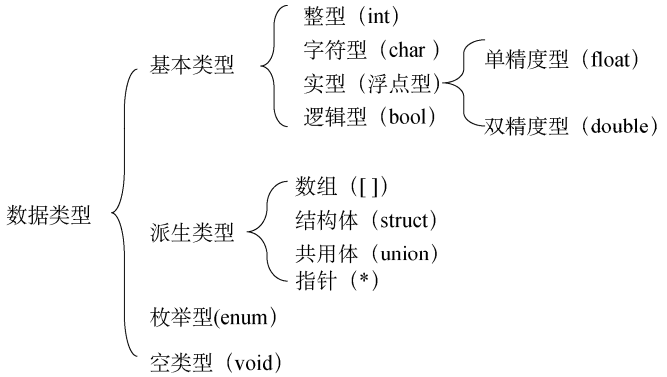
第 2 章　数据类型

学习目标

1. 理解常量、变量的含义及其基本类型。
2. 掌握 C 语言标志符的命名规则。
3. 掌握 C 语言中常见数据类型及其表示方法。

2.1　C 语言数据类型



（1）基本类型：其值不可以再分解为其他类型。包括整型、字符型、实型（分为单精度型和双精度型）和逻辑型四种。

（2）派生类型：根据已定义的一个或多个基本数据类型，使用构造的方法来定义。也就是说，一个派生类型的值可以分解成若干个“成员”或“元素”。每个“成员”都是一个基本数据类型或一个构造类型。派生类型有以下几种。

① 数组类型。

② 结构体类型。

③ 共用体（联合）类型。

④ 指针类型（保存某个变量在内存中的地址）。

（3）空类型：空类型 void 只能声明函数的返回值类型，不能用来声明变量。在调用函数值时，通常应向调用者返回某种类型的一个函数值，如果不需要函数返回值，则在函数定义时在函数前面加上 void，表示空类型。例如，void main()。

2.2　常量

基本数据类型，又分为常量和变量两种。在程序中，常量可以直接引用（无须事先说明），而变量则必须先定义才能使用。

所谓常量是指在程序运行过程中，其值不能被改变的量。在 C 语言中，常量分为五种：整型常量、实型常量、字符常量、字符串常量、符号常量。

整型常量和实型常量又称为数值型（或数字）常量，它们有正值和负值之分。两者的区别是：整型常量只能用数字表示，不带小数点。实型常量必须用带小数点的数表示。

1. 整型常量

整型常量有两种分类方法。一种是短整型常量、长整型常量、基本整型常量、无符号型常量。另一种是十进制整型常量、八进制整型常量和十六进制整型常量。

（1）十进制整型常量：由数字 0～9 组成。如 0，80，267，−82 等。十进制整型常量还有一种常量叫作长整型常量。一般在十进制常量后加上 L（或 l）就变成长整型常量，如 1256L。

（2）八进制整型常量：以 0 开头，由数字 0～7 组成，如 030，0345，−032，071 等。

（3）十六进制整型常量：以 0x 或 0X 开头，由数字 0～9、a～f 或 A～F 组成，如 0x31、0Xf1af、0xC5、−0XCAC 等。

2. 实型常量

实型常量（实数又称浮点数），有两种表示形式。

（1）十进制小数形式。由数字和一个小数点组成，且小数点不可缺少。如 3.12，.123，123.，0.0 等均是合法形式。注意：小数点前面或后面没有数字也是合法的形式，但必须有小数点。

（2）指数形式。任何一个合法的指数形式的实型常量从左往右都是由数字、字母 e(或 E）和指数三部分组成的。如 45.3e3(表示 45.3×10^3)，−231.23E12，−0.12e−2，12e0 等均是合法形式。

> 注意：
>
> 用指数形式表示实数时，e 或 E 之前必须有数字（如 e6 错误，系统会将其视为一个变量），e 或 E 后面的指数必须是整数。如写成 15e3.5 是错误的。

3. 字符常量和转义字符

（1）普通字符常量。ASCII 基本字符集中包括 94 个可视字符和 34 个控制字符，直接用单引号括起来即为字符常量。如'a'，'z'，'R'，'#'，'\n'等都是字符常量。

注意：

① 单引号不占存储空间。

② C 语言中，'a'和 a 不同。

③ 'B'和'b'，单引号括起的大写字母和小写字母代表不同的字符常量。

④ 单引号括起来的空格' '也是字符常量，存储的是空格的 ASCII 值 32。

⑤ 一个字符常量存放到一个字符变量中，是将该字符的相应的 ASCII 码放到存储单元中。字符型数据和整型数据在 ASCII 码值范围内可以共用。

（2）转义字符。ASCII 中的控制字符是不可见字符，不能直接用单引号括起的形式表示。在某些特定字符前加“\”，表示某种特殊的意义或控制动作，这种形式的字符为转义字符。

转义字符及转义字符的意义见表 2-1。

表 2-1 转义字符及转义字符的意义

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

注意：

'\101'表示 ASCII 为 65（八进制 101 等于十进制数 65）的字符。'\013'表示 ASCII 为 11 的字符（八进制数 13 等于十进制数 11）。

【例 2-1】

问题描述

转义字符的使用。

问题分析

\n 表示换行，\t 表示横向跳到下一制表位置。'\101'八进制数 101 相当于十进制 65，而 ASCII

值 65 对应的是字符 A。

程序实现

```
#include<stdio.h>
  int main()
{
printf ("%c\t%c\t%c\n"，'\101'，'\61'，'\x63')；return 0；
}
```

例 2-1 程序运行结果如图 2-1 所示。

图 2-1　例 2-1 程序运行结果

4. 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，"student" ，"what" ，"1a23" 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量。它们之间主要有以下区别。

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，只占一个字节，字符串常量则可以包含一个或多个字符。系统在存放字符串常量时一般会在这个字符串的后面加上字符'\0'（ASCII 值为 0）即“字符串结束标志”，所以所占的内存字节数等于字符串中字节数加 1。例如，字符常量'c'和字符串常量"c"虽然都只有一个字符，但'c'在内存中占一个字节，"a"在内存中占两个字节，因为最后多一个结束标记'\0'。

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。因为在 C 语言中没有相应的字符串变量，如果想将一个字符串存放在变量中，必须使用字符数组。相关内容会在数组中进行详细介绍。

5. 符号常量（选学内容）

在 C 语言中，可以用一个标志符来表示一个常量，称为符号常量。符号常量在使用之前必须先定义，其一般形式为：

```
#define 标志符 常量
```

① 注意行末尾没有分号。

② #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，其功能是把该标志符定义为其后的常量值。

③ 符号常量一经定义就只能代表被定义的那个常量或表达式，不能再做更改。

④ 一般符号常量的标志符用大写字母，变量标志符用小写字母，以示区别。

⑤ 使用符号常量的好处：含义清楚；能做到“一改全改”（尤其是代码中多处用到同一常量的情况）。

【例 2-2】

问题描述

符号常量的使用。

问题分析

通过#define PI 3.14 事先定义符号常量 PI，并赋值为 3.14。如果对 PI 进行调整，则只需在#define PI 3.14 前面修改这一处的值即可。

程序实现

```
#define PI 3.14
#include <stdio.h>
int main()
{
float r;
float s，c;
printf("请输入圆半径：");
scanf("%f"，&r);
c=2*PI*r;
s=PI*r*r;
printf("半径为：%f 的圆周长为：%f，圆面积为：%f\n"，r，c，s);
return 0;
}
```

例 2-2 程序运行结果如图 2-2 所示。

图 2-2　例 2-2 程序运行结果

2.3　变量

变量必须先定义后使用，定义变量实际上是为了在内存里为此变量分配相应的存储单元。

变量必须要有变量名，在内存中占据一定的存储单元，存储单元里存放的是该变量的值。不同数据类型的变量分配的存储单元大小不同。要区分变量名和变量值是两个不同的概念。变量的命名必须符合标志符的命名规则。标志符是一个名字，用来标志变量名、常量名、函数名、数组名、数据类型名和程序名等有效字符序列。变量名实际上是一个符号地址，代表内存中一定的存储单元，程序从变量中取值，实际上是通过变量名找到相应的内存地址，然后从其存储单元中读取数据。变量存储示意图如图 2-3 所示。

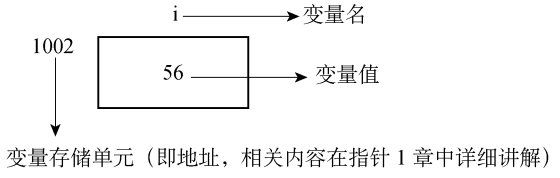


图 2-3　变量存储示意图

程序中用到的所有变量都必须遵守以下规则。

① 必须要有一个名字作为标志。

② 变量的名字由用户定义（必须符合标志符命名规则）。

③ 所有变量都必须先定义后使用。

④ 变量的定义可以放在函数体的前部，也可以放在函数体的外部或复合语句的开头（作用域不同，后面章节将具体介绍）。

⑤ 定义变量的同时必须说明其类型。

C 语言标志符的命名规则如下。

① 由字母（区分大小写）、数字和下画线组成。

② 由字母（区分大小写）或下画线开头。

③ 不能使用程序中具有特殊意义的关键字。

关键字（也称为保留字）就是具有特定含义的标志符，用户不能用来作为自定义标志符。C 语言中的关键字较少，由 ANSI 标准推荐的关键字有 32 个。

变量名、常量名、函数名、数组名、数据类型名和程序名等命名都必须符合标志符命名规则。

2.3.1　整型变量

1. 整型变量的分类

整型数据在内存中是以二进制补码形式存放的。如定义一个整型变量 a：

```
int a;
a=8;
```

整型变量的基本类型符为 int，在 int 之前可以根据需要分别加上修饰符：short（短型）或 long（长型），得到以下三种整型变量。

① 基本整型：类型说明符为 int，在内存中占 4 个字节（不同的 C 编译系统会有差异，有的占 2 个字节）。

② 短整型：类型说明符为 short int 或 short。所占字节和取值范围与基本整型相同。

③ 长整型：类型说明符为 long int 或 long，在内存中占 4 个字节。

【例 2-3】

问题描述

使用 sizeof() 函数测试并返回编译系统中各种数据类型所占的字节数。

问题分析

求字节数运算函数：sizeof()。

作用：可求出相应参数所占的内存字节数。

例：sizeof(int)；

sizeof(float)；

sizeof(double)；

sizeof(char)；

sizeof("Hangzhou")；

也能以 int i；sizeof(i)；的形式返回所占字节数。

程序实现

```
#include <stdio.h>
int main()
{
printf("int 型所占空间：%dB\n"，sizeof(int))；
printf("short int 型所占空间：%dB\n"，sizeof(short int))；
printf("long int 型所占空间：%dB\n"，sizeof(long int))；
return 0；
}
```

例 2-3 程序运行结果如图 2-4 所示。

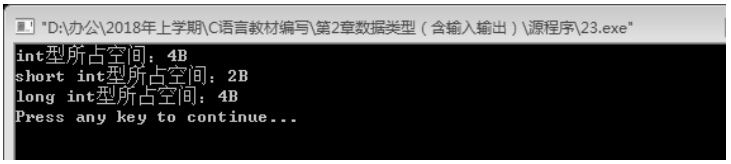


图 2-4　例 2-3 程序运行结果

试一试

编写程序测试编译系统中各种基本数据类型所占的字节数。在编辑环境中输入下列代码，验证几种数据类型的长度。

参考代码如下：

```
#include<stdio.h>
int main()
{
printf("int：%d\n"，sizeof(int))；
printf("long int：%d\n"，sizeof(long int))；
printf("long long int：%d\n"，sizeof(long long int))；
printf("char：%d\n"，sizeof(char))；
printf("float：%d\n"，sizeof(float))；
printf("double：%d\n"，sizeof(double))；
return 0；
}
```

归纳起来，有以下六种常见的整型变量。

① 有符号基本整型：　[signed] int

② 无符号基本整型：　unsigned [int]

③ 有符号短整型：　[signed] short[int]

④ 无符号短整型：　unsigned short [int]

⑤ 有符号长整型：　[signed] long [int]

⑥ 无符号长整型：　unsigned long[int]

各种无符号类型变量所占的内存空间字节数与相应的有符号类型变量相同。但由于省去了符号位（有符号数在存储单元中最高位表示符号，最高位为 0 表示正数，为 1 表示负数），不能表示负数，故一个无符号整型变量中可以存放的正数的范围比一般整型变量中正数的范围大一倍。各种类型变量表示数的范围及占用字节数见表 2-2。

表 2-2　各种类型变量表示数的范围及占用字节数

类型说明符	数的范围	字节数
int	−32768～32767 即-2^{15}～（$2^{15}-1$）	2
	−2147483648～2147483647 即-2^{31}～（$2^{31}-1$）	4
unsigned int	0～65535 即 0～（$2^{16}-1$）	2
	0～4294967295 即 0～（$2^{32}-1$）	4
short int	−32768～32767 即-2^{15}～（$2^{15}-1$）	2
unsigned short int	0～65535　即 0～（$2^{16}-1$）	2
long int	−2147483648～2147483647 即-2^{31}～（$2^{31}-1$）	4
long long int	−9223372036854775808～9223372036854775807 即-2^{63}～（$2^{63}-1$）	8
unsigned long	0～4294967295 即 0～（$2^{32}-1$）	4
unsigned long long	0～18446744073709551615 即 0～（$2^{64}-1$）	8

2. 整型变量的定义

C 语言是一种强类型语言，里面所有用到的变量都必须先定义后使用，定义时指明类型并指定变量名。

变量定义的一般形式为：

```
类型说明符  变量名标志符 1，变量名标志符 2，…；
```

例：

Int i，j；（指定 a，b，c 为整型变量）

Long x，y；（指定 x，y 为长整型变量）

Unsigned a，b；（指定 p，q 为无符号整型变量）

注意：

① 允许在一个类型说明符后，定义多个相同类型的变量。各变量名之间用逗号隔开；

② 类型说明符与变量名之间至少用一个空格间隔。

③ 最后一个变量名之后必须以“；”号结尾。

④ 变量定义必须放在变量使用之前。

【例 2-4】

问题描述

有符号整数−2 以无符号数形式输出。

问题分析

由于−2 的原码为：10000000000000000000000000000010

反码为：11111111111111111111111111111101

补码为：11111111111111111111111111111110

−2 在内存中以补码形式保存并以无符号数的形式输出，则前面的符号位也作为数值部分。11111111111111111111111111111110 等于 4294967295（相当于比全 1 要少 1），故为−1，显示为 4294967294。

程序实现

```
#include<stdio.h>
  int main()
{
    int a=2，b=−2;
    printf("a=%u，b=%u\n"，a，b)；// %u 以无符号数形式输出
```

```
    return 0;
}
```

例 2-4 程序运行结果如图 2-5 所示。

图 2-5　例 2-4 程序运行结果

试一试

请分析有符号整数−1 和−3 以无符号数形式输出结果为多少？

3. 整型数据的溢出

在 C 语言中，所有整型数据都有一定的取值范围，如一个 int 型变量在 C-free5 编译系统中占 4 个字节，最大允许值为 2147483647，如果再加 1，会出现什么情况呢？

【例 2-5】

问题描述

整型数据的溢出。

问题分析

2147483647 的原码为 01111111111111111111111111111111（由于在 C-free 编译系统中 int 占 4 个字节，所以 2147483647 转换成二进制为：符号位为 0 加 31 个 1，共 32 位）。

b=a+1 后编码为 10000000000000000000000000000000（1 加 31 个 0），则符号位变成 1（负数），而负数数值部分编码代表的是补码，故将其数值部分取反，取反结果（11111111111111111111111111111111 即 31 个 1）加 1 得到其表示的数值部分 10000000000000000000000000000000（等于 2^{31} 即 2147483648），再加上符号（由于符号位是 1）则是负数，即−2147483648。

在数学上，2147483647 加 1 是 2147483648，现在却变成−2147483648，这种情况即是 C 语言里面所谓的“溢出”现象，运行时是不会报错的。

程序实现

```
#include<stdio.h>
 int main()
{
   int a，b；
```

```
    a=2147483647;
    b=a+1;
    printf("a=%d, b=%d\n", a, b);
  }
```

例 2-5 程序运行结果如图 2-6 所示。

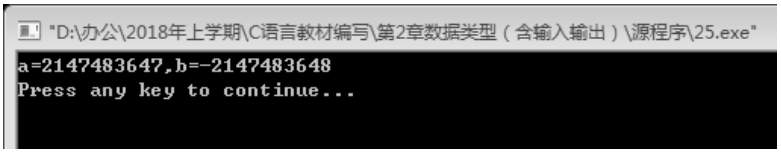


图 2-6　例 2-5 程序运行结果

【例 2-6】

问题描述

整型数据溢出的解决。

问题分析

要解决这种问题，我们先要估算数据的大小，然后选择一种比较合适的数据类型，如例 2-5，将 b 改为 long long int 型就可以得到预期结果 2147483648。

程序实现

```
#include<stdio.h>
 int main()
{
 long long int a, b;
   a=2147483647;
   b=a+1;
   printf("long long int 占字节数：%d\n", sizeof(a));
   printf("a=%lld, b=%lld\n", a, b);
  }
```

例 2-6 程序运行结果如图 2-7 所示。

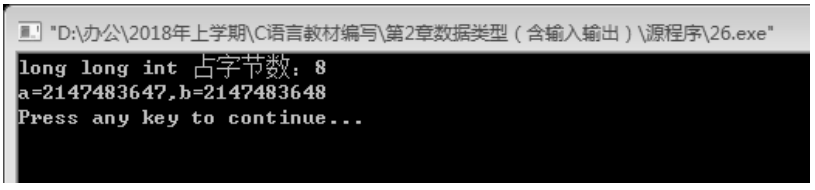


图 2-7　例 2-6 程序运行结果

2.3.2　实型变量

实型变量分为：单精度（float 型）、双精度（double 型）和长双精度（long double 型）三类。实型变量见表 2-3。

表 2-3　实型变量

类型说明符	字节数	有效数字	可表示的范围
float	4	6～7	10^{-37}～10^{38}
double	8	15～16	10^{-307}～10^{308}
long double	10	18～19	10^{-4931}～10^{4932}

注意：

long double（多精度浮点类型或长双精度浮点类型）是 1999 修订后的 C 语言标准“关键字”，但对 long double 的处理，取决于编译器。ANSI C 标准规定了 double 变量存储为 IEEE 64 位（8 个字节）浮点数值，但并未规定 long double 的精度。所以对于不同平台可能有不同的实现，有的是 8 字节，有的是 10 字节，有的是 12 字节或 16 字节。规定 long double 的精度不少于 double 的精度，就像 int 和 long int 一样。关于具体的编译器的情况，可以输出 sizeof（long double）得知。

实型变量定义的格式和书写规则与整型相同。

例：

float　a，b；（a，b 为单精度实型变量）

double　i，j；（i，j 为双精度实型变量）

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总是有限的，在有效位以外的数字将被舍弃。由此会产生一些误差，如例 2-7。

【例 2-7】

问题描述

实型数据的舍入误差。

问题分析

程序运行时，出现 a 值和 b 值相等，原因是 1234567.89e5+500 本来等于 1234567890500，但是一个单精度型实数变量保证的有效位只有 7 位，超过的数字是不准确的，因此才会出现上面的 a 和 b 的值相等的现象。

程序实现

```
#include<stdio.h>
int main()
{float a，b；
 a=1234567.89e5；
 b=a+500；
printf("%f\n"，a)；
printf("%f\n"，b)；
return 0；
}
```

例 2-7 程序运行结果如图 2-8 所示。

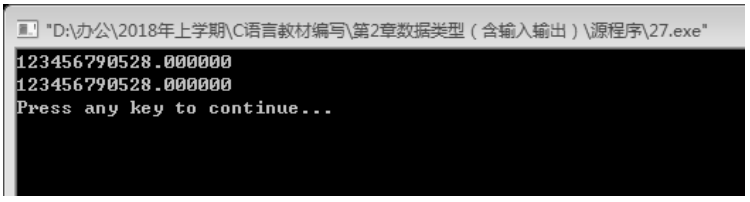


图 2-8　例 2-7 程序运行结果

【例 2-8】

问题描述

已知三角形的三边长 *a*，*b*，*c*，求出三角形的周长及面积。

问题分析

先判断三角形的三边长是否可以组成三角形（任意两边之和大于第三边），如果可以构成三角形则按照公式计算。周长=*a*+*b*+*c*，面积则可以用海伦公式[*p*=(*a*+*b*+*c*)/2]，则面积 *S*=sqrt[*p*(*p*−*a*)(*p*−*b*)(*p*−*c*)]。

程序实现

```
#include <stdio.h>
#include <math.h>
main()
{ float a，b，c，s，p，area；
a=3.2；
b=5.6；
c=6.53；
if((a+b>c)&&(a+c>b)&&(b+c>a))
{s=a+b+c；
p=(a+b+c)/2.0；
```

```
    area=sqrt(p*(p-a)*(p-b)*(p-c));
    printf("a=%f，b=%f，c=%f\n 周长=%f\n"，a，b，c，s);
    printf("面积=%f\n"，area);
    }
    else
    printf("你输入的三边长不能组成一个三角形");
    }
```

例 2-8 程序运行结果如图 2-9 所示。

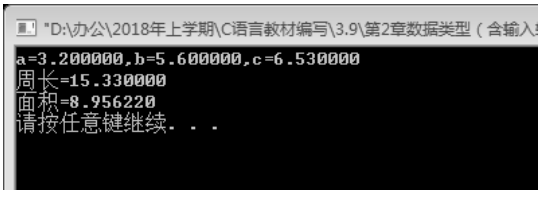


图 2-9　例 2-8 程序运行结果

2.3.3　字符变量

字符变量用来存储字符常量，即单个字符。字符变量的类型说明符是 char。字符变量类型定义的格式和书写规则都与整型变量相同。

例：

char c，d;（定义 c 和 d 为字符类型变量，各为一个字符）

对字符变量赋初值可以采用以下方式：

c='c'；d='d';（将字符常量 c 赋值给变量 c，字符常量 d 赋值给变量 d）

每个字符变量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中的。

注意：

① 记住几个常见字符的 ASCII 码。'A'：65，'a'：97，'0'：48

② 'A'～'Z'的 ASCII 码为 65～90，'a'～'z'的 ASCII 码为 97～122，'0'～'9'的 ASCII 码为 48～57。

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：01111000

b：01111001

所以也可以把它们看成整型变量。C 语言允许对整型变量赋予字符值，也允许对字符变量赋予整型值。在输出时，允许字符变量按整型量输出，也允许整型变量按字符变量输出。

注意：

整型变量占用 4 个字节，字符变量占用 1 个字节，当整型变量按字符变量处理时，只有低的 1 个字节（即 8 位）参与处理。

【例 2-9】

问题描述

向字符变量赋予整数，并分别以字符型及整型输出。

问题分析

字符型和整型在一定范围内通用，即一个整数可以直接以字符形式输出（输出这个整数对应的 ASCII）

本程序中定义 a，b，c 为字符型，但在赋值语句中赋予整型值。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。

由于 c=129，以%c 字符形式输出时，由于其二进制编码为：10000001，其最高位符号位为 1 表示负数，故无法正常输出，而输出'?'，将其符号位后面的取反加 1 即数值部分为 1111111，连续 7 个 1（故以%d 形式输出则为−127，也即 10000001 二进制代码对应的数值就是−127）。

程序实现

```
#include<stdio.h>
int main()
{
   char a，b，c；
   a=65；
   b=48；
   c=129；
printf("%c，%c，%c\n"，a，b，c)；
printf("%d，%d，%d\n"，a，b，c)；
return 0；
}
```

例 2-9 程序运行结果如图 2-10 所示。

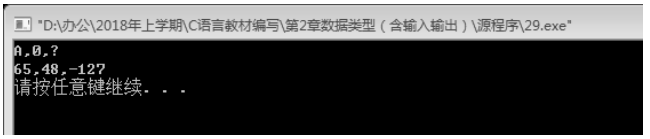


图 2-10　例 2-9 程序运行结果

分析以下程序的输出结果是_____。

```
#include<stdio.h>
int main()
{
char c;
c=0362;
printf("c=%d\n", c);
return 0;
}
```

参考分析：

c=0362 是八进制表示方法，对应的十进制数为 242。char 是一个字节的有符号类型，表示范围为−128～127。将 242 化为二进制形式（原码：11110010，反码：10001101，补码：10001110）后可以得出结果−14，所以输出−14。

【例 2-10】

问题描述

输入一个字母，如果是大写则转换为小写，如果是小写则转换为大写。

问题分析

本例中，ch 表示字符变量并赋予字符值，C 语言允许字符变量参与数值运算，即字符的 ASCII 码参与运算。由于大、小写字母的 ASCII 码相差 32，因此 ch−32 运算后把小写字母换成大写字母，ch+32 运算后把大写字母换成小写字母。

程序实现

```
#include<stdio.h>
int main()
{   char   ch;
printf("请输入一个字母：");
scanf("%c", &ch); /*   从键盘输入一个字母   */
if(ch>='A'&&ch<='Z')
printf("%c\n", ch+32);
if(ch>='a'&&ch<='z')/*   将该字母转换为大写字母   */
printf("%c\n", ch−32); /*   输出转换后的结果          */
return 0;
}
```

例 2-10 程序运行结果如图 2-11 所示。

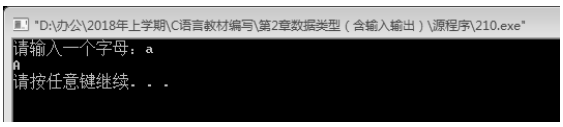


图 2-11　例 2-10 程序运行结果

变量赋初值一般有以下几种方式。

① 先定义，后赋初值。

例：

```
float a;
char c1，c2;
a=3.4;
c1='a'；c2='k';
```

② 在定义变量时赋初值。如上例也可以写成：

```
float a=3.4;
char c1='a'，c2='k';
```

2.4 输入与输出函数 printf()和 scanf()

2.4.1 用 printf()函数实现输出

printf()函数是格式化输出函数，一般用于向标准输出设备按规定格式输出信息。printf()函数的调用格式为：

```
printf("<格式化字符串>"，<参量表>)；
```

其中，格式化字符串包括两部分内容，一部分是正常字符，这些字符将按原样输出（如printf("a=……")；另一部分是格式控制字符，以%开始，后跟一个或几个控制字符，用来确定输出内容格式。

参量表是需要输出的一系列参数，可以是常量、变量或表达式，其个数必须与格式化字符串所说明的输出参数个数一样多，各参数之间用，分开，且顺序一一对应，否则将会出现错误。

在输出时，对不同类型的数据要使用不同的格式字符。常用的有以下几种：

（1）d 格式符。用来输出十进制整数。

① %d。按十进制整型数据的实际长度输出。

② %md。M 为指定的输出字段宽度。如果数据的位数小于 m，则左端补以空格，若大于 m，则按实际位数输出。

③ %ld。输出长整型数据。

（2）o 格式符。以八进制整数形式输出。由于将内存单元中的各位的值（0 或 1）按八进制形式输出，因此输出的数值不带符号，即符号位也作为八进制数的一部分输出。

（3）x 格式符。以十六进制数形式输出整数（不会出现负的十六进制数）。

（4）u 格式符。用来输出 unsigned 型数据，即无符号数，以十进制整数形式输出。

一个有符号整数（int 型）也可以用%u 格式输出；反之，一个 unsigned 型数据也可以用%d 格式输出。按相互赋值的规则处理。unsigned 型数据也可以用%o 和%x 格式输出。

（5）c 格式符。用来输出一个字符。一个整数，只要它的值在 0～255 内，也可以用%c 格式，使之按字符形式输出，在输出前，系统会将该整数作为 ASCII 码转换成相应的字符；反之，一个字符数据也可以用整数形式输出。

（6）s 格式符，用来输出一个字符串，有几种用法。

① %s。输出字符串（不包括双引号）。

② %ms。输出的字符串占 m 列，如字符串本身长度大于 m，则突破 m 的限制，将字符串全部输出，若字符串长度小于 m，则左补空格。

③ %-ms。与%ms 区别是右补空格。

④ %m.ns。输出的字符串占 m 列，只取字符串左端 n 个字符，不足位数，左补空格。

⑤ %-m.ns。与%m.ns 的区别就是右补空格。

（7）f 格式符。用来输出实数（包括单、双精度），以小数形式输出。有以下几种用法：

① %f，不指定字段宽度，由系统自动指定，整数部分全部输出，并输出 6 位小数。单精度实数的有效位数一般为 7 位（包括整数和小数位数）。

显然，只有前 7 位数字是有效数字，千万不要认为凡是计算机输出的数字都是准确的。双精度数也可以使用%f 格式输出，它的有效位数一般为 16 位，给出小数 6 位。

② %m.nf，指定输出的数据共占 m 列，其中有 n 位小数。长度不够，左补空格。

③ %-m.nf，右补空格。

（8）e 格式符。以指数形式输出实数。

（9）g 格式符。用来输出实数，它根据数值的大小，自动选择 f 格式或 e 格式。

【例 2-11】

问题描述

利用 printf()函数输出数据。

问题分析

%o：八进制数形式输出。%x：十六进制形式输出。%5.3f：总宽度为 5，其中小数位为 3（若不到 5 位则左边产生空格，若超过 5 位则按照实际位数输出）。%−7.3f：总宽度为 7，若不到 7 位则在右边产生空格。

程序实现

```
#include<stdio.h>
int main()
{int a=21;
float b=3.1415926;
printf("八进制为(21)：%o\n 十六进制为(21)：%x\n"，a，a)；
printf("%5.3f\n"，b)；
printf("%7.3f\n"，b)；
printf("%-7.3f"，b)；
return 0;
}
```

例 2-11 程序运行结果如图 2-12 所示。

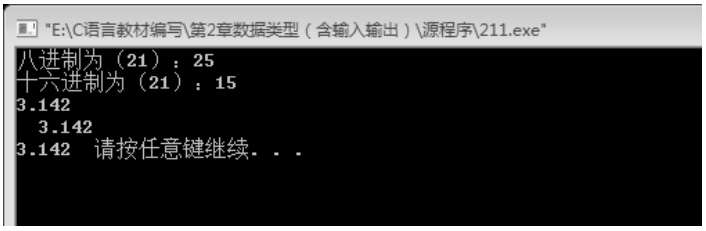


图 2-12 例 2-11 程序运行结果

2.4.2 用 scanf()函数实现输入

scanf()函数是格式化输入函数，它从标准输入设备（键盘）读取输入的信息。

其调用格式为：

```
scanf("<格式化字符串>"，<地址表>)；
```

格式化字符串包括以下三类不同的字符。

（1）格式化说明符。格式化说明符与 printf()函数中的格式化字符串基本相同。

（2）空白字符。空白字符会使 scanf()函数在读操作中略去输入中的一个或多个空白字符。

（3）非空白字符。一个非空白字符会使 scanf()函数在读入时剔除与这个非空白字符相同的字符。

地址表是需要读入的所有变量的地址，而不是变量本身。这与 printf()函数完全不同，要特别注意。各个变量的地址之间使用，分开。scanf()函数从标准输入中读取字符串序列，按照 format 中的格式说明对字符序列进行解释，并把结果保存到其余的参数中，所有参数都必须是指针。

说明：

（1）对于字符串数组或字符串指针变量，由于数组名和指针变量名本身代表的就是地址，

因此使用 scanf()函数时，不需要在它们前面加上"&"操作符。其余的需要在它们前面加上"&"操作符。

（2）可以在格式化字符串中的"%"与各格式化规定符之间加入一个整数，表示读操作中的最大位数。

非格式化输入输出函数可以由上面讲述的标准格式化输入输出函数代替，但这些函数编译后代码少，相对占用内存也小，从而提高了速度，同时使用也比较方便。下面分别进行介绍。特别注意以下容易出现错误的情况。

① scanf 函数中的“格式控制”后面应当是变量地址，而不应是变量名。例如，如果 a、b 为整型变量，则

```
scanf("%d，%d"，a，b)；
```

是错误的，应将“a，b”改为“&a，&b”。开始学编程时，容易造成该错误。

② 如果在“格式控制”字符串中除了格式说明外还有其他字符，则在输入数据时相应的位置上应输入同样的字符（即除%规定的格式控制字符外部分格式应严格一致）。例如，

```
scanf("%d，%d"，&a，&b)；
```

输入时应输入：3，4。3 与 4 之间的逗号应与 scanf 函数中的“格式控制”中的逗号相对应，输入其他符号（如 3　4 或 3#4 等）是不对的。

```
scanf("a=%d，b=%d"，&a，&b)；
```

输入时应输入：a=3，b=4。

③ 用"%c"格式输入字符时，空格字符和转义字符都作为有效字符输入。

```
scanf("%c%c%c"，&c1，&c2，&c3)；
```

如输入：a b c。字符'a'赋给 c1，字符（空格）' '赋给 c2，字符'b'赋给 c3。

④ scanf 中不使用 u 说明符，对 unsigned 型数据，在 scanf 函数中使用 d、o、x 说明符输入。

⑤ 在使用 scanf 函数读取数据时，如果读取的是浮点数，则不能指定精度。

⑥ double 类型的数据在输入时，注意不要使用%f 形式输入，而应使用%lf 形式。

【例 2-12】

问题描述

输入三个整数 a，b，c，输出一元二次方程式 $ax^2+bx+c=0$ 在 1～8 之内的整数解。