第3章

数据类型及其应用

本章主要知识点：

- C语言的数据类型
- 常量和变量
- 运算符和表达式
- 数据的输入/输出

熟练掌握C语言中的基本数据类型及其应用，是正确进行C语言程序设计的基础。

在C语言中，对不同的数据用不同的数据类型来区分。程序在执行的过程中，需要对数据进行运算，也需要存储数据。这些数据通过变量存储在内存中，以便程序随时取用。由于不同的数据在存储时所需要的容量各不相同，所以不同的数据就需要分配不同大小的内存空间来存储。

C语言中，数据类型可分为基本数据类型、构造数据类型、指针类型和空类型四大类。其详细内容如图3-1所示。

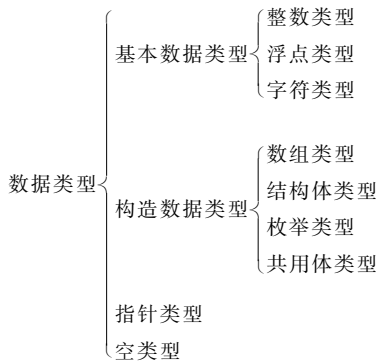

图3-1　C语言的数据类型

1. 基本数据类型

基本数据类型最主要的特点是其值不可以再分解为其他类型。基本数据类型包括整数类型、浮点类型和字符类型。

2. 构造数据类型

构造数据类型是根据已定义的一个或多个数据类型用构造的方法来定义的。也就是说，一个构造类型的值可以分解成若干个“成员”。每个“成员”都是一个基本数据类型或是一个构造类型。构造类型包括数组类型、结构体类型、枚举类型和共用体类型。

3. 指针类型

指针类型是一种特殊的，同时又是具有重要作用的数据类型。其值用来表示某个变量

在内存储器中的地址。

4. 空类型

在调用函数时，通常应向调用者返回一个函数值。这个返回的函数值是具有确定的数据类型的，应在函数定义时予以说明。如果函数调用后并不需要向调用者返回函数值，这种函数可以定义为“空类型”。其类型说明符为 void。

在程序中用到的所有数据都必须指定其数据类型。数据有常量和变量之分，它们分别属于以上数据类型。

本章主要介绍基本数据类型及其应用。

3.1　常　　量

在程序执行过程中，其值不发生改变的量称为常量。如 y＝x＋5 中的 5 在程序运行过程中其值始终不改变，它就是常量。

C 语言中的常量分为数值型常量、字符型常量、符号常量三种类型，其中数值型常量又分为整型常量和实型（浮点型）常量；字符型常量又分为字符常量、转义字符和字符串常量。常量分类如表 3-1 所示。

表 3-1　常量分类表

常量分类		常量示例
数值型常量	整型常量	－123，076，0x2a
	实型（浮点型）常量	0.618，3.14e5
字符型常量	字符常量	‘P’，‘2’
	转义字符	\n，\t
	字符串常量	“Hello!”
符号常量		＃define PI 3.14159

3.1.1　数值型常量

数值型常量分为整型常量和实型（浮点型）常量两种。

1. 整型常量

整型常量就是整常数。在 C 语言中，使用的整常数有十进制、八进制和十六进制三种。

（1）十进制整常数

十进制整常数没有前缀。其数码为 0～9。

以下各数是合法的十进制整常数：

237　　－568

在程序中是根据前缀来区分各种进制数的。因此在书写常数时不要把前缀弄错造成结果不正确。

(2) 八进制整常数

八进制整常数必须以 0 开头,即以 0 作为八进制数的前缀。其数码取值为 0～7。

以下各数是合法的八进制数:

015(十进制为 13)　0101(十进制为 65)

以下各数不是合法的八进制数:

256(无前缀 0)　03A2(包含了非八进制数码 A)

(3) 十六进制整常数

十六进制整常数的前缀为 0X 或 0x。其数码取值为 0～9,A～F。

以下各数是合法的十六进制整常数:

0X2A(十进制为 42)　0XA0(十进制为 160)

以下各数不是合法的十六进制整常数:

5A(无前缀 0X)　0X3Y(含有非十六进制数码 Y)

(4) 整型常数的表示范围

整型常数分为有符号和无符号两类,其表示数值的范围不同。

十进制无符号整常数的范围为 0～65535,有符号数的范围为－32768～＋32767。

八进制无符号数的表示范围为 0～0177777。

十六进制无符号数的表示范围为 0X0～0XFFFF。如果使用的数超过了上述范围,就必须用长整型数来表示。长整型数是用后缀"L"或"l"来表示的。例如:

十进制长整型常数　158L(十进制为 158)

八进制长整型常数　012L(十进制为 10)　077L(十进制为 63)

十六进制长型整常数　0X15L(十进制为 21)　0XA5L(十进制为 165)

长整数 158L 和基本整常数 158 在数值上并无区别。但对 158L,因为是长整型量,C 编译系统将为它分配 4 个字节存储空间。而对 158,因为是基本整型,只分配两个字节的存储空间。因此在运算和输出格式上要特别注意,避免出错。

无符号数也可用后缀表示,整型常数的无符号数的后缀为"U"或"u"。例如:358u,0x38Au,235Lu 均为无符号数。另外,前缀后缀可同时使用以表示各种类型的数。如 0XA5Lu 表示十六进制无符号长整数 A5,其十进制为 165。

2. 实型常量

实型常量也称为实数或者浮点数。在 C 语言中,实数只采用十进制。它有两种形式:十进制数形式与指数形式。

(1) 十进制数形式

由数码 0～ 9 和小数点组成。例如:0.0,.25,5.789,0.13,5.0 等均为合法的实数。

(2) 指数形式

由十进制数和阶码标识"e"或"E"以及阶码(只能为整数,可以带符号)组成。其一般形式为:a E n(a 为十进制数,n 为十进制整数),其值为 $a\times10^n$。

例如:2.1E5(等于 2.1×10^5),3.7E－2(等于 3.7×10^{-2})。

以下不是合法的实数:

345(无小数点)

E7（阶码标识 E 之前无数字）

53.－E3（负号位置不对）

2.7E（无阶码）

标准 C 允许浮点数使用后缀。后缀为“f”或“F”即表示该数为浮点数。如 356f 和 356.0 是等价的。

【实例 3-1】 实型常量示例。

```
void main()
{
    printf("%f\n%f\n",123.,123f);
}
```

运行结果如图 3-2 所示。

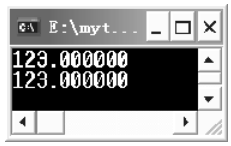

图 3-2　实例 3-1 的运行结果

3.1.2　字符型常量

1. 字符常量

字符常量是用单引号括起来的一个字符。例如‘a’,‘＝’,‘?’都是合法的字符常量。在 C 语言中,字符常量有以下特点：

(1) 字符常量只能用单引号括起来,不能用双引号或其他括号。

(2) 字符常量只能是单个字符,不能是字符串。

(3) 字符可以是字符集中的任意字符。数字被定义为字符型之后就不能参与数值运算。

2. 转义字符

转义字符是一种特殊的字符常量。转义字符以反斜线“\”开头,后跟一个或几个字符。转义字符具有特定的含义,不同于字符原有的意义,故称“转义”字符。例如,在前面各例题中“printf”函数的格式串中用到的“\n”就是一个转义字符,其意义是“换行”。转义字符主要用来表示控制代码。

常用的转义字符及其含义如表 3-2 所示。

表 3-2　转义字符及其含义

转义字符	转义字符的意义
\n	换行
\t	横向跳到下一制表位置
\v	竖向跳格
\b	退格
\r	回车
\f	走纸换页
\\	反斜线符“\”
\’	单引号符
\ddd	1～3 位八进制数所代表的字符
\xhh	1～2 位十六进制数所代表的字符

广义地讲,C 语言字符集中的任何一个字符均可用转义字符来表示。表 3-2 中的\ddd 和\xhh 正是为此而提出的。ddd 和 xhh 分别为八进制和十六进制的 ASCII 代码。如\101 表示字母“A”,\102 表示字母“B”,\134 表示反斜线,\x0A 表示换行等。

【实例 3-2】 转义字符的使用。

```
void main()
{
    int a,b,c;
    a = 5; b = 6; c = 7;
    printf( " % d\n\t % d % d\n % d % d\t\b % d\n",a,b,c,a,b,c);
}
```

运行结果如图 3-3 所示。

程序在第一列输出 a 的值 5 之后,接着就是“\n”,即换行;接着是“\t”,于是跳到下一制表位置(设制表位置间隔为 8),输出 b 的值 6,接着输出 c 的值 7 后又是“\n”,因此再换行;在第三行首先输出 a 的值 5,接着输出 b 的值 6,然后又是“\t”,于是跳到下一制表位置,接着又是“\b”,表示退回一格,再输出 c 的值 7。

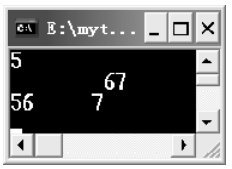

图 3-3 实例 3-2 的运行结果

3. 字符串常量

字符串常量是由一对双引号括起来的字符序列。例如“CHINA”,“C program:”,“$12.5”等都是合法的字符串常量。字符串常量和字符常量是不同的,它们之间的区别主要有:

(1) 字符常量由单引号括起来,字符串常量由双引号括起来。

(2) 字符常量只能是单个字符,字符串常量则可以包含零个或多个字符。不包含任何字符的字符串为空串。

(3) 可以把一个字符常量赋予一个字符变量,但不能把一个字符串常量赋予一个字符变量。

(4) 字符常量占一个字节的内存空间。

对于字符串,C 语言规定以字符‘\0’作为结束标识,系统将根据该字符来判断字符串是否结束。字符‘\0’由系统自动加入到每个字符串的结束处,不必人工加入。例如,字符串“China”在内存中所占的字节序列为:China\0。字符串常量占的内存字节数等于字符串中字节数加 1。字符常量‘a’和字符串常量“a”虽然都只有一个字符,但在内存中的情况是不同的。

‘a’在内存中占一个字节,可表示为:a。

“a”在内存中占两个字节,可表示为:a\0。

3.1.3 符号常量

在 C 语言中,可以用一个标识符来表示一个常量,称之为符号常量。符号常量在使用之前必须先定义,其定义形式为:

```
#define  标识符  常量
```

其中#define是一条预处理命令(预处理命令都以“#”开头),称为宏定义命令,其功能是把该标识符定义为其后的常量值。符号常量一经定义,那么以后在程序中所有出现该标识符的地方均代之以该常量值。

【实例 3-3】 符号常量的使用。

```
#define PI 3.14159
void main()
{
   float s,r;
   r = 10;
   s = PI * r * r;
   printf("s = %f\n",s);
}
```

运行结果如图 3-4 所示。

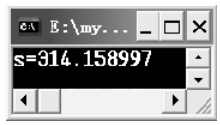

图 3-4　实例 3-3 的运行结果

本程序在主函数之前由宏定义命令定义 PI 为 3.141 59,在程序中即以该值代替 PI。s = PI * r * r 等效于 s = 3.141 59 * r * r。应该注意的是,符号常量不是变量,它所代表的值在程序中不能再改变。也就是说,在程序中,不能再用赋值语句对它重新赋值。

点拨　习惯上符号常量的标识符用大写字母,变量标识符用小写字母,以示区别。

3.2　变　　量

本节主要介绍变量的分类、变量的取值与变量的地址。

3.2.1　变量的要素

在 C 程序执行过程中,其值可以改变的量称为变量。

每一个变量都有三个基本要素,即变量的名称、变量的类型和变量的值。

例如:

```
int  age = 20;
```

从这一行我们可以得到以下信息:变量的名称为 age;变量的类型为整数类型,在内存中占两个字节的存储空间;变量的当前值是 20。这三者的关系如图 3-5 所示。

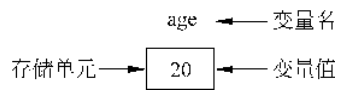

图 3-5　变量名与变量值的关系图

要使用变量,必须先给变量命名。

C 语言变量命名规则是:

(1) 变量名只能由字母、数字和下划线组成,且只能由字母和下划线开头。

(2) 变量名须区分字母的大小写。

(3) 尽量取一些有意义的名称,做到见名知意。如用 sum 表示求和,用 age 表示年龄等。

(4) 变量名不能是 C 语言的保留字(关键字)。

C 语言保留字如下所示。

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

按照变量名的命名规则,sum、x10、_p9 等是合法的变量名,而 211dx、do、x+y 等是错误的变量名。

3.2.2 变量的分类

C 语言中的变量分为整型变量、实型变量和字符型变量三种类型。

1. 整型变量

整型变量可分为基本型、短整型、长整型和无符号型。各种整型变量的类型说明符、所占的内存空间字节数与数的表示范围如表 3-3 所示。

表 3-3 整数类型的有关数据

类 型	类型说明符	分配字节数	数 的 范 围
整型	int	2	−32 768～32 767 即 -2^{15}～$2^{15}-1$
无符号整型	unsigned int 或 unsigned	2	0～65 535 即 0～$2^{16}-1$
短整型	short int 或 short	2	−32 768～32 767 即 -2^{15}～$2^{15}-1$
无符号短整型	unsigned short	2	0～65 535 即 0～$2^{16}-1$
长整型	long int 或 long	4	−2 147 483 648～2 147 483 647 即 -2^{31}～$2^{31}-1$
无符号长整型	unsigned long	4	0～4 294 967 295 即 0～$2^{32}-1$

点拨 在 C 语言程序中,常量是可以不经说明而直接引用的,而变量则必须先说明后使用。

变量说明的一般形式为:

```
类型说明符  变量名标识符,变量名标识符,……;
```

例如:

```
int  a,b,c; (a,b,c 为整型变量)
```

```
long  x,y; (x,y为长整型变量)
unsigned  p,q; (p,q为无符号整型变量)
```

在书写变量说明时,应注意以下几点:

(1) 允许在一个类型说明符后,说明多个相同类型的变量。各变量名之间用逗号间隔,类型说明符与变量名之间至少用一个空格间隔。

(2) 最后一个变量名之后必须以";"结尾。

(3) 变量说明必须放在变量使用之前。

【实例 3-4】 整型变量示例。

```
void main(){
    long x,y;
    int a,b,c,d;
    x = 1 ;
    y = 2 ;
    a = 3 ;
    b = 4 ;
    c = x + a ;
    d = y + b ;
    printf("c = x + a = %d,d = y + b = %d\n",c,d) ;
}
```

运行结果如图 3-6 所示。

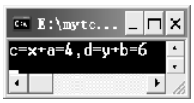

图 3-6　实例 3-4 的运行结果

主函数 main 返回类型为 void,即表示不返回任何类型的值。

从程序中可以看到:x,y 是长整型变量;a,b 是基本整型变量。它们之间允许进行运算,运算结果为长整型。但 c,d 被定义为基本整型,因此最后结果为基本整型。其中的类型转换是由编译系统自动完成的。有关类型转换的规则见第 3.3.8 节。

2. 实型变量

实型变量分为两类:单精度类型和双精度类型。

表 3-4　实型数据的表示

类　型	类型说明符	分配字节数	有效数字	数 的 范 围
单精度类型	float	4	6～7	$-3.4\times10^{-38}\sim3.4\times10^{38}$
双精度类型	double	8	15～16	$-1.7\times10^{-308}\sim1.7\times10^{308}$

实型变量说明的格式和书写规则与整型相同。

例如:

```
float  x,y;              (x,y为单精度实型变量)
double  a,b,c;           (a,b,c为双精度实型变量)
```

实型常数不分单、双精度,都按双精度 double 型处理。

【实例 3-5】 实型变量示例。

```
void main()
{
  float  a;
  double  b;
  a = 11111.166992;
  b = 11111.167456;
  printf(" % f\n % f\n",a,b);
}
```

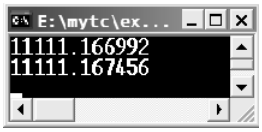

图 3-7 实例 3-5 的运行结果

运行结果如图 3-7 所示。从本例可以看出,由于 a 是单精度浮点型,有效位数只有 7 位。故对这个数来说,前 7 位是准确的,后 4 位是不准确的。b 是双精度型,有效位为 16 位,故输出结果是准确的。

3. 字符型变量

字符型变量的取值是字符常量,即单个字符。字符变量的类型说明符是 char,字符变量类型说明的格式和书写规则都与整型变量相同。

例如:

```
char a,b;
```

每个字符变量被分配一个字节的内存空间,因此只能存放一个字符。字符值是以 ASCII 码的形式存放在变量的内存单元之中的,字符 ASCII 码值的范围为 0～255。如 x 的十进制 ASCII 码是 120,y 的十进制 ASCII 码是 121。对字符变量 a,b 赋予'x'和'y'值:a='x'; b='y'; 实际上是在 a,b 两个单元内存放 120 和 121 的二进制代码。

```
a  →  0  1  1  1  1  0  0  0
b  →  0  1  1  1  1  0  0  1
```

所以也可以把它们看成是整型量。C 语言允许对整型变量赋以字符值,也允许对字符变量赋以整型值。在输出时,允许把字符变量按整型量输出,也允许把整型量按字符量输出。整型量为两字节量,字符量为一字节量,当整型量按字符型量处理时,只有低 8 位字节参与处理。

【实例 3-6】 字符变量示例 1。

```
void main()
{
  char  a,b;
  a = 100;
  b = 101;
  printf(" % c, % c\n % d, % d\n",a,b,a,b);
}
```

运行结果如图 3-8 所示。

图 3-8 实例 3-6 的运行结果

本程序中 a,b 说明为字符型变量,但在赋值语句中赋以整型值。从结果看,a,b 值的输出形式取决于 printf 函数格

式串中的格式符，当格式符为“%c”时，对应输出的变量值为字符，当格式符为“%d”时，对应输出的变量值为整数。

【实例 3-7】 字符变量示例 2。

```
void main()
{
    char a,b;
    a = 'c';
    b = 'd';
    a = a - 32;
    b = b - 32;
    printf(" % c, % c\n % d, % d\n",a,b,a,b);
}
```

运行结果如图 3-9 所示。

本例中，a，b 被说明为字符变量并赋予字符值，把小写字母换成大写字母以整型和字符型输出。C 语言允许字符变量参与数值运算，即用字符的 ASCII 码参与运算。由于大小写字母的 ASCII 码相差 32，因此运算后把小写字母换成大写字母，然后分别以整型和字符型输出。

图 3-9　实例 3-7 的运行结果

3.2.3　变量的地址——指针

指针是 C 语言中广泛使用的一种数据类型。运用指针编程是 C 语言最主要的风格之一。学习指针是学习 C 语言中最重要的一环。只有正确理解指针的基本概念，才能在程序设计中灵活运用。

1. 指针的基本概念

在计算机中，所有的数据都是存放在存储器中的。我们一般把存储器中的一个字节称为一个内存单元，不同的数据类型所占用的内存单元数不同，如整型量占两个单元，字符量占一个单元。为了正确地访问这些内存单元，必须为每个内存单元编上号，根据一个内存单元的编号即可准确地找到该内存单元。

内存单元的编号也叫做地址，通常也把这个地址称为指针。变量的指针和变量的内容是两个不同的概念。变量的指针就是变量的地址。在 C 语言中，允许用一个变量来存放指针，这种变量称为指针变量。因此，一个指针变量的值就是某个内存单元的地址或称为某变量的地址。

例如：

```
int a = 5,b = 0,c = 8;
```

变量 a，b，c 的内存表示如图 3-10 所示(注意：本图是以 TC 为例，VC 中略有不同，因在 TC 中，一个 int 整型数据占两个字节，在 VC 中一个 int 整型数据占 4 个字节)。图中的指针变量 p1、p2 和 p3 的值分别为 2001、2003 和 2005。

我们可以这样描述：指针变量 p1 的值是 2001，它存放变量 a 的地址，即指针。由它指

向的连续的两个内存单元中存放的是整数5；指针变量p2的值是2003，它存放变量b的地址，由它指向的连续的两个内存单元中存放的是整数0，也可以用赋值语句表示：p1＝&a；p2＝&b；p3＝&c。

其中，“&”是取变量地址运算符。“p1＝&a”表示把变量a的地址赋值到指针变量p1中。

在内存中，数据存放原则是“低地址存放低字节，高地址存放高字节”。

这里，请读者仔细领会变量的类型、变量的值、变量的地址这三个概念的含义。

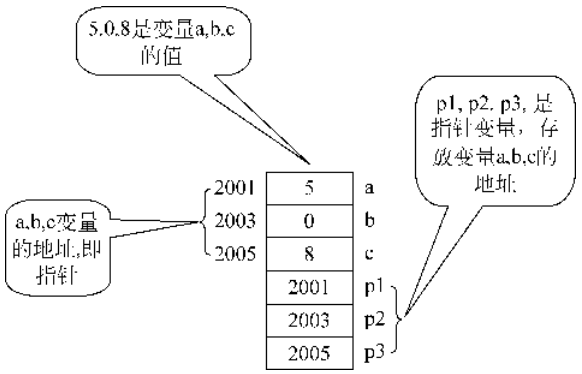

图3-10　变量的指针、变量的内容与指针变量示意图

2. 指针变量的定义

和普通变量一样，指针也要先定义后使用。其定义格式为：

```
类型说明符　*指针变量名;
```

其中，“*”表示这是一个指针变量，类型说明符表示本指针变量所指向的变量的数据类型。

例如int *p1；表示p1是一个指针变量，它的值是某个整型变量的地址。或者说p1指向一个整型变量。至于p1究竟指向哪一个整型变量，应由向p1赋予的地址来决定。

再如：

```
float *q1;                /*q1是指向浮点变量的指针变量*/
char *q2;                 /*q2是指向字符变量的指针变量*/
```

应该注意的是，一个指针变量只能指向同类型的变量，如q1只能指向浮点变量，不能时而指向一个浮点变量，时而又指向一个字符变量。

本章只是简单介绍了指针的基本概念，关于指针的具体用法，后续章节将逐渐介绍。

3.3　运算符和表达式

运算符用于控制运算对象，C语言很多操作都可以用运算符处理。用运算符将操作数连接起来，就构成了表达式。表达式的种类也很多，如算术表达式、逻辑表达式、关系表达式