ASCII字符，也可表示－128～＋127之间的整数。无符号字符可以用来表示一个字节或0～255之间的整数。字符和无符号字符在内存中都占一个字节。字符变量用关键字char在一个程序块的开始部分加以说明，如

```
char sex,party;
```

字符常数由单引号引起来的字符表示，'1'、'A'、'*'等都是字符常数。对于那些不可显示的控制类字符，如换行符、制表符、退格符等，用一反斜线开头的转义序列表示，看起来象两

个字符，实际上与一个单一的字符等效。因此，'\n'、'\t'、'\b' 等也是字符常数。

我们已经看到，对于整型变量，可以很直观地知道它的值，例如：a 的值为 123，b 的值为 45 等。但如象 'A'、'1'、'\n' 这样的字符常数又如何估值呢？"字符值"究竟是什么呢？是常数呢？还是变量？事实上，在 C 语言中，字符是数字值。通常采用的规范是把它与美国标准信息交换代码(ASCII)联系起来。在 ASCII 中，大写字母 'A' 是 65，'B' 是 66 等，而小写字母 'a' 是 97，'b' 是 98 等。对空格、制表、回车换行等比较特殊的字符，都有相对应的规定数值。

使用整数表示字符，使得在很多情况下 int 和 char 变量值可以互相交换。

§1.3 浮 点 型

由于 C 语言最适宜于用来做系统软件的工作——写解释程序、编译程序、操作系统、编辑程序等，因此浮点型数据在用 C 语言编程的场合中起不了多大的作用。虽然如此，当把 C 语言用来解决诸如统计分析、数据压缩、线性规划等方面的应用时，的确有使用浮点型数的必要，为使 C 语言有更广泛的应用面，使之具有处理浮点数的功能也是很重要的。

浮点数据的值是实数的近似值，形如 3.1415 或 0.01745 这样的数都是实数。很大或很小的实数值常常写成指数形式或用所谓"科学"表示法表示。如阿佛加德罗常数 6.02×10^{23} 和普朗克常数 6.626×10^{-27} 可以分别写成 6.02e+23 和 6.626e-27 这样的科学表示法，其中 e 代表指数(exponent)。

除非常数很大或很小，通常 C 程序员都避免使用科学表

示法，但C语言在内部存储所有的浮点数时都仍然按指数形式存储，而不管其大小如何。这些存储的细节不属于一般读者关心的范围，但有一个重要之点必须引起读者的注意，即用软件或硬件处理浮点数时都比处理整型或字符型的数据有较大的开销。

用来说明浮点型变量的关键字是float，下面是一段说明和赋值：

```
float x,y,z;
    ⋮
x=12.345;
y=.12345e+2;
z=12345.0e-3;
```

上面这个例子示出了用三种不同的方法，把x、y、z赋值成相同的浮点常数。

要注意的是，在浮点型表示法中，数都是近似的，误差积累很快。通过使用下面将要讨论的第四种基本数据类型（双精度型），在程序的大小和运行速度方面作出一些牺牲的情况下可以把误差减小。

为了对浮点数的数值范围有个概念，这里列出一个典型的C语言编译程序所支持的数值范围。其值大约（因浮点数都是近似值）在±2.9387e－39～±1.7014e＋38之间，小数点后的精度大约有六位尾数。

§1.4 双 精 度 型

简单地说，双精度值就是一种具有较高精度的浮点值，它所占的存储空间为浮点型的两倍。也要注意，双精度的使用并

不保证使所得结果的有效数字的位数加倍，但却大大地改善了计算的精度，并减小了舍入误差的积累。具体能得到的精度随不同的机器而异。为了说明双精度型和浮点型的差别，曾经在机器上做过如下的实验：将变量 x 说明为 float，而 y 说明为 double，先把这两个变量置为 0.0，再分别与 100000.0 相加，共执行一万次，然后再分别用 100000.0 除，最后得到 y 值是 100000.000000，而 x 值却为 100014.820313，x 有 0.015% 的误差。

双精度变量用关键字 double 加以说明，如：

```
double logE,sin－per－sec;
        ⋮
logE＝2.718281828459;
sin－per－sec＝4.848136811076e－7;
```

可以看到，双精度常数用了与单精度浮点常数一样的指数表示，存在的唯一区别是有效数字的位数。编译程序处理实型常数时，都当成双精度来处理。

§1.5 变量的说明及初始化

在C 语言中，所有的变量都必须在使用前加以说明。说明语句包括两个部分：说明类型的关键字及关键字后面跟着的一个以上的变量，如：

```
int lower,upper,step;
char c,line[1000];
```

说明语句中的变量可以按任意方式分布，因此，上面的两个说明语句也可写为如下形式：

```
int lower;
```

```
int upper;
int step;
char c;
char line[1000];
```

后一种方式占用了更多地方，但易于给每一个说明加上注释，以后作修改时就比较方便。

变量的初始化也就是给变量赋初值。此前我们已经看到了一些先对变量加以说明，然后再赋值的例子，如：

```
double logE
    ⋮
logE=2.718281828459;
```

当我们写一个程序的时候，常常是以给变量赋初始值作为开始。可以把上述的两种操作合并在一个初始化过程中，如：

```
double logE=2.718281828459;
```

浮点型、整型和字符型的变量也可以用同样的办法加以初始化，如：

```
float   x=12.345;
int     rank=473;
char    sex='f';
```

当有几个变量时，也可以按需要仅对其中的一个进行初始化，如：

```
floar x,y,z=12.345;
```

它与以下的几条语句等效；

```
float x;
float y;
float z=12.345;
```

上述的两种方法都仅仅对变量 z 进行了初始化，而变量 x 和

y 未受影响。

若变量是外部的或静态的，那么初始化就要一次作好。从概念上说，在程序开始执行之前就要完成这些初始化动作。经过显式初始化的自动变量在其所在函数被调用时每次都要重新赋值。如果自动变量没有显式地加以初始化，则其值是不确定的(即无用信息)。外部及静态变量在未加说明的情况下被初始化为 0，但不管如何，明确地加以初始化是一种好的程序设计风格。

后面介绍新数据类型时还将进一步讨论其初始化问题。

§1.6 常 数

在C语言中，常数可以分为直接常数和符号常数两类。直接常数可以直接使用(如 1、1.0、'1' 等)，而符号常数在使用前必须使用编译命令加以定义。在上一章中，已经讨论过用编译命令"#define"来定义符号常数，如：

```
#define MAXLINE 1000
```

用符号常数可改善程序的可读性。

直接常数可以用多种数据类型。前面，我们已经讨论过整型、浮点型这样的常数，下面再讨论一下其他类型。

长型(long)常数写成如 123L 这样的模样。通常太长而不能用 int 型表示的整常数可采用 long 型。

现介绍一下代表八进制和十进制的记号：以 0(零)打头的 int 整常数是八进制；以 0x 或 0X 打头的是十六进制。比如，十进制数 31 可写成八进制的 037，或写成十六进制的 0x1f 或 0X1F。十六进制数和八进制数也可后跟 L 成为 long 型。

字符常数是单引号中的单个字符，如'x'。任一字符常数的值就是机器所用字符集中该字符的数值值。例如，在ASCII字符集中的字符零，即'0'，是48。在EBCDIC码集中，'0'为240，两种码都不是数值0。写'0'而不写48或240使程序可独立于特定的值。字符常数可象其他数字一样参加数值运算，但主要还是用在与其他字符的比较上。后面有一节要专门谈转换规则。

前已述及，某些非图形字符可用转义序列以字符常数表示出。如：\n(换行)，\t(制表)，\o(空白)，\\(反斜线)，\'(单引号)，看起来都是两个字符，其实与一个单一字符等效。任意的按字节计算的二进位模式可以用格式

```
'\ddd'
```

来产生，其中ddd代表1～3个八进制数。于是，可以用

```
#define FORMFEED'\014'      /* ASCII form feed */
```

来定义变量FORMFEED(意为换行)。因为在ASCII码中，换行符对应的八进制数为014。

字符常数'\0'代表具有0值的字符。'\0'常用来代替0以强调某些表达式的字符特性。

常数表达式是只包含常数的表达式。这种表达式是在编译时估值，而不是在运行时估值。常数表达式可用在常数可以出现的任何地方，如：

```
#define MAXLINE 1000
char line[MAXLINE+1];
```

或者

```
seconds=60*60*hours;
```

串常数由0个或多个字符序列组成，并用双引号引起来，

如：

"I am a string"

或者

"" /* a null string */

注意：这里所用的双引号并不是字符串的一部分，只当作定界符用。在字符串中同样可以使用转义序列。符号\" 表示双引号字符。

从技术上讲，字符串是元素为单位字符所组成的数组 。编译程序自动将 0 字符\0 放在每个串的末尾以标示结束。这种表示法意味着对串长没有限制，但程序只有对整个串全部扫描之后，才能决定串的长度。需用的物理存储单元比写进双引中之间的字符数目多一个。下面的函数 strlen(s)返回字符串 s 的长度，但不包含结尾的\0。

```
/* strlen:return length of s */
int strlen(char s[])
{
    int i;
    i=0
    while (s[i]! ='\0')
      ++i
    return i;
}
```

注意区别字符常数和只有单个字符的串：'x' 与"x"不一样，前者是单个字符，用以产生字母 x 在机器字符集中的数值值；后者是字符串，它包含一个字符 (字母 x)及一个\0。

§1.7 混合运算及类型转换

如果在表达式中用了各种类型的操作数，将出现什么情况呢？比如，用一个整数来除浮点数，得到的结果显然只能是一种数据类型，而不是两种。但究竟是哪一种呢？

当不同的数据类型混合在一个算术表达式中时，就要出现类型变换。变换哪一个呢？规则也相当简单。已讨论的各种基本数据类型可以排出以下的等级

char<int<long<float<double

右边的比左边的等级高。类型转换以隐含的方式自动完成。一般地，象"+""*"这样要用两个操作数的二元运算符，若它具有类型不同的操作数，类型较低的操作数就要在运算之前提升成"比较高级"的类型。结果也是那种比较高级的类型。更精确的说，对每一种算术运算符，要运用下列规则：

• 若等级最高的操作数是 double 型，则其他操作数（不管类型如何）都变换成 double 型，且结果亦为 double 型。

• 若等级最高的操作数为 float 型，则其他操作数（不管类型如何）都变换成 float 型，且结果也为 float 型。

• 若等级最高的操作数为 long 型，则其他操作数（不管类型如何）都变换成 long 型，且结果也为 long 型。

• 若等级最高的操作数为 int 型，则 char 型，（此时也只有 char 型）要转换成 int 型，且结果为 int 型。

类型转换也出现在赋值操作符的两边，规则也很简单，不管赋值符右边的类型如何，都变换成左边的类型。如果右边的值具有较高的等级，则将被截短或舍入，以和赋值符左边的变量类型一致。换句话说，左边的变量总是具有较高的优先