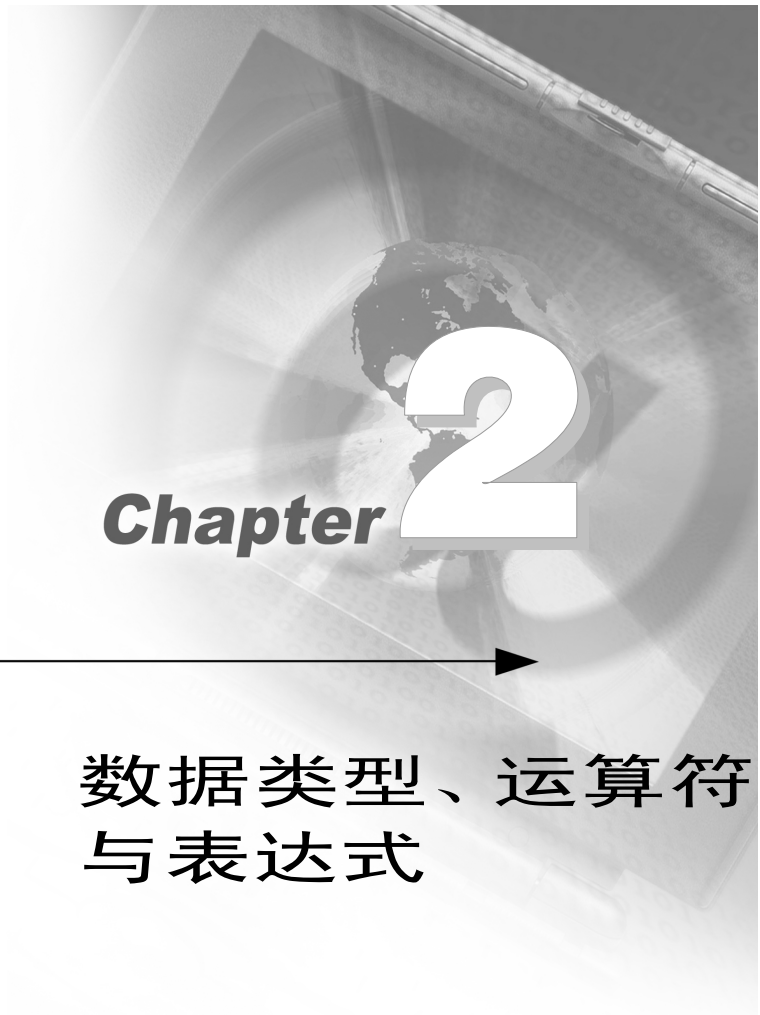

第 2 章　数据类型、运算符与表达式

在计算机的使用中，数据的存储和处理是必不可少的。C 语言为用户提供了各种标准的数据（基本类型）类型，也允许用户自己定义新的数据类型，本章介绍 C 语言的标准数据类型、指针类型和相关运算。

2.1　C 语言的数据类型

众所周知，计算机内使用二进制数来存放各种信息，比如图像、字符、音乐等，那么计算机是如何区分这些信息的？这主要取决于计算机如何解释这些二进制数据。比如 65 解释为数字是 65，而解释为字符则是“A”，这种不同的解释是人们对信息存放做出的规定，也就是数据的组织形式。

C 语言中是如何规定数据的存放形式的呢？C 语言的数据类型如图 2.1 所示，分为基本数据类型、构造数据类型、指针类型和空类型。

本章讲解基本数据类型中的整型、实型、字符型和指针类型，其余数据类型在以后各章中将陆续介绍。

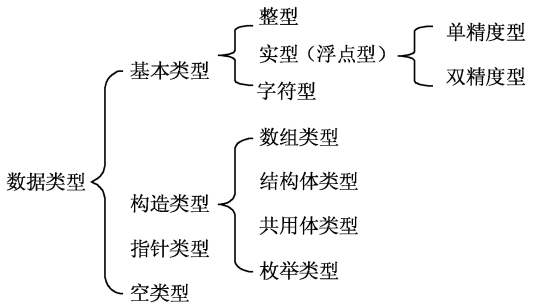

图 2.1　C 语言的数据类型

2.2　常量与变量

C 语言对常量和变量的定义是，在程序运行过程中，其值不会发生变化的量为常量；在程序运行过程中，其值可以发生变化的量为变量。

给变量所取的名字叫变量名，给变量取名时要遵循标识符的命名规则。所谓标识符是用来标识变量、符号常量、数组、函数、文件等名字的有效字符序列。

C 语言规定标识符只能由字母、数字和下画线三种字符组成，并且第一个字符必须为字母或下画线。例如：sum，day，_class，student_No，a123 等都是合法的标识符。

需要注意的是：

（1）用户选用的标识符不能和 C 语言的关键字重名。如 if（C 语言的关键字），main（C 语言的关键字）都是不合法的标识符。

（2）在 C 语言中，大写字母和小写字母被认为是两个不同的字符，如 max 和 MAX 是两个不同的标识符。

（3）标识符的长度在不同的 C 编译系统中有不同的规定。许多系统规定前 8 个字符有效，而在 Turbo C 中规定前 32 个字符有效，超过部分被忽略。

2.2.1　直接常量和符号常量

1．直接常量（字面常量）

【例 2.1】　已知每千克牛肉的价格为 20 元，问买 6 斤需要多少钱？

```
#include <stdio.h>
main( )
{
    float sum;                    /*变量定义*/
    sum=20.0*6;                   /*给变量赋值*/
    printf("总价=%f \n",sum);     /*输出*/
}
```

程序运行结果如下：

总价=120.000000

显而易见，程序中的 20.0 和 6 都是常量，按其字面形式又可区分为不同的类型，20.0 是实型常量，6 是整型常量。

2．符号常量

所谓符号常量，就是用一个标识符来代表一个常量。

【例 2.2】　符号常量的使用。

将例 2.1 改写如下：

```
#include <stdio.h>
#define PRICE 20                    /*宏定义语句*/
main()
{
    float num;
    float sum;                      /*变量定义*/
    num=6.0;
    sum=num* PRICE;
    printf("总价=%f",sum);          /*输出*/
}
```

程序运行的结果仍然为：

```
总价=120.000000
```

程序中用标识符 PRICE 来代表常量 20，PRICE 就是一个符号常量。为了与变量区别，习惯上符号常量用大写字母表示，变量名用小写字母表示。这并不是 C 语言的规定，仅仅是一种习惯，目的是便于程序的阅读。

符号常量在使用之前必须先定义，其一般形式为：

```
#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（在第 6 章中将详细介绍），其作用是把该标识符定义为其后的常量值。一经定义，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。

常量的值在其作用域内不能改变，也不能再被赋值。如：

```
#define PRICE 20
main()
{    ...
    PRICE=25;                   /*这里试图改变符号常量 PRICE 的值*/
    ...
}
```

程序在编译时，系统会给出错误的提示信息。

3．使用符号常量的好处

使用符号常量有以下优点：

（1）阅读程序方便。在程序设计中，如果我们直接给出常量，阅读程序的人很难一下看出各常量的含义；而使用符号常量，可以起到“见名识义”的效果，提高程序的可读性。

（2）修改程序方便。程序中可能出现多处使用同一个常量的情况，如果需要修改该常量，程序员修改程序的操作烦琐而且容易遗漏，导致程序运行结果出错。当使用符号常量时只须修改定义处即可，做到“一改全改”。

在例 2.2 中，如果牛肉单价上涨，由每千克 20 元上涨为 25 元，只须把常量定义语句修改为：

```
# define   PRICE   25
```

这样，程序中所有 PRICE 全都改为 25 了。

2.2.2 变量

每一个变量都必须有一个名字，在程序中才能对其进行操作，变量名应该是合法的 C 标识符。每个变量在内存中占有一定的存储单元，在存储单元中存放的是该变量的值。

注意，变量名和变量值是两个不同的概念。变量名实际上是一个符号地址，即变量在内存中的存放位置。程序运行过程中从变量取值，实际上是通过变量名找到相应的内存地址，从存储单元读取数据；而对变量的赋值操作也是通过变量名找到相应的内存地址，然后将数据写入存储单元中。

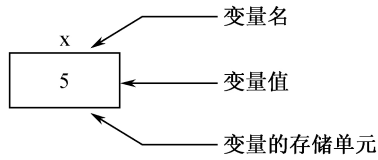

图 2.2　变量名与变量值

不同类型的变量在内存中占据存储单元数量及存储的格式也不相同。C 语言要求对所有用到的变量进行强制性的定义，即对变量要“先定义，后使用”，这样做的目的主要体现在：

第一，保证程序中变量名的正确使用，凡是未被定义的不能用做变量名。

如有以下程序：

```
#include <stdio.h>
main( )
{
    int a=45,b=32,sum=0;
    svm=a+b;
    printf("sum=%d",sum);
}
```

程序的第 5 行错将 sum 写为 svm，在编译时，系统会报告 svm 未定义，并给出提示“undeclared identifier”。如果 C 语言中没有对变量做强制定义的要求，程序可以执行，但结果为 0，这样的错误很难被发现。

第二，变量定义时被指定为某一数据类型，在编译时就能为其分配相应的存储单元。如在 Turbo C 中，为 int 类型变量分配 2 字节的存储单元，为 float 类型分配 4 字节的存储单元。

第三，定义时指定变量的类型，便于在编译程序时检查对该变量的运算是否合法。如 int 型变量可以进行求余运算，而 float 型变量不允许进行求余运算。

2.3 整型数据

2.3.1 整型常量

整型常量就是整常数。C 语言中，整型常量可以用以下三种形式表示。

（1）十进制整常数：就是通常整数的写法，其数码为 0～9。

例如：123、−5、+256 等。

（2）八进制整常数：八进制整常数必须以数字 0 开头，即以 0 作为八进制数的前缀。数码取值为 0～7。

以下各数都是合法的八进制数：

015（十进制为 13）、0101（十进制为 65）、0177777（十进制为 65535）；

以下各数不是合法的八进制数：

256（无前缀 0）、0392（包含了非八进制数码 9）。

（3）十六进制整常数：十六进制整常数必须以数字 0 和字母 X（或 x）开头，即 0X 或 0x，其数码取值为数字 0～9 和字母 A～F（或 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）、0XFFFF（十进制为 65535）；

以下各数不是合法的十六进制整常数：

5A（无前缀 0X）、0X3H（含有非十六进制数码）。

注意：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混淆前缀，否则会造成结果不正确。

2.3.2 整型变量

1．整型数据在内存中的存放形式

计算机内部数据是以二进制的形式存放在内存中的。例如定义了一个整型变量 i，并给变量赋初值 12。

```
int i;
i=12;
```

在 Turbo C 编译系统中，每个整型变量在内存中占两个字节，其中最高位为符号位，整数符号位为 0，负数符号位为 1。

十进制数 12 的二进制形式为 1100，在内存中实际存放的情况如下：

0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0

实际上，在计算机中，为了方便计算，整数值是以补码表示的。一个正整数的补码就是其二进制表示形式，而负整数补码的求解方法是：将该数的绝对值的二进制形式按位取反后再加 1。

例如，求−12 的补码：

（1）−12 的绝对值为 12。

（2）12 的二进制形式为：

0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0

（3）按位取反：

1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1

（4）再加 1，结果为：

1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	0

可以看出，左面的第一位就是符号位。

2．整型变量的分类

整型的类型符是 int，C 语言提供以下 4 种整数类型。

（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）无符号型：存储单元中所有二进制位都用来表示数值，没有符号位。

无符号型又可与上述三种类型匹配而构成：

① 无符号整型，以 unsigned int 或 unsigned 表示。

② 无符号短整型，以 unsigned short 表示。

③ 无符号长整型，以 unsigned long 表示。

如果不指定为无符号型，则默认为有符号型（signed）。无符号型变量只能存放不带符号的整数，而不能存放负整数。

表 2.1 列出了 Turbo C 中各类整型量在内存中所占的字节数及数的表示范围。

表 2.1　整型类型符

类型说明符	数值范围		占用字节数
int	−32768~32767	即-2^{15}～($2^{15}-1$)	2
unsigned	0~65535	即 0～($2^{16}-1$)	2
short	−32768~32767	即-2^{15}～($2^{15}-1$)	2
unsigned short	0~65535	即 0～($2^{16}-1$)	2
long	−2147483648~2147483647	即-2^{31}～($2^{31}-1$)	4
unsigned long	0~4294967295	即 0～($2^{32}-1$)	4

下面以整数 12 为例，我们来看看 12 被定义为不同的类型在内存中的存放形式。

int 型：

00	00	00	00	00	00	11	00

short int 型：

00	00	00	00	00	00	11	00

long int 型：

00	00	00	00	00	00	00	00	00	00	00	00	00	00	11	00

unsigned int 型：

00	00	00	00	00	00	11	00

unsigned short int 型：

00	00	00	00	00	00	11	00

unsigned long int 型：

00	00	00	00	00	00	00	00	00	00	00	00	00	00	11	00

3．整型变量的定义及初始化

C 语言程序中有多种方法为变量提供初值，在定义变量的同时给变量赋初值的方法称为初始化。

变量定义及初始化的一般形式：

类型说明符　变量 1[=值 1]，变量 2[=值 2]，……；

例如：

```
int    a, b, c;              /*  定义 a,b,c 为整型变量*/
long   x=15 ;                /*  定义 x 为长整型变量，且赋初值 15 */
unsigned   p=3, q;           / *p,q 为无符号整型变量，为 p 赋初值 3 */
```

变量定义时，需要注意以下几点：

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

（2）最后一个变量名之后必须以“;”号结尾。

（3）变量定义必须放在变量使用之前，一般放在函数体的开头部分。

【例 2.3】　整型变量的定义与使用。

```
#include <stdio.h>
main( )
{
    int a=12,b= −24,c,d,u=10;                /*定义 a,b,c,d,u 为整型变量*/
    c=a+u;
    d=b+u;
    printf("a+u=%d, b+u=%d\n",c,d);          /*输出语句*/
}
```

运行程序，结果为：

```
a+u=22, b+u=−14
```

4．整型数据的溢出

当数据超出所定义的数据类型范围时，就会产生溢出。我们先看下例。

【例 2.4】　整型数据的溢出。

```
main()
{
    int a,b;
    a=30000;
```

```
    b=a+20000;
    printf("a=%d,b=%d\n",a,b);
}
```

在 Turbo C 中运行程序，结果为：

```
a=30000,b=-15536
```

读者一定会认为变量 b 的值应该是 50000，怎么会是-15536 呢？这是因为整型数据表示的数值范围最大是 32767，变量 b 的值已经超出了该范围，产生溢出，得不到正确结果。

【例 2.5】 无符号整型数据的溢出。

```
main()
{
    int a,b;
    a=65535;
    b=a+100;
    printf("a=%u,b=%u\n",a,b);
}
```

在 Turbo C 中运行程序，结果为：

```
a=65535,b=99
```

请读者自己分析出现这种结果的原因。

程序中如何避免整数的溢出？应该根据具体情况将整数相应地表示为长整型、无符号型或无符号长整型。

需要说明的是，例 2.4 和例 2.5 如果在 VC 环境下运行是不会产生溢出的。因为在 C++中，没有规定为每种整型变量分配的内存空间，每种整数类型的取值范围都取决于编译器。

5．整型常数的后缀

对于在基本整型表示范围内的常量，要说明其是长整型数，可以用字母“L”或“l”作为后缀来表示。

例如：158L、358000L、012L、0X15L 等。

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，Turbo C 编译系统将为它分配 4 字节的存储空间。而对 158，因为是基本整型，仅分配 2 字节的存储空间。要注意两者之间的区别。

同样，无符号数也可用后缀表示，整型常数的无符号数的后缀为字母“U”或“u”。

例如：358u、0x38Au、235Lu 均为无符号数。

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型数。

实训 2　使用整型数据

1．实训目的

掌握整型数据，选择合适的整型变量存放数据。

2．实训内容

（1）编写程序，求圆的面积。

```
#include <stdio.h>
#define PI 3.14159                    /*  定义符号常量 PI*/
main()
{
      float s;
      int r=2;
      s=PI*r*r;
      printf("s=%f\n",s);
}
```

运行程序，输出结果为：

```
s=12.566360
```

在实际应用中，应根据需要选择适当的数据类型，以保证程序的正确。

（2）某校二年级共有五个班级，人数分别为 41、39、37、40、42。求二年级的总人数。

```
#include <stdio.h>
main()
{
     unsigned a,b,c,d,e,sum;
     a=41;b=39;c=37;d=40;e=42;
     sum=a+b+c+d+e;
     printf("总人数=%u",sum);
}
```

程序中将班级人数用 a,b,c,d,e 五个变量来存放，sum 用来存放年级的总人数，由于人数不可能为负数，所以把上述变量均定义为无符号整型。

（3）某校发动学生义务植树，预计植树 50 000 株。由于天气原因，实际植树 35 126 珠，求实际植树株数和预计植树株数的差值。

```
#include <stdio.h>
main()
{
      long y,s,c;                     /*y 为预计植树株数，s 为实际植树株数，c 为差值*/
      y=50000;s=35126;
      c= y−s;
      printf("差值=%ld",c);
}
```

程序中把预计植树数、实际植树数和差值都定义为长整型，因为这些数值超出了基本整型的表示范围。

3．实训思考

在实际应用中，选择合适的数据类型是必要的，为什么？

2.4 实型数据

2.4.1 实型常量的表示方法

实型也称为浮点型，实型常量也称为实数或者浮点数。在 C 语言中，实数只使用十进制表示，它有两种形式：一般的小数形式和指数形式。

（1）一般的小数形式：由数码 0～9 和小数点组成。

例如：0.0、25.0、5.789、0.13、300.、−267.8230 等均为合法的实型常量。

注意，实型常量中必须含有小数点。

（2）指数形式：由十进制数，加阶码标志小写字母“e”或大写字母“E”以及阶码（只能为整数，可以带符号）组成。

例如：2.1E5（等于 2.1×10^{5}）、3.7E−2（等于 3.7×10^{-2}）、−2.8E−2（等于-2.8×10^{-2}）均为合法的实型常量。

以下不是合法的实型常量：

345（无小数点）

E7（阶码标志 E 之前无数字）

53. −E3（负号的位置不对）

2.7E（无阶码）

在 Turbo C 中，实型常数不分单、双精度，都按双精度（double）型处理。但可以添加后缀“f”或“F”，即表示该数为单精度浮点数。如 128f 和 128.0 是等价的。

【例 2.6】 实型常量举例。

```
main()
{
    printf("%f\n ",128.);
    printf("%f\n ",128);
    printf("%f\n ",128f);
}
```

在 Turbo C 中运行程序，结果为：

```
128.000000
0.000000
128.000000
```

由于第二行语句中 128 不是一个实型常量，所以其输出为 0.000000。

2.4.2 实型变量

1. 实型数据在内存中的存放形式

在 Turbo C 中实型数据占 4 字节（32 位）的内存空间，按指数形式存储。

例如，实数 3.14159 在内存中的存放形式如下：

+	.314159	+	1
数符	小数部分	指符	指数

说明：小数部分占的位（bit）数越多，数的有效数字就越多，精度越高。指数部分占的位数越多，则能表示的数值范围就越大。

2．实型变量的分类

在 Turbo C 中实型变量分为三种：单精度型（float）、双精度型（double）和长双精度型（long double）。在实际应用中，长双精度型用得比较少。

各种实数类型在内存中所占的字节数及数的表示范围如表 2.2 所示。

表 2.2　实数类型符

类型说明符	所占的字节数	有效数字	数值范围
float	4	6~7	−3.4E38~−3.4E−38、0、3.4E−38~3.4E38
double	8	15~16	−1.7E308~−1.7E−308、0、1.7E−308~1.7E308
long double	16	18~19	−1.2E4932~−1.2E−4932、0、1.2E−4932~1.2E4932

3．实型变量的定义及初始化

实型变量的定义及初始化形式与整型变量类似。

例如：

```
double a,b,c;              /*  定义 a,b,c 为双精度实型变量*/
float x=1.2 , y=3.5 ;      /*定义 x 为单精度实型变量，且初值为 1.2 */
                           /*定义 y 为单精度实型变量，初值为 3.5*/
```

4．实型数据的舍入误差

由于实型变量是由有限的存储单元组成的，因此能提供的有效数字也是有限的。我们先看一个例子。

【例 2.7】　实型数据的舍入误差。

```
#include<stdio.h>
main()
{
    float a=1.234567E10, b ;
    b=a+20;
    printf("a=%f\n", a);
    printf("b=%f\n", b);
}
```

运行程序，输出结果为：

```
a=12345669632.000000
b=12345669632.000000
```

对于上述结果读者可能要质疑，变量 b 比变量 a 要大 20，怎么显示结果都是 12345669632.000000 呢？这是因为变量 a 是浮点数，尾数只能保留 6～7 位有效数字，变量 b 所加 20 被舍弃。因此在进行计算时，要避免一个大数和一个小数直接相加/减。

注意：如果实型数据（float）的运算超出了所表示的最大范围，也会产生溢出，这时可以把数据定义为 double 类型或者 long double 类型。

实训 3　使用实型数据

1．实训目的

正确书写实型常量，选择合适的实型变量存放数据。

2．实训内容

（1）已知三角形的底为 2.8cm,高为 4.3cm，求三角形的面积。
程序如下：

```
#include<stdio.h>
main()
{
    float   d=2.8, h=4.3, s;
    s=d*h/2;
    printf("s=%f",s);
}
```

运行程序，输出结果为：

```
s=6.020000
```

（2）编写程序将摄氏温度 27.5 度转换为华氏温度显示。
转换公式为：$c=\frac{5}{9}(f-32)$

```
#include<stdio.h>
main()
{
    float f=27.5,c;
    c=5.0/9*(f-32);
    printf("c=%f",c);
}
```

运行程序，输出结果为：

```
c=-2.500000
```

2.5　字符型数据

2.5.1　字符常量

字符常量是用一对单引号括起来的一个字符。例如'a'、'B'、'='、'+'、'？'都是字符常量。在 C 语言中，字符常量有以下特点：

第一，字符常量只能用单引号括起来，不能用双引号或其他符号括起来。

第二，字符常量只能是单个字符，不能是多个字符。

第三，字符可以是字符集中的任意字符。

字符集是一套允许使用的字符的集合，在中小型计算机和微型机上广泛采用的是 ASCII 字符集。常用字符与 ASCII 码对照表见附录 A。但是数字被定义为字符型时，其含义就发生了变化。如'5'和 5 是不同的。

转义字符是一种特殊的字符常量，以反斜线“\”开头，后跟一个或若干个字符。转义字符与字符原有的意义不同，具有特定的含义，故称“转义”字符。例如，在例 2.7 中，有语句 printf（"a=%f\n", a）;其中“\n”就是一个转义字符，表示“回车换行”。

转义字符主要用来表示键盘上的控制代码或特殊符号，常用的转义字符见表 2.3。

表 2.3　常用的转义字符及其含义

转义字符	转义字符的意义	ASCII 码
\n	回车换行	10
\t	横向跳到下一制表位置（Tab）	9
\b	退格（Backspace）	8
\r	回车（不换行）	13
\f	走纸换页	12
\\	反斜线符（\）	92
\'	单引号符	39
\"	双引号符	34
\a	鸣铃	7
\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 2.3 中的\ddd 和\xhh 正是为此而提出的。ddd 和 hh 分别为八进制和十六进制的 ASCII 码。如\101 或\x41 表示大写字母“A”，\141 或\x61 表示小写字母“a”，\134 表示反斜线，\x0A 表示换行等。

【例 2.8】　转义字符的使用。

```
#include <stdio.h>
main()
{
    printf("\"China\"\n");
    printf("An\tHui\n");
}
```

运行程序，输出结果为：

```
" China "
An      Hui
```

2.5.2 字符变量

字符变量用来存储字符数据，即存储单个字符。

字符变量的类型说明符是 char。字符变量的类型定义及初始化格式与整型变量相同。

例如：

```
char ch1= 'x';              /*定义 ch1 为字符型变量，且初值为'x'*/
char ch2= 'y';              /*定义 ch2 为字符型变量，且初值为'y' */
unsigned char ch3;          /*定义 ch3 为无符号的字符型变量*/
```

在 C 语言中，每个字符变量被分配一个字节的内存空间，因此一个字符变量只能存放一个字符。字符变量是以 ASCII 码值的形式存储的。char 型数据的取值范围为−128～127，unsigned char 型数据的取值范围为 0～255，ASCII 码值为 0～127。

例如，小写字母 x 的 ASCII 码值是 120，小写字母 y 的 ASCII 码值是 121，那么上述定义的字符变量 ch1 和 ch2 在内存中的存储情况如下。

变量 ch1：

0	1	1	1	1	0	0	0

变量 ch2：

0	1	1	1	1	0	0	1

【例 2.9】 给字符变量赋整数值。

```
#include <stdio.h>
main( )
{
    char ch1, ch2;
    ch1=120;
    ch2=121;
    printf("%c, %c\n", ch1, ch2);
    printf("%d, %d\n", ch1, ch2);
}
```

运行程序，输出结果为：

```
x, y
120, 121
```

本例中 ch1、ch2 均为字符型变量，为什么可以赋给整型数值，而且又能以整数形式输出呢？

在 C 语言中，字符变量在内存中存储的是其对应的 ASCII 码值，字符型和整型密切相关，可以把字符型看做是一种特殊的整型。因此 C 语言允许对字符变量赋以整型值，允许把字符变量按整型输出；同样也允许对整型变量赋以字符型值，把整型变量按字符型输出。

从上述程序运行结果看，变量 ch1、ch2 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。

需要说明的是整型变量为双字节，字符变量为单字节，当整型变量按字符型处理时，只有低八位字节参与处理。

【例 2.10】　将小写字母转换为大写字母并输出。

```
#include <stdio.h>
main( )
{
    char ch1, ch2;
    ch1='a';
    ch2='b';
    ch1=ch1-32;
    ch2=ch2-32;
    printf("%c, %c\n%d, %d\n",ch1,ch2,ch1,ch2);
}
```

运行程序，输出结果为：

```
A, B
65, 66
```

本例中，变量 ch1、ch2 被定义为字符型变量并赋予字符值，C 语言允许字符变量参与数值运算，即允许字符变量用其 ASCII 码值参与运算。由于大写字母和小写字母的 ASCII 码值相差 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。

【例 2.11】　有以下程序：

```
#include <stdio.h>
main( )
{
    char a='\256';
    int b;
    b=a;
    printf("b=%d", b);
}
```

运行程序，输出结果为：

```
b= −82
```

程序中字符型变量 a 被赋值为转义字符'\256'，其十进制的 ASCII 码值为 174，那么把字符变量 a 的值赋给整型变量 b 后，b 的值为什么不是 174，而是−82 呢？这是因为把字符型变量按整型变量处理时，需要把字符的 ASCII 码值由一个字节扩展为两个字节。如字符'\256'的扩展情况如下。

一个字节：

1	0	1	0	1	1	1	0

扩展为两个字节：

1	1	1	1	1	1	1	1	1	0	1	0	1	1	1	0

这种扩展称为带符号位的扩展，即用字符 ASCII 码值的最高位填充扩展字节（高 8 位）。

如果把字符变量定义为 unsigned char 类型，将该字符变量赋给整型变量时，整型变量的高 8 位全部填入 0，即数值不变。请看下例。

【例 2.12】　将定义为 unsigned char 类型的字符变量赋给整型变量。

```
#include <stdio.h>
main( )
{
    unsigned char a='\256';
    int b;
    b=a;
    printf("b=%d", b);
}
```

程序运行的结果为：

```
b=174
```

2.5.3 字符串常量

字符串常量是由一对双引号括起的字符序列。例如，“CHINA”、“C program”、“$12.5”等都是字符串常量。

字符串常量和字符常量不同，二者具有以下区别：

（1）字符常量是由单引号括起来的字符，而字符串常量是由双引号括起来的字符。

（2）字符常量只能是 1 个字符，而字符串常量可以是多个字符。

（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。在C语言中没有专门的字符串变量，可以用一个字符型数组来存放一个字符串，本书在第 4 章数组中将详细介绍。

（4）字符常量占 1 字节的内存空间，而字符串常量所占内存的字节数等于其字符的个数加 1。C 语言规定，在字符串的结尾加一个字符串结束的标志'\0'（ASCII 码值为 0），以便系统据此判断字符串是否结束。

例如，字符串“CHINA”的字符长度为 5，在内存中占 6 字节：

C	H	I	N	A	\0

再比如，字符常量'a'和字符串常量“a”虽然都只有一个字符，但它们在内存中的存储情况是不同的。

'a'在内存中占 1 字节，可表示为：

a

"a"在内存中占 2 字节，可表示为：

A	\0

注意：两个连续的双引号""也是一个字符串，我们称其为“空字符串”。空字符串在内存中也要占一个字节的存储空间来存放'\0'。

实训 4　使用字符型数据

1．实训目的

熟练使用字符型数据。

2．实训内容

（1）将小写字母 j 转换为大写字母输出。

```
#include<stdio.h>
main( )
{
    char ch='j';
    ch=ch-'a'+'A';
    printf("%c\n",ch);
}
```

由于大、小写字母的 ASCII 码值是连续，所以字母 j 和 J 的差值与字母 a 和 A 的差值是相等的。程序中语句：ch=ch-'a'+'A';实现了小写字母转换为大写字母的作用。

请考虑，如果用语句 ch=ch-32; 替换语句 ch=ch-'a'+'A'; 能否实现程序功能。

（2）按以下格式输出某个学生的成绩。

```
数学                英语
80.500000           90.000000
```

```
#include<stdio.h>
main( )
{
    float math=80.5, english=90;
    printf("数学\t\t 英语\n");
    printf("%f\t%f\n", math, english);
}
```

程序使用了转义字符\t 和\n，请读者分析转义字符的作用。

（3）有如下程序：

```
#include<stdio.h>
main( )
{
   char ch1='a',ch2='b',ch3='c';
   char ch4='\101',ch5='\102',ch6='\103';
   printf("a%cb%c\tc%c\tabc\n",ch1,ch2,ch3);
   printf("\t%c%c\b%c\n",ch4,ch5,ch6);
}
```

请上机运行程序，并分析产生结果的原因。

3．实训思考

在 C 语言中，字符型数据和整型数据可以相互通用吗？

2.6　算术运算符和算术表达式

计算机通过各种运算完成对数据的处理，例如对数据可以进行加、减、乘、除等算术运

算，也可以进行关系运算、逻辑运算、位运算等。

用来表示各种运算的符号称为运算符，用运算符把各种运算对象（常量、变量、函数等）连接起来的、符合 C 语言语法规则的式子称为表达式。只有一个运算对象的运算符称为单目运算符，有两个运算对象的运算符称为双目运算符，有三个运算对象的运算符称为三目运算符。

当一个表达式中出现多个运算符时，就要考虑哪个运算符先运算，哪个运算符后运算，这就是运算符的优先级问题。优先级相同的运算符还有运算方向的规定，即结合性。自左向右进行运算的结合方向就为左结合性；而自右向左进行运算的结合方向为右结合性。因此在表达式中，各运算对象参与运算的先后顺序不仅要遵守运算符优先级的规定，还要受到运算符结合性的制约。

C 语言具有丰富的运算符和表达式，本章只介绍算术运算符、赋值运算符、逗号运算符和位运算符，其他运算符将在后续章节中介绍。

1．算术运算符

C 语言的算术运算符有 5 种：

+、−、*、/、%

它们分别是加法运算符、减法运算符、乘法运算符、除法运算符和求余（或取模）运算符。这些都是我们熟悉的运算，运算规则和数学中的基本一致。

以下有几点说明。

（1）“−”减法运算符：既是双目运算符又是单目运算符，用做单目运算符时，进行取负值运算，如，−5，−x 等。

（2）“/”除法运算符：当运算对象都是整型数据时，结果也为整型，舍去小数部分。如果运算对象中有一个是实型数据，则结果为实型。

（3）“%”求余运算符：要求运算对象必须为整型数据，结果是整除后的余数。如 a%b，结果为两数相除后的余数，结果的符号与 a 相同。

【例 2.13】 运行以下程序：

```
#include<stdio.h>
main( )
{
    printf("%d,%d\n",20/7,−20/7);
    printf("%f,%f\n",20.0/7,−20.0/7);
     }
```

输出结果为：

```
2,−2
2.857143,−2.857143
```

例中 20/7 和−20/7 的运算结果均为整型，而 20.0/7 和−20.0/7 由于有实数参与运算，因此结果为实型。

【例 2.14】 求余运算符应用举例。

```
#include<stdio.h>
main( )
{
```