第 3 章

CHAPTER 3

数据类型、运算符与表达式

数据是程序处理的对象，也是程序的必要组成部分。C 语言提供了丰富的数据类型，以便对现实中不同特性的数据加以描述。而运算是对数据的加工，C 语言提供了相当丰富的运算符和表达式，使程序的设计变得方便灵活。本章主要介绍 C 语言的基本元素、数据类型、运算符和表达式。

3.1 关键字、标识符和保留标识符

正如人类的自然语言具有其语法规则一样，C 语言也规定了它的语法。为了按照一定的语法规则构成 C 语言的各种成分（如常数、变量等），C 语言规定了基本词法单位。基本的词法单位是单词，而构成单词的最重要的形式是关键字、标识符和保留标识符。下面作简单介绍。

1．关键字

关键字是具有特定含义的、专门用来说明 C 语言的特定成分的一类单词。例如，关键字 int 用来定义整型变量，而关键字 float 则用来定义实型变量。C 语言的关键字都用小写字母书写，不能用大写字母书写。例如，关键字 int 不能写成 Int。由于关键字有特定的用途，所以不能用于其他场合，否则就会产生编译错误。下面是标准 C 语言中的 32 个关键字：

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			

2．标识符

在 C 语言中用于标识名字的有效字符序列称为标识符。C 语言对标识符有如下规定。

（1）标识符的第一个字符必须是英文字母或下划线（_）。

（2）如果第一个字符后面还有字符序列，则它应是英文字母、下划线或数字字符组成的序列。标识符中的英文字母大小写是有区别的，如标识符 abc 与标识符 ABC 不相同。

为了便于读者对标识符有进一步的认识，下面列举若干正确的标识符和不正确的标识符。

（1）正确的标识符：program、to、file_2、alb2c3。

（2）不正确的标识符：

- yes?（含有不合法字符“?”）；

- 2from（第一个字符不允许为数字）；
- yes no（标识符中不允许有空格）；
- yes / no（含有不合法字符“ / ”）；
- π r（π为不合法字符）。

标识符中有效字符个数（也称长度）视系统不同而不同。例如，Turbo C 规定前 32 个字符有效，超过的部分忽略。例如，对于 8 个字符有效的标识符而言，identifi 与 identifier 被视为同一标识符，因为后者中的 er 已被忽略。

以后将会看到，标识符用来为变量、符号常量、数组和函数等命名。使用时，标识符的选择由程序员自定，但是不能与关键字相同。另外，为了增加程序的可读性，选择标识符时应遵循“见名知义”的原则，即选择描述性的标识符，标识符应尽量与所要命名的对象有一定的联系，以助于识别和记忆。例如：

- length（表示长度）；
- time（表示时间）；
- pi（表示圆周率π）。

3．保留标识符

保留标识符是系统保留的一部分标识符，通常用于系统定义和标准库函数的名字，例如，以下划线开始的标识符通常用于定义系统变量。不应该用这些标识符来定义自己的变量，虽然它们也是合法的标识符，但是用它们来做一般标识符就可能会出问题。

3.2 数据与数据类型

数据是程序加工处理的对象，也是加工的结果，所以数据是程序设计所要涉及和描述的主要内容。程序所能处理的基本数据对象被划分成一些组，或者说是一些集合。属于同一集合的各数据对象都具有同样的性质，例如对它们能够做同样的操作，它们都采用同样的编码方式等。程序语言所具有这样性质的数据集合被称为数据类型。

计算机硬件也把被处理的数据分成一些类型，例如有定点数、浮点数等。CPU 对不同的数据类型提供了不同的操作命令，程序语言中把数据划分成不同类型与此有密切关系。但在程序语言中，类型的意义远不止于此。所有程序语言都由数据类型来描述程序中的数据结构、数据表示范围、数据在内存中的存储分配等。实际上，数据类型是计算机领域中的一个非常重要的概念。在学习程序设计的过程中，将要不断地与数据类型打交道，应该予以特别的关注。

C 语言中所用到的每一个常量、变量及函数等都是程序的基本操作对象，它们都隐式或显式地与一种数据类型相联系。每种数据类型都表明了它的可能取值范围及能在其上所进行的运算。

作为程序设计人员，在解决某一问题时，必须认真考虑和设计适应此问题的数据结构（即数据类型）和操作步骤（即算法）。这样，才能设计出正确、科学的程序。

例如，对 100 名学生的成绩排名次，显然要用到数组；求 10!，显然原始数据可用基本整型，而结果要放入长整型中。

C 语言的数据类型相当丰富，可分类如图 3.1 所示。

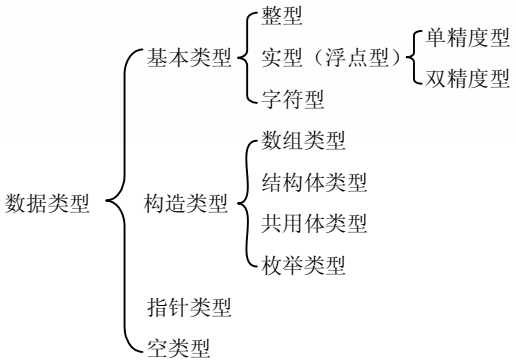

图 3.1　C 语言的数据类型

C 语言为基本类型数据定义了标识符，通常把它们称为类型名。例如，整型用 int 标识，字符型用 char 标识。类型名仅用于说明数据属于哪一种类型，它并不会在程序的另一处被引用。

C 语言的数据类型很丰富，它有基本数据类型，还有构造其他多种数据类型的能力。基本类型比较简单，构造类型一般由其他数据类型按照一定的规则构造而成，结构比较复杂，而指针类型是 C 语言中使用灵活、颇具特色的一种数据类型。本章主要介绍常用的基本数据类型。

3.3　基本数据类型及其表示

C 语言基本数据类型包括整型、实型和字符型。这些不同数据类型如何表示？如何使用？数据范围是多少？下面分别进行介绍。

3.3.1　常量与变量

1．常量和符号常量

在程序运行过程中，其值不能被改变的量称为常量。常量分为不同的类型，C 语言提供的基本常量有整型常量、实型常量、字符常量和字符串常量。如 12、0、－3 为整型常量，4.6、－1.23 为实型常量，'a'、'd'为字符常量，"china"、"hello"为字符串常量。为了增加程序的可读性，可以用一个名字（字符序列）来代表一个常量，称为符号常量。在程序中需要使用该常量时就可直接引用表示该常量的标识符。

C 语言中用宏定义命令对符号常量进行定义，其定义形式如下：

```
#define 标识符 常量
```

其中，#define 是宏定义命令的专用定义符，标识符是对常量的命名，常量可以是前面介绍的几种类型常量中的任何一种。该命令的含义是使用指定的标识符来代表指定的常量，这个被指定的标识符就称为符号常量。例如，在 C 程序中，要用 PI 代表 3.1415926，可用下面这个宏定义命令：

```
#define PI 3.1415926
```

【例 3.1】 已知产品的数量和单价，求总金额。

```
#include "stdio.h"
#define PRICE 30
void main()
{int num,total;
num=10;
total=num*PRICE;
printf("total=%d",total);
}
```

程序中用#define 命令定义 PRICE 代表常量 30，此后凡是在程序中出现的 PRICE 都代表 30，可以和常量一样进行运算。但必须注意符号常量不同于变量，它的值在其作用域（本例中为主函数）内不能改变，也不能再被赋值。一般，符号常量名用大写，变量名用小写，以示区别。

2．变量

其值可以改变的量称为变量。程序中的变量可以看成是一个存储数据的容器，它的功能就是可以存储数据。对变量的基本操作有两个：向变量中存入数据值，这个操作被称为给变量“赋值”；取得变量当前值，以便在程序运行过程中使用，这个操作称为“取值”。变量具有保持值的性质，也就是说，若某个时刻给某变量赋了一个值，此后使用这个变量时，每次得到的将总是这个值。

因为要对变量进行“赋值”和“取值”操作，所以程序中使用的每个变量都要有一个变量名，程序是通过变量名来使用变量的。在 C 语言中，变量名是作为变量的标识，其命名规则符合标识符的所有规定。

在 C 语言中，字母大小写有区别。例如，RAN、Ran、ran 是 3 个不同的变量。

组成变量名（标识符）的有效字符个数随 C 语言的编译系统而定。有的编译系统允许使用长达 31 个字符的变量名，而有的编译系统只取变量名的前 8 个字符作为有效字符，后面的字符无效，不被识别，这样，只要变量名的前 8 个字符相同，就被认为是同一个变量。因此，在进行程序设计之前，应首先了解所使用的编译系统对变量名长度的规定，以免造成变量使用上的混乱。

在选择变量名和标识符时，应注意做到“见名知义”，并且对所使用的变量在使用之前，必须先对其进行声明。声明变量包括为变量取名和指定变量类型。例如，下面声明两个变量 i 和 j 为实型变量：

```
float i,j;
```

其中，float 是类型名关键字，指明其后变量的类型是实型。通常，相同类型的变量共用一个类型名关键字，也可以分别表示。如：

```
float i;
float j;
```

但是，不允许在程序的同一处将同一变量声明为不同类型。如下面的声明是错误的：

```
int i;
float i;
```

C 语言中每一数据类型都有与之对应的类型名关键字。

3.3.2　整型数据、实型数据和字符型数据

1．整型数据

1）整型常量

整型常量可以用三种数制来表示。

（1）十进制整型常量：如 250、－12 等，其每个数字位可以是 0～9。

（2）十六进制整型常量：以 0x 或 0X 开头，如十进制数的 128，用十六进制表示为 0x80，其每个数字位可以是 0～9 和 A～F（或 a～f）。

（3）八进制整型常量：以 0 开头，如十进制数的 128，用八进制表示为 0200。其每个数字位可以是 0～7。

以上是整型常量的表示，对于长整型常量同样可以用十进制、八进制和十六进制 3 种形式表示。其表示形式是在常量之后加上字母 l 或 L。例如，123L、0xaeL 和 0171 都是长整型常量。

2）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。无符号型又分为无符号整型、无符号短整型和无符号长整型。整型变量以关键字 int 作为基本类型说明符，另外配合 4 个类型修饰符，用来改变和扩充基本类型的含义，以适应更灵活的应用。各种类型可表示如下。

（1）基本型：以 int 表示。

（2）短整型：以 short int 或 short 表示。

（3）长整型：以 long int 或 long 表示。

（4）无符号整型：以 unsigned int 或 unsigned 表示。

（5）无符号短整型：以 unsigned short 表示。

（6）无符号长整型：以 unsigned long 表示。

以 IBM PC、Turbo C 编译系统为例，整型数据所占的内存字节数以及数的表示范围如表 3.1 所示，可根据数的范围来选用。

表 3.1　整型数据所占位数与表示范围

数据类型	所占位数	数的范围
int	16	－32 768～32 767
short [int]	16	－32 768～32 767
long [int]	32	－2 147 483 648～2 147 483 647
unsigned [int]	16	0～65 535
unsigned short	16	0～65 535
unsigned long	32	0～4 294 967 295

整型变量的定义如下所示：

```
int a,b;              /*指定变量 a,b 为整型*/
unsigned short c,d; /*指定变量 c,d 为无符号短整型*/
```

```
long e,f;            /*指定变量 e,f 为长整型*/
```

在一个函数内部，应该首先将所用到的变量集中起来进行说明，然后才是函数的执行语句部分。

【例 3.2】

```
#include "stdio.h"
   void main()
   {int a,b,c,d;
   unsigned u;
   a=12;b=－24;u=10;
   c=a+u;d=b+u;
   printf("a+u=%d,b+u=%d\n",c,d);
   }
```

运行结果为：

```
a+u=22,b+u=－14
```

给变量赋值时应注意所赋值与被赋值的变量类型匹配问题。

还要注意以下几点。

（1）一个整型常量，如果其值在－32 768～32 767 范围内，认为它是 int 型，它可以赋值给 int 型和 long int 型变量。

（2）一个整型常量，如果其值超过了上述范围，而在－2 147 483 648～2 147 483 647 范围内，则认为它是 long int 型。可以将它赋值给一个 long int 型变量。

（3）如果某一计算机系统的 C 版本确定的 short int 与 int 型数据在内存中占据的字节数相同，则它的表示范围与 int 型相同。因此一个 int 型的变量也同时是一个 short int 型变量，int 型与 short int 型变量间可以相互赋值。

（4）常量中无 unsigned 型。但一个非负值的整型常量可以赋值给 unsigned 整型变量，只要它的范围不超过变量的表示范围即可。例如，将 50 000 赋给一个 unsigned int 型变量是可以的，而将 70 000 赋给它是不行的（溢出）。

（5）在一个整型常量后面加一个字母 l 或 L，则认为是 long int 型常量。例如，123l、432L、0L 等。这往往用于函数调用中。如果函数的形参为 long int 型，则要求实参也为 long int 型，此时用 123 作实参不行，而要用 123L 作实参。

（6）数据溢出问题。

【例 3.3】

```
#include "stdio.h"
   void main()
   { int a,b,c;
     a=32767;
     b=3;
     c=a+b;
     printf("c=%d\n",c);
    }
```

该程序的运行结果应该是 c＝32 770，而实际 Turbo C 环境下的运行结果是 c＝－32 766。

显然这个结果是错误的，但系统没有提示出错。为什么会出现这种情况呢？分析这个程序，a 和 b 的值都没有超出整型数的表示范围，而 a+b 后应得到 32 770，这个数已经超出了整数的表示范围，称为溢出。但这种溢出在内存变量 c 中的表现形式正好是数值－32 766 的补码形式，所以输出变量 c 的内容时自然就输出了－32 766。

如果把上述程序做以下修改：

```
#include "stdio.h"
void main()
{ long a,b,c;
  a=32767;
  b=3;
  c=a+b;
  printf("c=%ld\n",c);
 }
```

即把变量 a、b、c 定义成长整型，就可以得到正确的运行结果。

思考：如果只把 c 定义成长整型，a 和 b 还保持整型，结果会怎样？

注意：在 VC++环境中调试例 3.3，得到结果是 32 770，没有溢出现象，这是因为在 Visual C++环境中 int 类型数据分配 4 个字节，所以不会溢出。

2．实型数据

1）实型常量

实型常量在 C 语言中又称浮点数或实型数。在 C 语言中，实型常量一般都作为双精度数来处理，并且只用十进制数表示。实型常量有如下两种表示形式。

（1）小数形式：由符号、整数部分、小数点及小数部分组成。 0.123、.123、123.0、123.和 0.0 都是十进制数形式。其中，小数点是不可缺少的。例如，123. 不能写成 123，因为 123 是整型常量，而 123.是实型常量。

（2）指数形式：由十进制小数形式加上指数部分组成，其形式如下：

十进制小数 e 指数部分

或

十进制小数 E 指数部分

如 123e3 或 123E3 都代表 123×10^3。但注意字母 e（或 E）之前必须有数字，且 e 后面指数必须为整数，如 e3、2.1e3.5、e3 和 e 等都不是合法的指数形式。

对于上述两种书写形式，系统均默认是双精度实型常量，可表示 15～16 位有效数字。数的表示范围可达到 10^{-308} ~ 10^{308}。如果要表示单精度实型常量和长双精度实型常量，只要在上述书写形式后加上后缀 f（F）或 l（L）即可。例如：

2.3f、－0.123F、2e－3f 为合法的单精度常量，有 7 位有效数字。

1234.9L、－0.123L、2e3L 为合法的长双精度常量，有 18～19 位有效数字。

2）实型变量

C 实型变量分单精度、双精度和长双精度三种。ANSI C 标准允许定义三种实型变量的

关键字如下：

- float（单精度）；
- double（双精度）；
- long double（长双精度）。

对每一个实型变量都应在使用前加以定义。如：

```
float x,y;          /*指定变量 x, y 为单精度实数*/
double z;           /*指定变量 z 为双精度实数*/
long double e,f;    /*指定变量 e,f 为长双精度实数*/
```

在一般系统中，一个 float 型数据在内存中占 4 个字节，一个 double 型数据占 8 个字节，一个 long double 型数据占 16 个字节。单精度实数提供 7 位有效数字，双精度实数提供 15～16 位有效数字，长双精度实数提供 18～19 位有效数字。数值的范围随机器系统而异。在IBM PC MS-C 中，单精度实数的数值范围为 10^{-38} ～10^{38}，双精度实数范围为 10^{-308}～10^{308}，长双精度实数范围为 10^{-4931}～10^{4932}。

一个实型常量可以赋值给一个 float 型或 double 型变量。根据变量的类型截取实型常量中相应的有效数字位。假如 a 已指定为单精度实型变量：

```
float  a;
a=111111.111f;
```

由于 float 型变量只能接收 7 位有效数字，因此最后两位小数不起作用。如果 a 改为 double 型，则能全部接收上述 9 位数字并存储在变量 a 中。

3．字符型数据

1）字符常量

字符常量是用单引号括起来的一个字符，如'a'、'D'、'?'和'$'等。

除了以上形式的字符常量外，对于一些常用的但却难以用一般形式表示的不可显示字符，C 语言提供了一些特殊形式的字符常量，这些字符常量都是以反斜杠（\）开头的字符序列。常用的特殊字符如表 3.2 所示。

表 3.2　转义字符

字 符 形 式	功　　能
\n	换行，将当前位置移到下一行开头
\t	横向跳格（即跳到下一个输出区）
\v	竖向跳格
\b	退格，将当前位置移到前一列
\r	回车，将当前位置移到本行开头
\f	走纸换页，将当前位置移到下页开头
\\	反斜杠字符“\”
\'	单引号字符
\"	双引号字符
\0	空操作字符（ASCII 码为 0）
\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

表 3.2 中列出的字符称为“转义字符”，意思是将反斜杠（\）后面的字符转变成另外的意义。如'\ n'中的 n 不代表字母 n 而作为“换行符”。

可在\后面紧跟1～3位八进制数或在'\x'后面紧跟1～2位十六进制数来表示相应系统中所使用的字符的编码值。使用这种表示方法，可以表示字符集中的任一字符，包括某些难以输入和显示的控制字符，ASCII 码表中编码值小于 0x20 的字符就属于这一类字符。如响铃字符（beep）在 ASCII 码表中的编码值为 7，在程序处理过程中，为了发出响铃声音，可通过'\007'码来获得响铃效果。

需要注意的是，上面介绍的由\开头的特殊字符，仅代表单个字符，而不代表多个字符，它仅代表相应系统中的一个编码值。如'\x3a'代表 ASCII 码值为 58 的字符。

在实际应用中，转义字符的使用很多，例如有以下程序行：

```
printf("a=%f,b=%f\n",a,b);
```

其中'\n'就是换行转义字符。几乎每个程序中都会有一个或若干个这样的程序行。

2）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，如"string"。从表面上看，"string"是由 6 个字符组成的，但实际上它是由 7 个字符组成的，因为在 C 语言中，系统自动地在每个由双引号括起来的字符串的最后补上'\0'字符，即 ASCII 码值为 0 的字符。因此，C 语言中的每个字符串都是以'\0'为结束标志的。对字符串操作时，这个结束标志很重要。例如，输出字符串时，遇到这个结束标志才终止输出。

可见，字符常量与字符串常量的区别有两个方面：从形式上看，字符常量是用单引号括起来的一个字符，字符串常量是由双引号括起来的字符序列；从存储方式看，字符常量在内存中占一个字节，而字符串常量除了每个字符各占一个字节外，其字符串结束符'\0'也要占一个字节。例如，字符常量'A'占一个字节，而字符串常量"A"占两个字节。

3）字符变量

字符型变量用来存放单个字符，定义形式如下：

```
char   c1, c2;
```

可对 c1、c2 赋值，c1='a'，c2='b'。注意，不能将字符串常量赋给一个字符变量。

【例 3.4】

```
#include "stdio.h"
void main()
{char c1,c2;
c1='a';c2=98;
printf("%c  %c",c1,c2);
}
```

运行结果为：

```
 a     b
```

程序中第 3 行相当于以下两个赋值语句：

```
c1='a';c2='b';
```

4．变量的初始化

通常一个变量是先声明，然后再赋值，例如：

```
int x，y;
x=10,y=20;
```

在 C 语言中，也可以对变量进行初始化，即允许在声明变量的同时对变量赋初值。所以上面可改为：

```
int x=10，y=20;
```

所以变量赋值具有两种形式，一种是先声明后赋值，另一种是在声明变量的同时对变量赋初值，这就是变量的初始化。所有类型的变量都可以初始化，例如：

```
float  x=123.45f;              /*声明 x 为实型量，且赋初值为 123.45*/
int  a，b，c=10;               /*类型声明语句中给部分变量赋初值,即仅给 c 赋初值 10*/
int  a1=10，b1=6，c1=10;       /*声明整型变量 a1、b1、c1，并分别赋不同的初值*/
double  pi=3.14;               /*声明 pi 为双精度实型变量，且赋初值为 3.14*/
char  ch='a';                  /*声明字符变量 ch，并赋初值为'a'*/
```

以上为 x、c、a1、b1、c1、pi 和 ch 都赋了初值，它与用赋值语句赋值有同样的效果。

若有几个变量值相同，也必须一一进行初始化。例如：

```
int a=10,b=10,c=10;
```

写成下列形式则是错误的。

```
int a=b=c=10;
```

变量初始化不是在程序编译时完成的（除后面介绍的外部变量和静态变量），而是在程序运行时对变量赋初值。

5．不同类型数据之间的转换

字符型数据在内存中是以 ASCII 码形式存储的，与整数的存储形式类似。因此字符型数据和整型数据之间可以通用。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，同时，字符型数据可以赋给整型变量，整型数据也可以赋给字符型变量，只是当整型数据的大小超过字符型变量的表示范围时，需要截取相应的有效位数。

除了上述字符型数据和整型数据之间可以通用之外，不同类型的数据在进行混合运算时，往往需要进行类型转换。类型转换有两种方式：自动类型转换和强制类型转换。

1）自动类型转换

C 语言允许在整型、单精度浮点型和双精度浮点型数据之间进行混合运算，由于字符型数据可以和整型数据通用，所以，下列表达式是合法的。

```
100+'A'+3.6*27
```

显然，在进行混合运算时，不同类型的数据要首先转换成同一类型，然后才能进行运算，转换的规则如图 3.2 所示。