第3章 C语言基本程序语句

学习目标

- 掌握标识符、常量、变量、注释的使用规则；
- 了解C语言中常用的数据类型；
- 掌握各种运算符与表达式的运算方法；
- 掌握常用输入输出函数的语法规则；
- 能进行简单的顺序结构程序设计。

计算机程序设计涉及两个基本问题：一个是数据的描述；一个是动作的描述。计算机程序的任务就是对数据进行处理。没有数据，程序就没有了处理对象；没有动作，程序也就失去了意义。程序中定义变量存放数据，保存在内存单元中，变量区分不同的数据类型，每一种数据类型存储空间不同，通过不同类型的数据和简单的表达式，能够编写一些简单的顺序结构程序。

3.1 标识符、关键字、注释

3.1.1 标识符

简单地说，标识符就是一个名称，人的姓名就是对人的标识符。在C语言中为变量、数组、函数、数据类型以及文件等命名的名称，被统称为标识符。

C语言中标识符定义规则如下。

(1) 标识符只能由字母、数字和下画线组成，并且第一个字符不可以是数字。例如，sun和_sun是合法的标识符，6sun是不合法的标识符。

(2) 标识符区分大小写，score、Score、SCORE代表三个不同的标识符。

(3) 标识符不能和C语言的关键字相同。

(4) 标识符可以由一个或多个字符组成，最长不允许超过32个字符，标识符的有效长度取决于具体的C编译系统。

(5) 标识符最好是“见名知意”，这样可提高程序的可读性。例如，定义变量名为age表示年龄，name表示姓名。

表3-1为标识符示例。

表 3-1　标识符示例

标　识　符	说　　明
stu_name1	正确
Cno	正确
9class	错误,标识符不可用数字开头
ok%a1	错误,标识符中只能包含字母、数字和下画线,%不合法
int	错误,C 语言关键字不可以用作标识符
name 123	错误,标识符中不可以有空格

3.1.2　关键字

关键字又称保留字,是 C 语言预定义的单词,在程序中有特定的含义,在定义标识符的时候,不能使用这些关键字。C 语言中所有关键字必须用小写英文字母,共有 4 大类 32 个关键字,其中数据类型关键字 12 个(基本数据类型 9 个,自定义数据类型 3 个),控制语句关键字 12 个,存储类型关键字 4 个,其他类型关键字 4 个,具体内容如表 3-2 所示。

表 3-2　C 语言关键字

类　　型	关　键　字	作　　用
基本数据类型(9 个)	char	字符型数据类型,占 1 字节
	int	整型数据类型,占 2 或 4 字节
	short	短整型数据类型,占 2 字节
	long	长整型数据类型,占 4 字节
	float	单精度浮点型数据类型,占 4 字节
	double	双精度浮点型数据类型,占 8 字节
	signed	表示有符号整数,是默认值,一般省略不写
	unsigned	表示无符号整数
	void	空值,用于定义函数,表示无返回值
自定义数据类型(3 个)	struct	结构体数据类型
	union	联合体(共用体)数据类型
	enum	枚举数据类型
控制语句(12 个)	if-else	双分支结构的两个关键字
	switch-case default	多分支结构的三个关键字
	for	for 循环,一般用于循环次数固定的循环
	while	while 循环,先判断条件后运行语句
	do-while	do-while 循环,先运行一次语句,再判断条件
	continue	不终止循环,结束当前循环提前进入下一个循环
	break	终止语句,终止循环或跳出 switch 分支
	goto	无条件转向语句,不建议使用
	return	返回语句,返回到函数调用处

续表

类　型	关 键 字	作　用
存储类型(4 个)	auto	自动型,局部变量的默认值,存在内存动态区域
	extern	定义全局变量表示可被其他文件访问,在静态区域,定义函数表示外部函数; 全局变量和函数默认 extern 型
	register	寄存器型,只用于定义局部变量,存在寄存器中
	static	静态型,定义局部变量表示静态变量,存在静态区域,值保留并自动赋初值; 定义全局变量表示有效范围在该源文件; 定义函数表示内部函数
其他类型(4 个)	const	声明只读变量
	sizeof	计算数据类型长度,返回字节数
	typedef	类型定义,为自定义数据类型取别名
	volatile	变量在程序运行过程中可被隐含地改变

3.1.3 注释

程序注释是以特定的格式出现在程序代码中,它不是程序运行语句的一部分,而是程序员解释语句的文字说明。程序中可以有多处注释,注释不参与程序运行。

注释常用的两个作用: 一是解释说明; 二是将暂时不需要运行的代码注释,需要的时候再去掉注释标志。

C 语言中注释有两种: 一种是用/ * * /括起来的多行注释,可以放在程序的任意位置; 另一种是用两个反斜杠//表示的单行注释,只能放在程序代码行尾部。注释示例见如下代码:

```
#include <stdio.h>
int main()                                  /*注释 1.这是程序入口*/
{
    printf("hello world");                  //注释 2.这是输出语句
}
```

3.2 数据类型、常量和变量

3.2.1 数据类型

C 语言程序描述数据时必须先指定数据类型,不同类型的数据在计算机中的存储方式和处理方式都不相同,数据类型是计算机程序设计中一个非常重要的概念。

C 语言有一个很重要的特点就是数据类型十分丰富,因此,C 语言的数据处理能力很强。C 语言的数据类型如图 3-1 所示。

1. 基本数据类型

基本数据类型是指不可再分解的数据类型,包括整型、浮点型和字符型。

整型又分为有符号整型和无符号整型,有符号整型和无符号整型还细分为短整型、整型和长整型,它们占的存储空间大小不同。

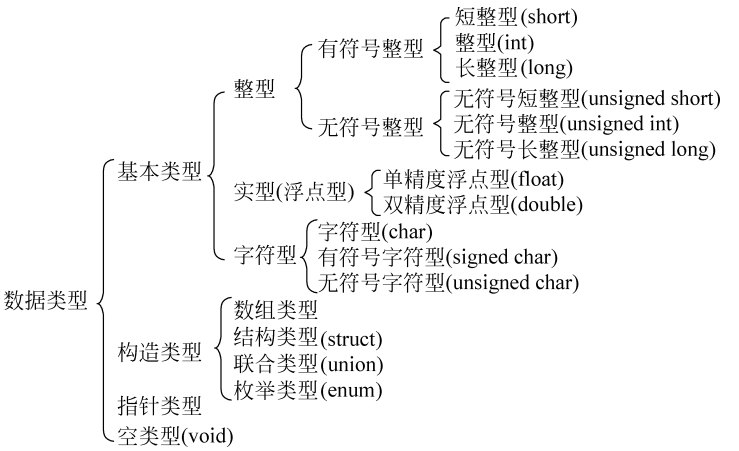

图 3-1　C 语言的数据类型

浮点型就是带小数的数据类型，分为单精度浮点型和双精度浮点型，二者区别是因分配的存储空间不同，从而可存储数据的精度不同，具体介绍见 2.5 节。

字符型用于存储英文字母、标点符号等字符。因为字符在计算机中以 ASCII 码存储，因此，字符型实际上也是整型，是 1 字节的整型。所以 C 语言允许字符型数据和整型数据在一定取值范围内通用。例如，小写字母 a 的 ASCII 码是 97，计算机中存储 a 是以 1 字节存储 97 的二进制值。通常可以通过 ASCII 码的运算来实现字母间的转换，如大写字母 A 加上 32 等于小写字母 a。表 3-3 为大小写字母十进制 ASCII 码对照表。

表 3-3　大小写字母十进制 ASCII 码对照表

十进制	缩写/字符	十进制	缩写/字符	十进制	缩写/字符
65	A	83	S	106	j
66	B	84	T	107	k
67	C	85	U	108	l
68	D	86	V	109	m
69	E	87	W	110	n
70	F	88	X	111	o
71	G	89	Y	112	p
72	H	90	Z	113	q
73	I	…	…	114	r
74	J	97	a	115	s
75	K	98	b	116	t
76	L	99	c	117	u
77	M	100	d	118	v
78	N	101	e	119	w
79	O	102	f	120	x
80	P	103	g	121	y
81	Q	104	h	122	z
82	R	105	i		

2. 构造数据类型

构造数据类型是指使用基本数据类型构造成的新的数据类型，也就是说一个构造数据类型可以分解成若干“成员”或“元素”，每个“成员”或“元素”都是一个基本数据类型或者另一个构造数据类型。C 语言中构造数据类型包括数组、结构体类型、联合体类型（共用体类型）和枚举类型。

提示：C 语言中没有字符串数据类型，用字符型数组实现字符串。

3. 指针类型

指针是 C 语言的精华，指针类型不同于其他类型的特殊性在于指针的值表示的是某个内存的地址。虽然指针变量的取值类似于整型值，但它和整型变量是完全不同的含义，不能混为一谈。

4. 空类型

空类型的关键字是 void，用于定义函数的返回值。如果一个函数没有返回值，可以定义该函数的返回值类型为 void。

数据类型中的构造数据类型、指针类型会在后面章节详细介绍。表 3-4 列出 C 语言基本数据类型所占字节数和取值范围，其中整型类型所占字节数与机器的 CPU 类型和编译器有关，有的占 2 字节，有的占 4 字节，VC++中整型占 4 字节。

表 3-4　C 语言基本数据类型所占字节数和取值范围

数 据 类 型	类型标识符	字节数	取 值 范 围
短整型	short	2	$-32768\sim32767(-2^{15}\sim2^{15}-1)$
整型	int	2 或 4	同 short 或 long，取决于 C 编译系统
长整型	long	4	$-2147483648\sim2147483647(-2^{31}\sim2^{31}-1)$
无符号短整型	unsigned short	2	$0\sim65535(0\sim2^{16}-1)$
无符号整型	unsigned int	2 或 4	同 unsigned short 或 unsigned long
无符号长整型	unsigned long	4	$0\sim4294967295(0\sim2^{32}-1)$
单精度浮点型	float	4	$-3.4\times10^{-38}\sim 3.4\times10^{38}$
双精度浮点型	double	8	$-1.7\times10^{-308}\sim 1.7\times10^{308}$
字符型	char	1	ASCII 码 0～127
有符号字符型	signed char	1	$-128\sim127(-2^{7}\sim2^{7}-1)$
无符号字符型	unsigned char	1	0～255

编写程序验证数据类型的字节长度，程序代码如下，运行效果如图 3-2 所示。

```
#include <stdio.h>          //包含头文件
void main()                 //主函数
{
    char a1;                //定义变量
    short int b1;
    int c1;
    long int d1;
    float e1;
    double f1;
    printf("size of (char) = %d\n", sizeof(a1));
```

```
size of (char)=1
size of (short int)=2
size of (int)=4
size of (long int)=4
size of (float)=4
size of (double)=8
```

图 3-2　VC 环境下不同数据类型所占字节数

```
    printf("size of (short int) = %d\n", sizeof(b1));
    printf("size of (int) = %d\n", sizeof(c1));
    printf("size of (long int) = %d\n", sizeof(d1));
    printf("size of (float) = %d\n", sizeof(e1));
    printf("size of (double) = %d\n", sizeof(f1));
}
```

说明：如果使用 VC 2010 环境，可在语句最后增加 getchar();语句，让运行画面停留。

3.2.2 常量

常量是指在程序运行过程中，其值不能被改变的量。也就是说，在 C 语言程序中，常量的值是固定的，是不能在程序运行过程中修改的。C 语言中常量分为直接常量和符号常量，直接常量又分为整型常量、实型常量、字符常量、字符串常量 4 种类型，整型常量和实型常量又称为数值型常量。

1. 整型常量

C 语言中可采用十进制数、八进制数、十六进制数来表示整型常量，C 语言不支持二进制常量。八进制数用数字 0 开头；十六进制数用数字 0 和字母 X 开头(0x 或 0X)。八进制数和十六进制数一般只用于无符号整数。表 3-5 为整型常量示例。

表 3-5 整型常量示例

常　量	说　明
−200	正确，十进制整型常量
200L	正确，十进制长整型常量，后缀 L(大小写均可)表示长整型
200lu	正确，十进制无符号长整型常量，后缀 u(大小写均可)表示无符号整数
010	正确，八进制整型常量，等于十进制数 8
0x10	正确，十六进制整型常量，等于十进制数 16
567	正确，十进制常量
019	错误，0 开头表示八进制常量，最大数应为 7，数字 9 错误
DD	错误，前面没有加 0x，应该是十进制常量，D 不是十进制数
0X3h	错误，前面加 0X，表明是十六进制数常量，最大应为 F，h 错误
OX11	错误，十六进制数常量开头应该是数字 0，而不是字母 O

2. 实型常量

实型就是带小数的浮点数。实型常量有两种表示法：浮点计数法和科学计数法。一般情况下，对太大或太小的数采用科学计数法。表 3-6 为实型常量示例。

表 3-6 实型常量示例

常　量	说　明
231.46	正确，双精度浮点型常量
7.36E−7	正确，双精度浮点型常量，以科学计数法表示
−0.0945	正确，双精度浮点型常量
231.46f	正确，单精度浮点型常量，后缀 f(大小写均可)表示单精度，默认双精度
1.2e7.5	错误，指数不可以是小数，1.2e7 或者 1.2e5 都是正确的
2.5e	错误，指数不可省略，e 后面必须有一个整数

续表

常　量	说　明
e7	错误，e 前面需有一个数，不可省略
.e2	错误，e 前面需有一个数，不可只写小数点

3. 字符常量

字符常量是由一对单引号括起来的单个字符。它又分为一般字符常量和特殊字符常量。

例如，'B'、'D'、'7'、'@' 等均为一般字符常量。这里的单引号只起定界作用，并不代表字符。在 C 语言中，字符是按其所对应的 ASCII 码来存储的，一个字符占一个字节，有效取值范围为 0～127。因此，字符型常量可以像整数一样在程序中参与运算，但要注意不要超过它的有效范围。

特殊字符常量就是转义字符，转义字符也是字符常量的一种表示形式，转义字符用反斜杠\后面跟一个字符或一个八进制数或十六进制数表示。例如，'\a'、'\0'、'\n' 等代表 ASCII 字符中不可打印的控制字符和特定功能的字符。

单引号 (')、双引号(")、反斜杠(\)和百分号(%)在程序中有特定用途，其本身作为字符时，要通过转义字符(\)或(%)来转义为普通字符进行输出。例如，'\'' 和 '\\' 分别代表单个字符单引号(')和反斜杠(\)，'%%' 表示输出一个百分号(%)。表 3-7 为常用转义字符表。

表 3-7　常用转义字符表

转义字符	意　义	ASCII 码(十进制)
\a	鸣铃(BEL)	7
\b	退一格(BS)	8
\f	换页(FF)	12
\n	回车换行(LF)，使用频率最高	10
\r	回到本行的开始(CR)	13
\t	水平制表符，横向调到下一个制表位(HT)	9
\v	垂直制表符(VT)	11
\\	表示反斜杠	92
\'	单引号	39
\"	双引号	34
\0	空字符(NULL)，也就是 ASCII 码为 0 的字符	0
\ddd	八进制 ASCII 码为 ddd 的字符(最多三位数)	ddd(八进制)
\xhh	十六进制 ASCII 码为 hh 的字符(最多两位，最大 7f)	hh(十六进制)
%%	表示一个百分号	37

提示：

(1) 转义字符中的字母只能是小写字母，每个转义字符只能看作一个字符。

(2) 表 3-7 中的\r、\v 和\f 对屏幕输出不起作用，但会在控制打印机输出时响应其操作。

(3) 在程序中，使用不可打印字符时，通常用转义字符表示。

表 3-8 为字符型常量示例。

表 3-8　字符型常量示例

常　　量	说　　明
'a'	正确,字符型常量,表示字母 a
'2'	正确,字符型常量,此处的 2 不是数字型,是字符型,在内存中占 1 字节,存的是字符 2 的 ASCII 码 50 的二进制数
'\''	正确,字符型常量,表示一个单引号
'\141'	正确,字符型常量,表示字母 a,141 是字母 a 的八进制 ASCII 码
'\x61'	正确,字符型常量,表示字母 a,61 是字母 a 的十六进制 ASCII 码
"a"	错误,字符常量应该用单引号定界,不可用双引号,双引号表示字符串
'12'	错误,字符型常量只能是一个字符
'\x101'	错误,\x 表示后面数字是两位十六进制 ASCII 码,最大是 7f
'\2A'	错误,\后面数字表示八进制 ASCII 码,8 进制最大数值是 7,A 错误

4. 字符串常量

字符串常量是指用一对双引号括起来的一串字符。

字符常量与字符串常量有以下区别。

(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) 字符常量只能表示一个字符,字符串常量可以是零个或多个字符。

(3) 字符常量单引号中必须有一个字符,''表述错误,""表述正确,表示一个空串;

(4) 在内存中,字符型常量占一个字节,字符串常量所占的字节数等于字符串中字符的个数加 1。因为,C 语言在内存中存储字符串常量时,自动在字符串末尾加一个结束标志'\0',表示 ASCⅡ码值为 0 的空字符(NULL)。因此,长度为 n 个字符的字符串常量,在内存中占有 n+1 个字节的存储空间。

例如,字符串"World",共 5 个字符,需占用 6 字节,其存储形式如图 3-3 所示。表 3-9 为字符串常量示例。

W	o	r	l	d	\0

图 3-3　字符串" World"存储示意图

表 3-9　字符串常量示例

常　　量	说　　明
"china"	正确,字符串常量,占 6 字节
"123.33"	正确,字符串常量,占 7 字节
"you love"	正确,字符串常量,占 9 字节
'ab'	错误,字符串用双引号,字符用单引号,字符串"ab"、字符'a'或'b'均正确

5. 符号常量

在 C 语言中,可以用标识符表示一个常量,称为符号常量。习惯上,符号常量用大写英文字母表示,以区别于变量。符号常量在使用前必须先定义,定义时根据数据类型决定如何书写,不需要用等号。符号常量的定义形式如下:

```
#define  <符号常量名>  <常量>
如: #define PI  3.14159
    #define SEX 'M'
    #define NAME  "张三"
```

其中 PI、SEX、NAME 均为符号常量；其值分别为数字常量 3.14159，字符常量'M'，字符串常量"张三"。

#define 是 C 语言的预处理命令，它和 #include 一样，都不是 C 语句，后面不加分号。在编译程序时会将程序中出现符号常量的地方直接用值替换，如出现 PI 的地方都替换为 3.14159。

定义符号常量可以提高程序的可读性，便于程序的修改和调试。在定义符号常量名时，应尽量使用能表达其实际意义的名字。表 3-10 为符号常量示例。

表 3-10　符号常量示例

符号常量定义	说　　明
#define PI　3.1415926	正确，符号常量 PI 表示数值型常量 3.1415926
#define PI　"3.1415926"	正确，符号常量 PI 表示字符串型常量"3.1415926"
#define SEX　'M'	正确，符号常量 SEX 表示字符'M'
#define SEX　"M"	正确，符号常量 SEX 表示字符串"M"，不是字符
#define SEX　M	错误，只有数值型常量才可以不用引号
#define NAME　"张三"	正确，符号常量 NAME 表示字符串"张三"
#define NAME　'张三'	错误，字符常量用单引号，字符串型常量必须用双引号
#define NAME　张三	错误，"张三"是个字符串，必须用双引号引上

3.2.3　变量

1. 什么是变量

在程序运行过程中，其值可以改变的量称为变量。变量需要有名字，用标识符来命名。变量在内存中占一定的存储空间，空间大小取决于变量的数据类型。变量必须先定义后使用，变量的值可以通过赋值的方法获得和改变。

提示：一个变量只能存放一个数据。

2. 变量的定义和初始化

C 语言规定：变量必须在使用之前定义。变量定义的一般形式是：

```
数据类型 变量名 1[ = 初值],变量名 2[ = 初值]…变量名 n[ = 初值];
```

说明：

(1) 数据类型必须是有效的 C 数据类型，如 int、float 等，数据类型决定了变量所占存储空间大小和取值范围。

(2) 数据类型与变量名之间至少空一个空格。

(3) 一个语句可以同时定义多个同一数据类型的变量，变量之间用逗号分隔，最后一个变量之后用分号结束。

(4) 变量名必须符合标识符命名规则，而且尽量“见名知意”。

(5) 同一个函数中,变量名不可以重复。

(6) 定义变量的同时也可以给变量赋初值。中括号[]内的项是可选项。

3. 变量的赋值

定义变量的同时可以给变量赋初值,在程序运行过程中可以随时改变变量的值。

例如:

```
void main()
{
   int i = 9, j;              //同时定义两个变量 i 和 j,i 赋初值 9,j 未赋初值
   j = 11;                    //为 j 赋值
   i = 80;                    //为 i 赋值,80 替换了初值 9
   printf(" % d\n", i);       //输出 i 值,格式符 % d 表示整数,\n 表示回车换行
}
```

程序运行结果是:80

提示:如果使用 VC 2010 环境,可在语句最后,结尾大括号}前面,增加 getchar();语句,避免画面一闪而过。

变量定义和赋值示例如表 3-11 所示。

表 3-11 变量定义和赋值示例

示 例	说 明
int i	正确,定义整型变量 i
unsigned int i, j, number;	正确,定义三个无符号整型变量,逗号分隔,分号结束
float high_value, price;	正确,定义两个浮点型变量 high_value 和 price
double lenth, t_wieight;	正确,定义两个双精度浮点型变量
int max = 0, min = 0;	正确,定义两个整型变量,都赋初值 0
int max, min = 0;	正确,定义两个整型变量,只给 min 赋初值 0
int max = min = 0;	错误,语法错误,min 变量未定义
int i, j, k; i = j = k = 3;	正确,定义三个整型变量,未赋初值 为三个变量都赋值为 3

【例 3-1】 以下关于 long、int 和 short 类型占用内存大小的叙述中正确的是(　　)。

A. 均占 4 个字节　　B. 根据数据的大小来决定所占内存的字节数

C. 用户自己定义　　D. 由编译器决定

答案:D

解析:各个类型数据所占内存空间是固定的,不能由用户决定。不管是哪一种编译器,long 型固定占 4 字节,short 型固定占 2 字节,而 int 型所占字节数与编译器有关,可能是 2 或 4 字节。

3.3 运算符与表达式

C 语言的基本表达式由操作数和操作符组成。操作数通常由变量或常量表示,操作符由各种运算符表示,一个基本表达式也可以作为操作数来构成复杂表达式。构成基本表达

式的常用运算符有以下 4 种：

(1) 算术运算符；

(2) 关系运算符；

(3) 逻辑运算符；

(4) 赋值运算符。

此外，还有位运算符、条件运算符、逗号运算符、自反赋值运算符、指针运算符等，甚至把括号、强制类型转换等都作为运算符处理。C 语言的运算类型极其丰富，表达式类型多种多样，计算功能、逻辑判断功能强大，可以实现在其他高级语言中难以实现的运算。

3.3.1 算术运算符及其表达式

C 语言中算术运算符主要用于完成变量的算术运算，如加、减、乘、除等。其含义和优先级如表 3-12 所示。

表 3-12 算术运算符及其作用和优先级

运 算 符	作 用	优 先 级
++	自增 1(变量值加 1)	高
--	自减 1(变量值减 1)	
*	乘法	中
/	除法	
%	求余运算(整数相除，取余数)	
+	加法	低
-	减法	

说明：

(1) 求余%运算的两个操作数都必须是整数，不可以是浮点数。

(2) 如果除法/运算的两个操作数都是整数，则结果为整数，否则结果是浮点数。

(3) 自增运算符(++)、自减运算符(--)可以前置，也可以后置，效果不同。前置++/--的运算规则是：先将变量的值加 1 或减 1，再使用变量的值；后置++/--的运算规则是：先使用变量的值，再将变量的值加 1 或减 1。

(4) 自增、自减(--)运算既可用于整型变量，也可用于浮点型变量，但不可用于常量和表达式。例如，(i+j)++或 5--是不合法的。

(5) 自增、自减运算符的组合原则是自左而右。例如，a+++b 等价于(a++)+b，而不是 a+(++b)。

(6) ++和--常用于循环语句中或指针变量中，使循环控制变量加(或减)1，或使指针上移(或下移)一个位置。

(7) 尽量使用简洁的语句表达自增、自减运算，以免因为开发工具不同而结果不同。如果必须使用复杂表达式，建议使用小括号提高程序可读性，对其他表达式亦然。

(8) 建议在函数的参数中不要使用表达式。

C 语言中算术表达式示例如表 3-13 所示。

表 3-13　C 语言中算术表达式示例

表　达　式	说　　明
2/5	正确，值为 0，整数相除结果为整数
5/2	正确，值为 2，整数相除结果为整数，小数全舍，不做四舍五入
5/2.0	正确，值为 2.5，操作数中有一个是浮点数，结果就是浮点数
2.0/5	正确，值为 0.4，操作数中有一个是浮点数，结果就是浮点数
5%3	正确，值为 2，因为 5 除以 3，商为 1，余数为 2
5%3.5	错误，求余%运算的两个操作数都必须是整数
3%5	正确，值为 3，因为 3 除以 5，商为 0，余数为 3
i=5;i++;	正确，i 值变为 6，将 5 加 1
i=5;--i;	正确，i 值变为 4，将 5 减 1
i=5;j=3*i++;	正确，i 值变为 6，j 值为 15，++后置，先使用变量后将 i 加 1
i=5;j=3*++i;	正确，i 值变为 6，j 值为 18，++前置，先将 i 加 1 后使用变量
i=5.6;j=--i;	正确，i 值变为 4.6，j 值为 4.6，--前置，先将 i 减 1 后使用变量
i=5.6;j=i--;	正确，i 值变为 4.6，j 值为 5.6，--后置，先使用变量后将 i 减 1

【例 3-2】 如已定义 x 和 y 为 double 类型，则表达式 x=1，y=x+3/2 的值是(　　)。

A. 1　　B. 2.0　　C. 2　　D. 2.5

答案：B

解析：3/2 是两个整数相除，结果是整数 1，而不是 1.5；

y=x+3/2=1+1=2，但 y 是浮点型，所以值不是整数 2，而是小数 2.0。

【例 3-3】 大写字母 A 的 ASCII 码是 65，小写字母 a 的 ASCII 码是 97，以下不能将变量 c 中大写字母转换为对应的小写字母的语句是(　　)。

A. c=(c-'A')%26+'a'　　B. c=c+32

C. c=c-'A'+'a'　　D. c=('A'+c)%26 -'a'

答案：D

解析：字符相减表示计算其 ASCII 码的差，整型和字符型可以互换。

3.3.2　关系运算符及其表达式

C 语言中关系运算符主要用于判断条件的表达，其含义和优先级如表 3-14 所示。

表 3-14　关系运算符及其含义和优先级

运　算　符	含　　义	优　先　级
>=	大于等于	高
>	大于	
<=	小于等于	
<	小于	
==	等于	低
!=	不等于	

提示：关系表达式的结果只有两个：真(值为 1)和假(值为 0)，C 语言中没有逻辑型数据，用 1 代表真，0 代表假。

假如有：

```
int a, b, c;
a = (5 > 0);
b = ((29 - 7) == (16 - 6));
c = ((29 - 7) == (16 + 6));
```

则，变量 a 的值为 1，变量 b 的值为 0，变量 c 的值为 1。

C 语言中关系表达式示例如表 3-15 所示。

表 3-15　C 语言中关系表达式示例

表　达　式	说　　明
0 <=1	表达式正确，0 小于等于 1 成立，表达式为真，表达式值为 1
7!='7'	表达式正确，比较数字 7 和字符 7 的 ASCII 码，表达式为真(1)
7==='7'	表达式正确，表达式为假，表达式值为 0
7==7==7	表达式语法没有问题，表达式值为 0。运算过程是：从左向右，先判断 7==7，两数相等，结果为真(1)；再判断 1==7，结果为假，这个结果和我们平时的理解不同
'a'>'A'	表达式正确。比较两个字符的 ASCII 码大小，小写字母的 ASCII 码大于大写字母的 ASCII 码，表达式为真，表达式值为 1
'A'>'a'	表达式正确，表达式为假，表达式值为 0
'a'<'b'<'c'	表达式语法正确，表达式值为 1。运算过程是：从左向右拆分为两个表达式 a'<'b'和 1 < 'c'，第一个表达式为真，结果为 1，用 1 再和'c'的 ASCII 码比较，结果是真，等于 1
7!=8 !=1	表达式语法正确，表达式结果是 0。运算过程是：从左向右，先判断 7!=8 为真，结果为 1；再判断 1!=1，结果为假，而不是判断 7!=8 和 8 !=1

3.3.3　逻辑运算符及其表达式

C 语言中逻辑运算符主要用于判断条件中的逻辑关系，其含义和优先级如表 3-16 所示。

表 3-16　逻辑运算符及其含义和优先级

运　算　符	含　　义	优　先　级
!	逻辑非	高
&&	逻辑与	中
||	逻辑或	低

说明：逻辑表达式和关系表达式一样，结果只有两个：真(值为 1)和假(值为 0)。逻辑运算规则如表 3-17 所示。

表 3-17　逻辑运算规则

A	B	A&&B	A||B	! A
真	真	真	真	假
真	假	假	真	假
假	假	假	假	真
假	真	假	真	真

说明：

(1) 表中的 A 和 B 均可以是其他关系表达式。

(2) 在 C 语言中，任何非 0 值均代表真，0 代表假。

(3) 逻辑运算存在逻辑短路现象。如果表达式结果已经确定，无论后面还有多少表达式，编译器都不会再计算，但会检查语法错误。

C 语言中逻辑表达式示例如表 3-18 所示。

表 3-18　C 语言中逻辑表达式示例

表　达　式	说　　明
(3>4)||(5<6)	表达式正确，(3>4)结果为 0，(5<6)结果为 1，0||1 结果为 1
(3>4)&&(5<6)	表达式正确，(3>4)结果为 0，0 逻辑与任何数结果都为 0，无须计算(5<6)
!2	表达式正确，结果为 0，因为任何非零数都是真
int a=2, b=3, c; c=(a<b)||(a++);	表达式正确，c 等于 1，a 依旧是 2，未运行自增运算。因为(a<b)结果为 1，逻辑或运算结果已经确定，后面表达式只检查语法不计算
int a=2, b=3, c; c=(a>b)&&(++a);	表达式正确，c 等于 0，a 依旧是 2，未运行自增运算。因为(a>b)结果为 0，逻辑与运算结果已经确定，后面表达式只检查语法不计算
int a=2, b=3, c; c=(a>b)||(++a);	表达式正确，c 等于 1，a 是 3，运行了自增运算。因为(a>b)结果为 0，逻辑或运算结果不确定，继续计算++a，++a 结果是 3，为真

3.3.4　位运算符及其表达式

C 语言中，位运算是直接对变量的二进制数按位进行操作。位运算只适合于整型和字符型，不适合于浮点型及其他数据类型，位运算的操作数只有两个：0 和 1。位运算符及其含义和优先级如表 3-19 所示。

表 3-19　位运算符及其含义和优先级

运　算　符	含　　义	优　先　级
~	按位取反	高
&	按位与	低
^	按位异或	低
|	按位或	低
<<	按位左移	中
>>	按位右移	

位运算规则如表 3-20 所示。

表 3-20　位运算规则

A	B	A|B	A^B	A&B	~A	~B
1	1	1	0	1	0	0
1	0	1	1	0	0	1
0	0	0	0	0	1	1
0	1	1	1	0	1	0

有关位运算的详细介绍见后面章节。C 语言中位运算示例如表 3-21 所示。

表 3-21　C 语言中位运算示例

位　运　算	说　　明
short int i, j, k; i=3;j=5;	声明变量并赋值,后面表达式均使用此变量
k=i|j;	表达式正确,k=7,i 的二进制码为 0000 0000 0000 0011,j 的二进制码为 0000 0000 0000 0101,i|j 的二进制码为 0000 0000 0000 0111
k=i^j;	表达式正确,结果为 k=6,i^j 的二进制码为 0000 0000 0000 0110
k=i&j;	表达式正确,结果为 k=1,i&j 的二进制码为 0000 0000 0000 0001
k=~i;	表达式正确,结果为 k=-4,i 的二进制码为 0000 0000 0000 0011,~i 的二进制码为 1111 1111 1111 1100,第一位为 1 表示是负数的补码,转换为原码是 1000 0000 0000 0100,有关补码问题见 2.4 节
k=j>>2;	表达式正确,结果为 k=1,j 右移两位为 0000 0000 0000 0001
k=i<<2;	表达式正确,结果为 k=12,i 左移两位为 0000 0000 0000 1100
k=i>>2.5;	表达式错误,位运算操作数不可以是浮点数
k=(i>>2)/(j>>2);	表达式正确,结果为 k=0,j 右移两位为 1,i 右移两位为 0,0/1=0

3.3.5　条件运算符及其表达式

条件运算符是 C 语言中唯一的三目运算符,由问号“?”和冒号“:”组成。“三目”是指有三个操作数。条件表达式的一般形式为:

表达式 1?　表达式 2:表达式 3;

条件表达式的语法规则:当表达式 1 的值为真(1)时,其结果为表达式 2 的值;当表达式 1 的值为假(0)时,其结果为表达式 3 的值。

提示:表达式 1 通常是关系表达式或逻辑表达式。

条件表达式示例如表 3-22 所示。

表 3-22　条件表达式示例

表　达　式	说　　明
int i=2, j=5; int　x, y, z;	声明变量并赋值,后面表达式均使用此变量
x=(i>j)? i:j;	表达式正确,结果为 x=j=5
y=(i>j)? i+j:i-j;	表达式正确,结果为 y=i-j=2 -5=-3
z=(i<j)? i+j:i-j;	表达式正确,结果为 z=i+j=2 + 5=7

3.3.6　逗号运算符及其表达式

逗号表达式由逗号运算符“,”将两个或多个表达式连接起来。逗号表达式的一般形式为:

表达式 1, 表达式 2, …, 表达式 n;

逗号表达式的语法规则:从前向后,按顺序逐个计算各个表达式,先计算表达式 1,再计

算表达式 2,一直计算到表达式 n；最后结果为表达式 n 的结果。

说明：变量说明和函数参数表中出现的逗号只是作为变量之间的间隔,不能构成逗号表达式。逗号表达式示例如表 3-23 所示。

表 3-23　逗号表达式示例

表　达　式	说　　明
int i=2,x, y, z;	声明变量并赋值,后面表达式均使用此变量
x=i*2, i+3;	表达式正确,结果为 x=i*2=4,i 值不变,依旧是 2,逗号表达式结果没有使用
y=(i*2, i+3);	表达式正确,结果为 y=i+3=5,i 值不变,依旧是 2,逗号表达式结果赋值给 y
z=(i=i+5, i*2, i+8);	表达式正确,结果为 z=i+8=7+8=15,i 值在第一个表达式 i=i+5 中变为 7,第二个表达式 i*2=7*2=14,但此结果没有使用

3.3.7　求字节运算符

sizeof 运算符是求字节数的运算符,它是一个单目运算符,用于返回某数据类型、变量或常量在内存中所占字节的长度。sizeof 运算符的一般形式为：

```
sizeof(数据类型名|变量名|常量)
```

求字节运算表达式示例如表 3-24 所示。

表 3-24　求字节运算表达式示例

表　达　式	说　　明
sizeof(short);	表达式正确,结果为 2,短整数在内存中占 2 字节
float x; sizeof(x);	声明变量 x 表达式正确,结果为 4,x 是单精度浮点型变量,在内存中占 4 字节
sizeof(char);	表达式正确,结果为 1,字符型在内存中占 1 字节
sizeof(2+3.14);	表达式正确,结果为 8,2+3.14 是双精度浮点型常量,在内存中占 8 字节

3.3.8　数据类型转换

C 语言允许表达式中混合有不同类型的常量和变量。C 语言编译器会将较短的数据类型值转换成较长的数据类型值。C 语言中数据类型转换分为自动转换和强制转换两种。

自动转换规则：按数据长度增加的方向转换,将较短的数据类型值转换成较长的数据类型值,以保证数据精度不变。若两种类型字节数不同,则转换成字节数高的类型；若两种类型字节数相同,但一种有符号,一种无符号,则转换成无符号的类型。

强制转换表达式形式：

```
(数据类型符)表达式;
```

或

```
(数据类型符)变量;
```

强制转换规则：将表达式或变量的值临时转换成小括号内指定的数据类型,但不改变变量原来的数据类型。

数据类型转换示例如表 3-25 所示。

表 3-25　数据类型转换示例

表　达　式	说　　明

int i=3, j, k;
float x=2.5, y=3.6;
float z, w;
	声明变量并赋值,后面表达式均使用此变量

j=i * x;	表达式正确,结果为 j=7, 因为 i * x=3 * 2.5=7.5,整数 7 赋给 j
k=i/2;	表达式正确,结果为 k=1, 因为两个整数运算,结果还是整数
y=i * x;	表达式正确,结果为 y=3 * 2.5=7.500000
z=i/2;	表达式正确,结果为 z=1.000000,而不是 1.500000
w=(float)i/2;	表达式正确,结果为 w=1.500000,将 i 强制类型转换为浮点数再参与计算,但 i 的数据类型不会改变,依旧是整型
k=(int)(x+y);	表达式正确,结果为 6,先计算 x+y=2.5+3.6=6.1,再将双精度浮点数 6.1 取整数赋值给 k
k=(int)x+y;	表达式正确,结果为 5,先将 x 强制转换为整型,变为 2,再计算 x+y=2+3.6=5.6,然后再将双精度浮点数 5.6 取整数赋值给 k

3.3.9　运算符优先级及结合性

C 语言共有各类运算符 44 个,按优先级可分为 11 个类别,共 15 个优先级,15 级最高,1 级最低。一般情况下,程序会优先计算优先级高的运算符组成的表达式,用小括号可以改变它们的运行顺序。运算符的优先级与运算方向见表 3-26。

表 3-26　运算符的优先级与运算方向

类　别	运　算　符	名　　称	优先级	结合性
小括号	()	强制类型转换、参数表	15(最高)	自左向右
下标	[]	数组元素下标		
成员	->、.	结构或联合成员		
逻辑	!	逻辑非(单目运算)	14	自右向左
位	~	按位取反(单目运算)		
算术自增、自减	++、--	自增 1,自减 1(单目运算)		
指针	&、*	取地址、取内容(单目运算)		
算术	+、-	正、负号(单目运算)		
长度	sizeof	求字节运算(单目运算)		
算术	*、/、%	乘、除、模(取余)	13	自左向右
	+、-	加、减	12	
位	<<	按位左移	11	
	>>	按位右移		
关系	>=、>	大于等于、大于	10	
	<=、<	小于等于、小于		
	==、!=	相等、不等于	9	
位	&	按位与	8	
	^	按位异或	7	
	|	按位或	6	
逻辑	&&	逻辑与	5	
	||	逻辑或	4	

续表

类　别	运　算　符	名　　称	优先级	结合性
条件	?:	条件(三目运算)	3	自右向左
赋值	=	赋值	2	
自反赋值	+=、-= *=、/= %= &= ^= |= <<= >>=	加赋值、减赋值 乘赋值、除赋值 模赋值 按位与赋值 按位异或赋值 按位或赋值 按位左移赋值 按位右移赋值	2	
逗号	,	逗号运算符	1(最低)	自左向右

说明：

(1) 表 3-26 中出现的自反赋值运算符是由两个运算符组成的，是一种简写方式，只适合于算术运算和位运算。例如，a%=b 就是 a=a%b 的简写；x*=y+7 是 x=x*(y+7)的简写，而不要错误写成 x=x*y+7，因为自反运算优先级低。

(2) 符号“=”为赋值运算符，作用是将赋值运算符右边的表达式的值赋给其左边的变量。在赋值号“=”的左边只能是变量，而不能是常量或表达式。例如，不能写成“2=x;”或“x+y=a+b;”。

(3) 一般而言，单目运算优先级比较高，赋值运算优先级低，逗号运算优先级最低。算术运算优先级高于关系运算和逻辑运算。在一个表达式中有多个优先级相同的运算时，则按照运算符的结合方向进行运算。

(4) C 语言中运算符的结合性分为两种：左结合性(自左向右)和右结合性(自右向左)。多数运算符具有左结合性，单目运算、三目运算和赋值运算符具有右结合性。

例如，算术运算符是左结合性，如有表达式 x+y-z，则从左向右计算，相当于对表达式(x+y)-z 的运算。赋值运算符是右结合性运算符，表达式 x=y=z，应先运行 y=z，后运行 x=y，相当于 x=(y=z)运算。

(5) 建议在复杂表达式中使用小括号来明确表示运算的优先级。

(6) 一个等号“=”是赋值运算符，两个等号“==”是比较是否相等的关系运算符，使用时不要混淆，否则会导致错误。

复杂表达式示例如表 3-27 所示。

表 3-27　复杂表达式示例

表　达　式	说　　明
int i=3, j=3, k=3; int x=1, y=2, z=7;	声明变量并赋值，后面表达式均使用此变量
i==j && j==k;	表达式正确，结果为 1(真)，判断多个值是否相等就应该先两两判断，再用逻辑运算符连接

续表

表 达 式	说 明
i==j==k;	表达式语法无问题，表达式结果为0(假)，是新手经常犯的错误，运算过程是：先判断 i==j 成立，结果为1(真)，再判断 1==k 不成立，结果为0(假)。比较三个数是否相等的正确写法是：i==j && j==k
x=y=z;	表达式语法无问题，这是赋值语句，不是判断是否相等的语句。新手经常将判断相等的两个等号误写为一个等号，此表达式的结果是将 x 和 y 都赋为 z 的值，都变为7，赋值操作正确，表达式结果为真
i+=i-=i*i;	表达式正确，运算结果为 i=-12，运算过程是从右向左：先计算 i=i-(i*i)=3-3*3=-6，再计算 i=i+i=-6 -6=-12
y=++i+j++;	表达式正确，运算结果为 y=7，此表达式可分解为三个表达式：i=i+1=3+1=4；y=i+j=4+3=7；j=j+1=3+1=4
z=--x&&++y;	表达式正确，运算结果为 z=0，x=x-1=0，y=2 值未变。因为逻辑短路，x 值变为0已经能够决定逻辑与运算的结果，因此++y 未运行
5=x+y;	表达式错误，赋值运算符左侧必须是变量
x+2=z;	表达式错误，赋值运算符左侧不可以是表达式
'x'='y';	表达式错误，赋值运算符左侧必须是变量，带单引号的'x'表示字符型常量 x

【例 3-4】 下列叙述错误的是(　　)。

A. C 程序中 #include 和 #define 行均不是 C 语句

B. 除逗号运算符外，赋值运算的优先级最低

C. 在 C 语言中，j++；是赋值语句

D. 在 C 程序中，+、-、×、÷、%是算术运算符，可用于整型和实型运算

答案：D

解析：求余运算的两个操作数必须都是整数，不可以是实数。

【例 3-5】 若变量均已正确定义，以下合法的 C 语言赋值语句是(　　)。

A. x=y==5;　　B. x=n%2.5;　　C. x+n=1;　　D. x=5=4+1;

答案：A

解析：B 错，求余运算中操作数 2.5 不是整数；C 错，只可以为变量赋值，不可以给表达式赋值；D 错，不可以给常量赋值；A 对，结果是 x=0 或者 x=1，取决于 y 是否等于 5。

3.3.10 表达式的书写规则

表达式是由运算符连接常量、变量、函数所组成的式子。每个表达式都按照其中运算符的优先级和结合性进行运算，最终获得一个运算结果，该结果就是表达式的值，该结果的数据类型就是表达式的数据类型。建议在复杂表达式中使用括号明确运算顺序，表达式的书写规则如下：

(1) C 语言表达式中的分界括号都是小括号，大括号和中括号有另外的用途和含义。

(2) 表达式中的乘号不可省略，如 2x+3y 必须写成 2*x+3*y，否则会显示语法出错。

(3) C 语言中比较多个数值大小的表达式应该两两判断，不能沿用数学中的表示方法。例如，数学表达式 x<=y<=z，在 C 语言中应该写为 x<=y && y<=z，否则，虽然语法检

测没有问题，但结果是错误的。

(4) 数学表达式中的一些符号，在 C 语言中应该使用数学函数，例如，对 b^2-4ac 求平方根，在 C 语言中应该使用平方根函数 sqrt()，写为 sqrt(b * b－4 * a * c)。使用数学函数应该先将 math.h 头文件包含到源程序中。

3.4 标准输入输出函数

C 语言的输入、输出操作都是通过调用系统函数来实现。常用的标准输入/输出函数有如下几种。

(1) 格式化输入/输出函数：scanf()/printf()。

(2) 字符输入/输出函数：getc()/putc()、getch()/putch()、getchar()/putchar()。

(3) 字符串输入/输出函数：gets()/puts()。

不同的函数在功能上有所不同，使用时应根据具体的要求选择合适的输入、输出函数。

3.4.1 格式化输出函数

格式化输出函数的作用：按照控制字符串指定的格式，向标准输出设备(一般为显示器)输出指定的输出项。

一般形式：printf("控制字符串"，输出项列表)。

如图 3-4 所示，printf()函数参数包括两部分：<控制字符串>和<输出项列表>。

<输出项列表>可以是常量、变量、表达式，其类型和个数必须与<控制字符串>部分的<格式说明>中的格式字符串的类型和个数一致。有多个输出项时，各项之间用逗号分隔。当<控制字符串>中只有普通字符时，则不需要<输出项列表>。例如，printf("请输入一个整数\n")；是经常用到的输入提示语，直接在屏幕上输出。

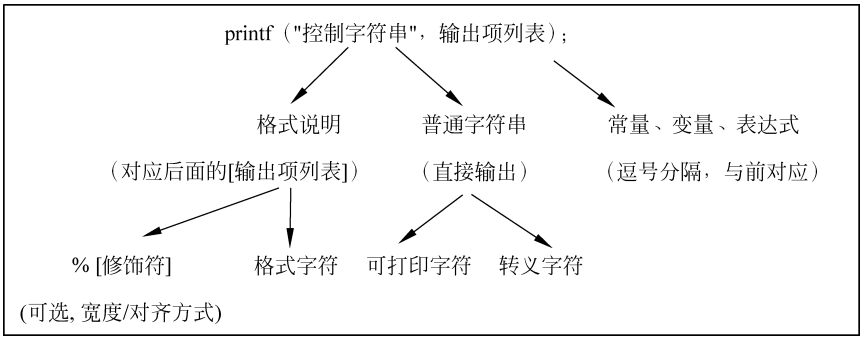

图 3-4 格式化输出函数一般形式

<控制字符串>必须用双引号括起来，可以由<格式说明>和<普通字符串>两部分组成，普通字符串直接在屏幕上输出。

1. 格式说明

格式说明的一般形式为：%[修饰符]格式字符。

格式字符规定了对应输出项的输出格式。在格式字符前面，还可用字母 1 和 h(大小写均可)来说明是用 long 型或 short 型格式输出数据。

常用格式字符如表 3-28 所示。

表 3-28　printf ()函数输出格式字符

格式字符	含　义	格式字符	含　义
d	输出十进制带符号整数	Ld 或 ld	输出十进制带符号长整数
u	输出十进制无符号整数	Hd 或 hd	输出十进制带符号短整数
o	输出八进制无符号整数	Lu 或 lu	输出十进制无符号长整数
x 或 X	输出十六进制无符号整数	Hu 或 hu	输出十进制无符号短整数
f	以小数形式输出单精度浮点数	c	输出单个字符
Lf 或 lf	以小数形式输出双精度浮点数	s	输出字符串
E 或 e	以科学计数法输出浮点数	p	指针类型,输出十六进制地址
G 或 g	按照 e 和 f 格式中较短的输出	%	输出百分号%

[修饰符]是可选的,用于确定数据输出的宽度、精度、小数位数、对齐方式等,默认则按系统默认设定。常用修饰符如表 3-29 所示。

表 3-29　printf ()函数常用修饰符

修饰符	格式说明	含　义
m	%md	以宽度 m 输出整数,不足 m 位时在前面补空格
0m	%0md	以宽度 m 输出整数,不足 m 位时在前面补数字 0
m. n	%m. nf	以宽度 m 输出浮点数,其中小数为 n 位(默认 6 位),小数点占 1 位; 当 m 小于数字实际宽度时,整数按实际宽度输出,小数四舍五入进位; 如省略 m,只写. n,则不限制输出的数字长度,只限制保留 n 位小数
-	%-md %-m. nf	输出的数据左对齐,默认是右对齐。对齐只起到美观的作用,不会影响数据的值
#	%#o	#用于 o 和 x 前,输出的八进制码前面加 0,十六进制码前面加 0x
*	%*d	灵活控制宽度,用常量或变量定义宽度

printf()函数的修饰符、格式符示例如表 3-30 所示。

表 3-30　printf ()函数的修饰符、格式符示例

printf()函数示例	说　明

int i=33, k=3;
char m='A';
float x=12.865;
	声明变量并赋值,后面使用这些变量

printf("%d", i);	输出 33,按实际位数输出十进制带符号整数,占 2 位
printf("%4d", i);	输出　33,输出 4 位十进制带符号整数,前面补 2 个空格
printf("%04d", i);	输出 0033,输出 4 位十进制带符号整数,不足 4 位前面补 0
printf("%1d", i);	输出 33,输出 1 位十进制带符号整数,给出的位数小于数据实际位数时按实际位数输出,实际输出占 2 位
printf("%*d", 5, i);	输出　　33,占 5 位,不足 5 位前面补 3 个空格
printf("%0*d", 5, i);	输出 00033,占 5 位,不足 5 位前面补 3 个 0
printf("%*d", k, i);	输出 33,占 3 位,前面补一空格,因为 k=3
printf("%0*c", 5, m);	输出 0000A,占 5 位

续表

printf 函数示例	说　明
printf("%－5c, ", m);	输出 A　　，占 5 位，－表示数据左对齐，后面 4 个空格
printf("%－05c, ", m);	输出 A　　，占 5 位，－表示数据左对齐，左对齐时修饰符 0 无效，在后面补 4 个空格，不是补 0
printf("%－04d", i);	输出 33　　，占 4 位，－表示数据左对齐，不足 4 位后面补空格，左对齐时修饰符 0 无效，均在后面补空格
printf("%o", i);	输出 41，格式符 o 表示八进制，41 是 33 的八进制数，八进制常量以数字 0 开头，但格式符 o 输出的八进制数不包括 0
printf("%#o", i);	输出 041，格式符 o 表示八进制，加上修饰符 # 表示输出的八进制前面加上数字 0
printf("%x", i);	输出 21，格式符 x 表示十六进制，21 是 33 的十六进制数，十六进制常量以 0x 开头，但格式符 x 输出的十六进制数不包括 0x
printf("%#x", i);	输出 0x21，格式符 x 表示十六进制，加上修饰符 # 表示输出的十六进制数前面加上 0x
printf("%f", x);	输出 12.865000，浮点数默认输出 6 位小数
printf("%7.2f", x);	输出　12.86，数据一共 7 位，小数点占一位，不足 7 位在前面补空格。小数部分四舍五入保留两位小数
printf("%.1f", x);	输出 12.9，格式符中只给出小数位数，四舍五入保留一位小数，总长度没有限制，按数据实际长度输出
printf("%%");	输出%，因为百分号%是格式字符的标志，要想输出一个%，则需要连写两个%

2. 普通字符串

普通字符包括可打印字符和转义字符。可打印字符在屏幕上原样显示，转义字符是不可打印的字符，是一些控制字符，控制产生特殊的输出效果。常用转义字符见表 3-7。例如，'\n'表示回车换行，自动换到新一行；'\t'表示水平制表符，作用是调到下一个制表位。一般情况下，水平制表位的宽度是 8 个字符，'\t'表示移到下一个制表位的列上(8 的倍数＋1)。

printf()函数输出普通字符示例如表 3-31 所示。

表 3-31　printf ()函数输出普通字符示例

printf()函数示例	说　明
printf("请输入一个整数：");	输出“请输入一个整数：”，光标停留在当前行
printf("请输入一个整数：\n");	输出“请输入一个整数：”，光标停留在下一行开始位置
printf("A\tB");	输出“A　B”，A 和 B 中间间隔 7 个空格，凑满一个制表位，在下一个制表位输出 B
printf("ABC\tB");	输出“ABC　B”，C 和 B 中间间隔 5 个空格，凑满一个制表位，在下一个制表位输出 B
printf("A\nB");	输出：A 　　B 分两行输出 A 和 B
printf("\"hello! \"");	输出“"hello!"”，输出的是带双引号的字符串，要输出双引号，需要用\进行转义

续表

printf()函数示例	说　明
printf("\'hello! \'");	输出“'hello!'”,输出的是前后带单号的字符串,要输出单引号,需要用\进行转义
printf("\101");	输出“A”,101是大写字母A的八进制ASCII码,\ddd表示输出八进制ASCII码对应的字符
printf("\x42");	输出“B”,大写字母B的十六进制ASCII码是42,\xhh表示输出两位十六进制ASCII码对应的字符
printf("%%hi!%%");	输出“%hi!%”,因为百分号%是格式字符的标志,要想输出一个%,则需要连写两个%

提示：有符号整数可以用%u格式输出,反之,一个无符号整数也可以用%d格式输出,输出其在内存中存放的数据。

【例3-6】 分析程序运行结果。

```
#include <stdio.h>
void main()
{    int a, b, c;
     a = 25; b = 025; c = 0x25;
     printf("%d %d %d\n", a, b, c);
}
```

运行结果：25 21 37

分析：%d表示输出十进制带符号整数；025表示是八进制数25,转为十进制数是21；0x表示十六进制数,0x25转换为十进制数是37。

【例3-7】 分析程序运行结果。

```
#include <stdio.h>
void main()
{  char c; int n = 100;
   float f = 10;  double x;
   x = f * = n/ = (c = 50);
   printf ("%d, %f\n", n, x);
}
```

运行结果：2，20.000000

分析：自反运算结合性是自右向左，

依次计算c=50；n=n/c=100/50=2；

f=f*n=10*2=20；x=f=20；

输出整数n=2,x=20；

x是浮点数,小数点后默认6个0。

问题：如何在输出浮点数x时只输出两位小数？

解决办法：修改语句为printf ("%d, %.2f\n", n, x);,浮点数格式符f前加宽度修饰符。

3.4.2 格式化输入函数

格式化输入函数的作用：从键盘上按指定格式输入数据，并将输入的数据赋值给指定的变量。

如图 3-5 所示，scanf()函数与 printf()函数类似，也包括两部分：<控制字符串>和<输入项列表>。

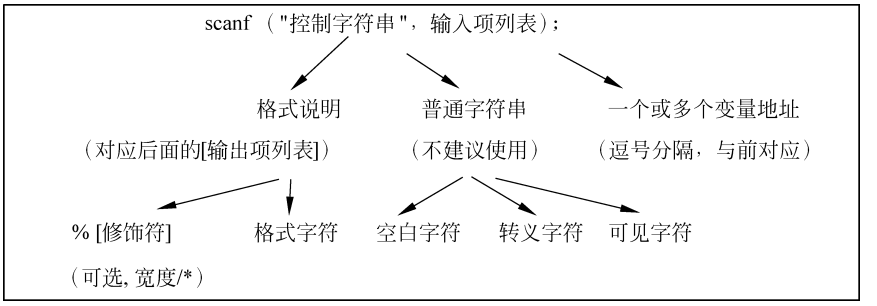

图 3-5 格式化输入函数一般形式

一般形式：scanf("控制字符串",输入项列表)。

<输入项列表>由一个或多个变量的地址组成，变量名前面加上地址符“&”表示变量的地址，多个变量地址之间用逗号“,”分隔(注意：<输入项列表>和 printf()函数的<输出项列表>有区别，<输出项列表>中是变量的名字，不加地址符“&”)。后面章节会学到用数组名和指针可以直接表示地址，可以省略地址符“&”。

如有 int a, b; 则：&a, &b 就可是<输入项列表>。

提示：<输入项列表>中变量的类型和顺序应该与<控制字符串>中格式说明的类型和顺序一致。

<控制字符串>规定了数据的输入格式，字符串必须用双引号括起来，<控制字符串>可以由<格式说明>和<普通字符串>两部分组成。一般情况下不提倡加普通字符串，如果加入了普通字符串，在输入数据时就必须在屏幕上原样输入。

1. 格式说明

格式说明的一般形式为：%[修饰符]格式字符

格式字符规定了对应输入项的格式，表示方法与 printf()函数相同。在格式字符前面可以增加修饰符。scanf()函数输入格式字符如表 3-32 所示。

表 3-32 scanf ()函数输入格式字符

格式字符	含义	格式字符	含义
d	输入十进制带符号整数	ld	输入十进制带符号 长 整数
		hd	输入十进制带符号 短 整数
u	输入十进制无符号整数	lu	输入十进制无符号 长 整数
		hu	输入十进制无符号 短 整数
o	输入八进制无符号整数	lo	输入八进制无符号 长 整数
		ho	输入八进制无符号 短 整数

续表

格式字符	含　义	格式字符	含　义
x	输入十六进制无符号整数	lx	输入十六进制无符号 长 整数
		hx	输入十六进制无符号 短 整数
f	以小数形式输入单精度浮点数	lf	以小数形式输入双精度浮点数
e	以科学计数法输入浮点数		
c	输入单个字符		
s	输入字符串		

修饰符包括字段宽度修饰符和 * 号，还有 l 和 h。字段宽度修饰符限制输入数据的有效范围，* 表示按规定格式输入但不赋值给变量，作用是跳过相应的数据。

修饰符是可选的，主要有以下三种。

1）字段宽度

字段宽度修饰符用数字表示，其作用是限定输入数据的有效范围，超过这个范围则截断数据。

如“scanf("%3d", &a);”，其变量 a 的宽度限定为 3 位，有效值范围为－99～999。若超过宽度，系统会截断，只取前 3 位。

【例 3-8】 分析程序。

```
#include <stdio.h>
int main()
{   int a, b;
    scanf("%d%3d", &a, &b);
    printf("a=%d b=%d\n", a, b);
}
```

运行结果：

输入：123456　123456

输出：a＝123456 b＝123

程序说明：以空格分隔输入两个 123456，变量 a 对应的格式符没有宽度限制，存入 123456，变量 b 对应的格式符限制宽度 3，所以只取前 3 位，存入 123，舍掉后面的 456。

2）l 和 h

字母(L，l)和(H，h)表示输入数据类型的长短，大小写作用相同，具体用法见表 3-32。

(L，l)和整数类型一起使用，表示长整型，和浮点型一起使用表示双精度浮点型。

(H，h)和整数类型一起使用，表示短整型。

3）字符“*”

* 的作用是跳过相应的数据，输入的数据不赋给变量。

【例 3-9】 分析程序。

```
#include <stdio.h>
int main()
{   int x=0, y=0, z=0;
    scanf("%d%*d%d", &x, &y, &z);
    printf("x=%d y=%d z=%d\n", x, y, z);
}
```

运行结果：

输入：11 22 33

输出：x=11 y=33 z=0

程序说明：格式符"%*d"表示跳过一个整数，输入的 22 被跳过，没有赋给任何变量，y=33，z 未读到数据，依旧是原值 0。

2. 普通字符

普通字符包括空白字符、转义字符和可见字符。如果 scanf()控制字符串中有普通字符，则输入时需要原样输入。

1) 空白字符

空格符(Space)、制表符(Tab)或换行符(Enter)都是空白字符，但它们的 ASCII 码是不一样的。

空白字符的作用：对输入的数据起分离作用。

运行 scanf()函数时，输入数据默认用空格符、制表符和换行符作为每个输入值结束的标志，以换行符作为此函数所有数据输入结束的标志。

例如，scanf("%d%d%d", &a, &b, &c)；语句中的控制字符串"%d%d%d"表示输入三个整数，三个%d 格式符之间没有加任何普通字符，则输入数据时可以用一个或多个空格、回车和 Tab 键分隔输入的数据，三个数都输入完毕按回车键结束输入。

若输入的数据中含有字符型的数据时，需要做一些技术处理，否则有可能出错。

例如：

```
int a;      char ch;
scanf("%d%c", &a, &ch);
```

若输入为：64 q

则结果为：a=64，ch=

结果并不是 a=64，ch=q。

若要让结果为：a=64，ch=q

scanf 语句需要改为：scanf("%d%*c%c", &a, &ch)；

修改说明：使用"%*c"格式符跳过中间的空格，空格也是一个字符。

提示：要注意数值型数据和字符型数据的取值特点。若要同时输入这两种类型的数据，可先输入字符型数据，后输入数值型数据，以减少错误的发生。

2) 转义字符：\n、\t

转义字符属空白字符，对输入的数据一般不产生影响，但还是建议在 scanf()控制字符串中不加入除格式符之外的任何字符。

3) 可见字符

可见字符是指 ASCII 码中所有通过键盘输入的可见字符，如数字、字母、其他符号等。但在程序运行时，这些可见字符并不会直接显示在屏幕上，而是要求用户按原样输入，充当每个变量输入值完毕的标志。这时，输入数据的间隔不再使用空格符、制表符和换行符。

再次提示：在 scanf()控制字符串中，不建议使用普通字符；如果使用了，可见字符需要“原样输入”，否则会有不可预料的后果。

scanf()函数的修饰符、格式符示例如表 3-33 所示。

表 3-33　scanf ()函数的修饰符、格式符示例

scanf()函数示例	说　　明

short b;
scanf("%d", &b);
printf("%hd\n", b);
	输入 32767,输出 32767,是短整型的最大值,数据正确
	输入－32768,输出－32768,是短整型的最小值,数据正确
	输入 32790,输出－32746。因为输入值超过 short 范围,数据出错
	输入－32790,输出 32746。因为输入值超过 short 范围,数据出错

int a; scanf("%d", &a);	输入 987654321,a 中存入 987654321,符合 int 范围,数据正确 输入－987654321,a 中存入－987654321,符合 int 范围,数据正确
int a; scanf("%3d", &a);	输入 123456,但 a 中存入的值是 123,只取前 3 位 输入－123456,但 a 中存入的值是－12,取前 3 位,符号占 1 位
int a, b; scanf ("% d% d", &a, &b);	输入：12 34 a 中存入 12,b 中存入 34,数据正确 输入：12，34 a 中存入 12,正确,b 中数据出错。因为 scanf()格式符中没有用逗号“，”这个普通字符,输入数据时不可以用逗号分隔两个数据,可以用空格、Tab 或回车换行符三种格式符分隔
int a, b; scanf("%d, %d", &a, &b);	输入：12 34 a 中存入 12,b 中数据出错。因为 scanf()格式符中有普通字符逗号“，”输入数据时必须用逗号分隔两个数据,不可以用默认的空格、Tab 或回车换行符三种格式符分隔 输入：12，34 a 中存入 12,b 中存入 34,数据正确。scanf()函数格式符中有普通字符,输入数据时必须原样输入
char c; scanf("%c", &c);	输入：hello c 中存入的是 h,字符型变量只能存一个字符 输入：97 c 中存入 9,存的是字符 9,而不是数字 9
char f[10]; scanf("%s", &f);	输入：hello f 中存入 hello,f 是字符型数组,字符型数组可以存字符串,字符型变量只能存一个字符
char f[10]; scanf("%s", f);	输入：hello f 中存入 hello,f 是字符型数组,数组名就表示数组的首地址,可以省略取地址的符号 &
char f[10]; scanf("%c", &f);	输入：hello f[0]中存入 h,并未将字符串 hello 存入数组 f 中,因为使用的格式符是 c,c 表示只读入一个字符
char c; scanf("%d", &c);	输入：97 c 中存入 a,因为 scanf()函数中用的格式符是整型 d,而不是字符型 c。整型和字符型可以互换,为字符型变量输入整型数据,变量中存入将该整数作为 ASCII 码对应的字符
float g; scanf("%f", &g);	输入：12.34 g 中存入 12.34,数据正确 输入：5 g 中存入 5.0,数据正确

续表

scanf()函数示例	说　明
float g; scanf("%d", &g);	输入：12.34 g 中存入 0.0，数据错误。因为格式符 d 表示整型，与数据类型浮点型不符 输入：5 g 中存入 0.0，同样错误，数据类型与格式符必须相符，只有整型和字符型可以互换
double e; scanf("%lf", &e);	输入：12.34 e 中存入 12.34，数据正确 输入：5 e 中存入 5.0，数据正确
double e; scanf("%f", &e);	输入：12.34 e 中数据错误。因为格式符不匹配，双精度浮点数应该用格式符 lf

float g;
scanf("%.2f", &g);
printf("g=%.2f", g);
	输入 1.22，输出结果却不是 1.22。因为 scanf()函数中不可以加入"%.2f"表示小数位数的修饰符，该修饰符应该用在 printf()函数中

【例 3-10】 分析程序运行结果。

```
#include <stdio.h>
void main()
{   int a, b;
    printf("请输入: ");
    scanf("%3d%3d", &a, &b);
    printf("a=%d, b=%d\n", a, b);
}
```

运行结果：

请输入：112345678

a=112，b=345

程序说明：scanf()函数中两个格式符都限定了 3 位的宽度，系统会自动截取数据，多余的数据无效。此方法也可用于字符型数据。

【例 3-11】 分析程序运行结果。

```
#include <stdio.h>
void main()
{   int a, b;
    printf("请输入: ");
    scanf("a=%d, b=%d", &a, &b);
    printf("a=%d, b=%d\n", a, b);
}
```

运行结果：

错误的输入：

请输入：100 4568

a=-858993460，b=-858993460

正确的输入：

请输入：a＝100，b＝4568

　　a＝100，b＝4568

程序说明：建议在 scanf()函数中不要加普通字符；如有加普通字符，输入数据时必须原样输入，否则读入的数据会出错。

3.4.3 字符输出函数

C 语言中有专门处理字符型数据的输出函数。表 3-34 中为几种常用的字符输出函数。VC 编译器中，putchar()和 putc()函数在 stdio.h 头文件中定义，putch()函数在 conio.h 头文件中定义。

表 3-34　常用字符输出函数

函数原型	函数功能	返回值
int putc(int ch, FILE * stream)	将 ch 所对应字符输出到 stream 指定的文件流中，stdout 表示屏幕，也可以输出到其他文件流中	成功：ch 失败：EOF
int putch(int ch)	将 ch 所对应字符输出到屏幕	成功：ch 失败：EOF
int putchar(int ch)	将 ch 所对应字符输出到屏幕	成功：ch 失败：EOF

putch()和 putchar()的功能都是向屏幕输出一个字符，与 printf()函数中%c 的功能相同，但 putch()是一个函数，而 putchar()其实不是函数，而是函数 putc(ch, stdout)的一个宏。putchar(ch)与 putc(ch, stdout)功能完全相同，putc()函数具有更多的功能。

putc(ch, stdout)、putch(ch)和 putchar(ch)中的 ch 参数可以是字符常量、字符变量和整型表达式，也可以是控制字符或转义字符。例如 putch('\n')和 putchar('\\')等，都是合法的。

如果参数 ch 是整型表达式，要求其表达式的值在 ASCII 码表有效范围内，输出的是该值作为十进制 ASCII 码对应的字符。例如 putc(97, stdout);、putch(98);和 putchar(99);，分别输出小写字母 a、b、c。

【例 3-12】 分析程序运行结果。

```
#include <stdio.h>
#include <conio.h>
int main()
{
   char c1 = 'X', c2 = 'Y', c3 = 'Z';
   int d1 = 97, d2 = 98;
1  putc(c1, stdout);                  //输出字符变量
2  putch(c2);
3  putchar(c3);
   putch('\n');                       //换行
4  putch('x');                        //输出常量
   putchar('\n');                     //换行
5  putchar(d1);                       //输出整数 ASCII 对应字符
```

```
6  putch(d2);
}
```

运行结果：


```
XYZ
x
ab
```

程序说明：语句 1、2、3 用三个不同函数输出三个变量的值，输出结果为 XYZ。语句 4 输出字符型常量 x。语句 5、6 输出整数 97、98 作为 ASCII 码对应的字母 a 和 b。

提示：使用 putch()函数需要加入头文件 conio.h。

【例 3-13】 分析程序运行结果。

```
#include <stdio.h>
int main()
{  int a; char b;
   b = 'b';
   a = b + 1;
   putchar(a);
   putchar('\n');
   putchar(b);
}
```

运行结果：

```
c
b
```

程序说明：字符型和整型可以互换，字符型＋1 表示取该字符在 ASCII 码表中下一个字符，小写字母 b 的下一个字符是小写字母 c。

3.4.4 字符输入函数

与字符输出函数相对应，C 语言也专门提供了专用的字符输入函数，用于读入字符，存在字符型变量中。表 3-35 列出几种常用的字符输入函数。其中只有 getch()函数在 conio.h 头文件中定义，而不是在 stdio.h 头文件中定义。

表 3-35 常用的字符输入函数

函数原型	函数功能	返回值
int getc(FILE * stream);	从指定的输入流 stream 中读取字符，stdin 表示从键盘读入，也可以从其他输入流读入	成功：字符 失败：－1
int getch();	将键盘输入的字符放入缓冲区，输入的字符不显示在屏幕上	成功：字符 失败：－1
int getchar();	将键盘输入的字符放入缓冲区，输入的字符会显示在屏幕上，须回车	成功：字符 失败：－1

与字符输出函数相对应，getchar()是 getc(stdin)函数的一个宏，它们的功能完全相同，但 getc()函数具有更多的功能。

getch()和 getchar()函数的功能都是读入一个从键盘输入的字符，存到字符型变量中，不需要输入参数，它们的功能与 scanf()函数中%c 的功能相同，但也有所区别。使用字符输入函数需要注意以下几个方面。

(1) 用字符输入函数接收字符时，并不是从键盘输入一个字符后立即响应，而是将输入的内容先读入缓冲区，待输入结束后再一并运行。

(2) 字符输入函数一次只能接受一个字符，如果是多个输入，只有第一个字符有效。

(3) 使用 getch()函数输入字符后，输入的字符不会显示在屏幕上；而使用 getchar()和 getc()输入，屏幕上会显示输入的字符。

(4) getch()函数输入字符后，不用回车，直接读入；而使用 getchar()和 getc()输入，需要用回车符结束输入，会先将输入内容存入缓冲区，回车之后再读入。

(5) 与 scanf()输入函数不同，字符输入函数将空格符、制表符、换行符也作为字符接收；而 scanf()函数把空格符、制表符、换行符作为输入数据的分隔符，不能读入。

【例 3-14】 分析程序。

```
#include <stdio.h>
void main()
{
  char c1, c2;
  c1 = getch();
  putchar(c1);putchar('\n');
  c2 = getchar();
  putchar(c2);putchar('\n');
}
```

运行结果：

```
x
y
y
```

程序说明：

(1) 运行程序，输入字母 x，屏幕上立即显示 x，并将光标移动到下一行。这个 x 是 putchar(c1)；语句运行的结果，使用 getch()语句输入字符时不需要按下回车键结束输入，但输入的字符也不在屏幕上显示。

(2) 输入字母 y，屏幕上显示字母 y，光标停留等待按下回车键结束输入。回车后屏幕上又显示一个字母 y，第二个 y 是 putchar(c2)；语句运行的结果。使用 getchar()函数读数据，输入的字母在屏幕上显示，但需要按下回车键结束。

【例 3-15】 分析如下程序。

```
#include <stdio.h>
int main()
{
  char c1, c2, c3;
```

```
    printf("请输入字符 c1:");
    c1 = getc(stdin);                    //输入字符存入 c1
    putc(c1, stdout);                    //输出 c1 的值
    printf("\n 请输入字符 c2:");
    c2 = getchar();                      //输入字符存入 c2
    putchar(c2);                         //输出 c2 的值
    printf("\n 请输入字符 c3:");
    c3 = getch();                        //输入字符存入 c3
    putch(c3);                           //输出 c3 的值
}
```

运行结果：

```
请输入字符 c1:A
A
请输入字符 c2:

请输入字符 c3:B
```

程序说明：

(1) 程序运行，显示“请输入字符 c1:”，然后运行 c1=getc(stdin)语句，等待用户输入。

(2) 输入字符 A(在屏幕上显示)，按回车键结束，屏幕上输出字符 A；同时输出“请输入字符 c2:”和一个空行，紧接着又输出“请输入字符 c3:”，此时才停下来等待输入。这是因为输入的字符 A 和回车符都被存入缓冲区，按下回车键结束输入后，语句 c1=getc(stdin)；读取缓冲区中的 A，语句 putc(c1, stdout)；输出 A；之后，语句 c2=getchar()；读到缓冲区里的回车符，语句 putchar(c2)；输出回车符(一个空行)。

(3) 输入字符 B(屏幕上不显示输入的内容)，输出 B(这是 putch(c3);语句运行的结果)。

【例 3-16】 分析如下程序。

```
#include <stdio.h>
int main()
{  char ch;
   while ((ch = getchar()) == '0')
     printf("#");
}
```

输入：1230

运行结果：

输入：0123

运行结果：#

输入：00123

运行结果：# #

程序说明：getchar()函数每次只能读入一个字符，读入后判断是否是字符 0，是，则输出 # 号，再读下一个字符进行判断；不是，则退出循环。

提示：程序设计时，根据情况选择适合的输入输出函数。使用字符输入输出函数时，一般会考虑配对使用。经常会将字符输入函数与循环条件语句合并为一个语句，如例 3-16。

3.4.5 字符串输出函数

C 语言提供专门输入输出字符串的函数，输出字符串的函数是 puts()。

一般形式：int puts(字符串或字符数组)；

作用：将字符串或字符数组中存放的字符串输出到终端(显示器)上，一次只能输出一个字符串。输出时，遇到'\0'结束，并换行。

【例 3-17】 分析程序运行结果。

```
#include <stdio.h>
int main()
{   char a[10] = "1234\0abcd";
    puts(a);
    puts("Hello!\0How are you?");
    puts("OK!\nI see!");
}
```

运行结果：

```
1234
Hello!
OK!
I see!
```

程序说明：'\0'为字符串结束标志，遇到'\0'结束输出，并换行。'\0'是字符串结束标志，函数 puts(a)遇到'\0'则换行结束输出，后面的内容不再输出。

3.4.6 字符串输入函数

C 语言提供的专门输入字符串的函数是 gets()。

一般形式：gets(字符数组)；

作用：专门用于读字符串，从标准设备(一般是键盘)读入字符串(包括空格)，直到遇回车符结束。

【例 3-18】 分析程序运行结果。

```
#include <stdio.h>
int main()
{   char a[10], b[10];
    puts("请输入字符串 1: ");
    gets(a);
    puts(a);
    printf("请输入字符串 2: ");
    scanf("%s", b);
    printf("%s\n", b);
}
```

运行结果：

```
请输入字符串 1:
abc 1234
abc 1234
请输入字符串 2: abc 1234
abc
```

程序说明：gets()函数能读入空格，scanf()函数不可以；puts()函数自动换行，printf()函数需用\n 换行。

提示：gets()函数可以用于读入带空格的字符串，但一次只能读入一个字符串，而且必须确保输入的字符串长度(个数)小于字符数组大小。而 scanf()函数不能读入空格，它将空格作为字符串结束的标志。

3.5 程 序 范 例

【例 3-19】 分析程序运行结果。

```
#include <stdio.h>
void main()
{
  int a;char c = 10;
  float f = 100.0;double x;
  a = f/ = c * = (x = 6.5);
  printf("%d %d %3.1f %3.1f\n", a, c, f, x);
}
```

运行结果：1　65　1.5　6.5

程序说明：这是一道混合运算题，事先需要知道运算符的优先级和结合性，查看表 3-26 知道赋值运算和自反运算的优先级相同，结合性是自右向左。先计算 x＝6.5，然后依次计算

c＝c * x＝10 * 6.5＝65；

f＝f/c＝100/65≈1.54；

a＝f＝(1.54)取整数＝1。

提示：注意数据类型变化。浮点数存入整型变量，只取整数，不进行四舍五入。

【例 3-20】 分析程序运行结果。

```
#include <stdio.h>
void main()
{ int x = 11, y = 22;
  printf("%d\n", (x, y));
}
```

运行结果：22

程序说明：(x, y)表示逗号表达式，结果为最后一个表达式的值。如果去掉小括号，改为 printf("%d\n", x, y);，则输出结果是 11，后一个变量 y 没有对应格式符，不输出。

【例 3-21】 分析程序运行结果。

```
#include <stdio.h>
void main()
{
  int x = 'd';
  printf("%c\n", 'Y' - (x - 'a' + 1));
}
```

运行结果：U

程序说明：这是字符型和整型转换的问题。x－'a'表示将变量 x 中的字符与字符 a 的 ASCII 码相减，等于 3；字符 Y 减去整数 4 表示比大写 Y 的 ASCII 码小 4 的那个字符，就是大写字符 U。

【例 3-22】 分析程序运行结果。

```
#include <stdio.h>
void main()
{   char c1, c2, c3, c4, c5, c6;
    printf("请输入：");
    scanf("%c%c%c%c",&c1,&c2,&c3, c4);
    c5 = getchar();
    c6 = getchar();
    putchar(c1);
    putchar(c2);
    printf("%c%c\n", c5, c6);
}
```

运行结果：

请输入：123 45678

输出：1245

程序说明：格式符%c 表示读入字符型数据，每次只读入一个字符。c1、c2、c3 读入的分别是 1、2 和 3，c4 读入的是空格，c5 和 c6 分别读入 4 和 5。

提示：使用 scanf()函数输入多个数值型数据时，可以默认用空格、Tab 制表符和回车键分隔数据。但输入字符型数据时，这三个符号都会作为一个字符读入，不能作为分隔符。

【例 3-23】 分析程序运行结果。

```
/*求三个整数的平均值*/
#include <stdio.h>
void main()
{
    int a, b, c;
    float average;
    printf("请输入三个数:");
    scanf("%d%d%d", &a, &b, &c);
    average = (a + b + c)/3.0;
    printf("平均分：%-7.2f\n", average);
}
```

运行结果：

请输入三个数：11 15 18

平均分：14.67

问题：

如果改为 average=(a+b+c)/3;是什么结果？为什么要除以 3.0？输出格式符“%-7.2f”是什么意思？如果去掉负号“-”是什么效果。

3.6 常见错误

【案例 3-1】 输出格式问题。

```
#include <stdio.h>
int main()
{   int a = 2, b = 3;
    printf("a = %ad, b = %d", a, b);
}
```

输出结果：a=ad，b=2

而不是：a=2，b=3

错误分析：输出格式符%ad 中间多了一个普通字符 a。

修改为：printf("a=%d，b=%d"，a，b)；。

【案例 3-2】 输出格式问题。

```
#include <stdio.h>
int main()
{   int D = 2;
    printf("D = %D", D);
}
```

输出结果：D=D

而不是：D=2

错误分析：整型数输出格式符是小写字母 d,不能写成大写字母。

修改为：printf("D=%d"，D)；。

【案例 3-3】 缺少地址符问题。

```
#include <stdio.h>
void main()
{   int a, b;
    printf("请输入：");
    scanf("%d%d", a, b);
    printf("a = %d, b = %d\n", a, b);
}
```

运行结果：编译、组建都没有问题,但运行程序输入数据后出现异常,程序停止运行。

错误分析：scanf()函数中的输入项列表必须是变量地址,不可以直接写变量名。printf()函数的输出项列表是直接写变量名,注意两个函数语法有区别。这是初学者的常见错误!

修改为：scanf("%d%d"，&a，&b)；。

本章小结

本章是C语言程序设计的基础内容，后面的章节都要用到本章内容。编写C语言程序时应注意以下几项。

(1) 标识符不要使用关键字，尽量“见名知意”，提高程序的可读性。

(2) 变量必须先定义后使用。变量有三要素：变量名、变量类型、变量值。变量名对应在内存中的地址，变量类型决定在内存中分配的字节数，变量值存储在分配的内存空间中。

(3) 不同类型变量有不同取值范围，避免不同类型变量之间赋值，以免数据丢失，造成程序运行结果错误，甚至有更严重的情况发生。

(4) 正确使用格式化输入输出函数printf()和scanf()，格式符要与数据类型对应，以免造成数据错误。

(5) 输入输出字符型、字符串数据时也可以使用专门的函数，字符输入输出函数有getc()/putc()、getch()/putch()和getchar()/putchar()，字符串的输入输出函数是gets()/puts()。其中getch()/putch()定义在conio.h头文件中，其他都定义在stdio.h头文件中。

(6) 代码书写要规范，语句缩进，适当使用注释。

(7) 正确使用运算符，复杂的表达式尽量用括号来区分优先级，增加程序可读性。

(8) 关系表达式和逻辑表达式常用于分支结构和循环结构中的条件判断。

习　　题

一、选择题

1. 以下标识符不是关键字的是(　　)。

A. SWITCHS　　B. break　　C. return　　D. char

2. 以下不能定义为用户标识符的是(　　)。

A. Main　　B. _0　　C. _int　　D. sizeof

3. 以下不合法的标识符是(　　)。

A. J2_KEY　　B. Double　　C. 4d　　D. _g

4. 按照C语言规定的用户标识符命名规则，不能出现在标识符中的是(　　)。

A. 大写字母　　B. 连接符　　C. 数字字符　　D. 下画线

5. 以下选项中，不能作为合法常量的是(　　)。

A. 1.234e05　　B. 1.234e0.5

C. 1.234e+5　　D. 1.234e0

6. 以下选项中合法的C语言常量的是(　　)。

A. -80　　B. -080　　C. -8e1.0　　D. -80.0e

7. 下面选项中，合法的一组数值常量的是(　　)。

A. 028　　B. 12　　C. 177　　D. 0x8a

.5e-3　　0xa23　　4c1.5　　10,000

-0Xf　　4.5e0　　0abc　　3.e5

8. 下面 4 个选项中，均是合法整型常量的选项是(　　)。

A.	B.	C.	D.
160	−0xcdf	−01	−0x48a
−0xffff	01a	986，012	2e5
011	0xe	0668	0x

9. 下面 4 个选项中，均是不合法浮点数的选项是(　　)。

A.	B.	C.	D.
160.	123	−.18	−e3
0.12	2e4.2	123e4	.234
E3	.e5	0.0	1e3

10. 下面正确的字符常量是(　　)。

A. "C"　　B. "\\"　　C. 'W'　　D. ''

11. 下列 4 组选项中，均不是 C 语言关键字的选项是(　　)。

A.	B.	C.	D.
define	getc	include	while
IF	char	scanf	go
type	printf	case	pow

12. 下列 4 组选项中，均是 C 语言关键字的选项是(　　)。

A.	B.	C.	D.
auto	switch	signed	if
enum	typedef	union	struct
include	continue	scanf	type

13. C 语言中的标识符只能由字母、数字和下画线 3 种字符组成，且第一个字符(　　)。

A. 必须为字母

B. 必须为下画线

C. 必须为字母或下画线

D. 可以是字母、数字和下画线中任一种字符

14. 以下叙述中正确的是(　　)。

A. 在 C 语言中，main 函数必须位于程序的最前面

B. C 语言的每行中只能写一条语句

C. C 语言本身没有输入输出语句

D. 在对一个 C 程序进行编译的过程中，可以发现注释中的拼写错误

15. 以下叙述中正确的是(　　)。

A. C 程序中注释部分可以出现在程序中任意合适的地方

B. 大括号{}只能作为函数体的定界符

C. 构成 C 程序的基本单位是函数，所有函数名都可以由用户命名

D. 分号是 C 语句之间的分隔符，不是语句的一部分

16. 以下叙述中不正确的是(　　)。

A. 一个 C 源程序可由一个或多个函数组成

B. 一个 C 源程序必须包含一个 main 函数

C. 在 C 程序中，注释说明只能位于一条语句的后面

D. C 程序的基本组成单位是函数

17. 在C语言中,int、char和short三种类型数据在内存中所占用的字节数(　　)。

A. 由用户自己定义　　B. 均为2字节

C. 是任意的　　D. 由所用机器的机器字长决定

18. 以下不属于C语言的数据类型是(　　)。

A. unsigned long int　　B. long short

C. unsigned int　　D. signed short int

19. 以下正确的字符串常量是(　　)。

A. "\\\"　　B. 'abc'

C. Olympic Games　　D. " "

20. 以下4个程序完全正确的是(　　)。

A.
```
#include <stdio.h>
int main()
{/*programming/*
    printf("programming! \n");}
```

B.
```
#include <stdio.h>
    int main()
{/*/programming/*/
printf("programming! \n");}
```

C.
```
#include <stdio.h>
int main()
{/*/*programming*/*/
    printf("programming! \n");}
```

D.
```
#include <stdio.h>
    int main()
{/*programming*/
printf("programming! \n");}
```

21. 以下正确的叙述是(　　)。

A. 在C语言中,每行中只能写一条语句

B. 若a是实型变量,C程序中允许赋值a=10,因此实型变量允许存放整数

C. 在C程序中,无论是整数还是实数,都能被准确无误地表示

D. 在C程序中,%是只能用于整数运算的运算符

22. 逗号表达式"(a=3*5, a*4), a+15"的值是(　　)。

A. 15　　B. 60　　C. 30　　D. 不确定

23. 以下定义和语句的输出结果是(　　)。

```
char c1 = 'a', c2 = 'f';
printf("%d, %c\n", c2 - c1, c2 - 'a' + 'B');
```

A. 2, M　　B. 5, !　　C. 2, E　　D. 5, G

24. 若有以下定义,则能使值为3的表达式是(　　)。

```
int k = 7, x = 12;
```

A. x%=(k%=5)　　B. x%=(k-k%5)

C. x%=k-k%5　　D. (x%=k)-(k%=5)

25. 假设所有变量均为整型,则表达式x=(a=2,b=5,b++,a+b)的值为(　　)。

A. 7　　B. 8　　C. 6　　D. 2

26. 假设所有变量均为整型,则表达式x=(i=4, j=16, k=32)后x的值为(　　)。

A. 4　　B. 16　　C. 32　　D. 52

27. 若有代数式$|x^3+\log_{10}^x|$，则正确的 C 语言表达式是(　　)。

A. fabs(x * 3+log(x))　　B. fabs(pow(x, 3)+log(x))

C. abs(pow(x, 3.0)+log(x))　　D. fabs(pow(x, 3.0)+log(x))

28. 设变量 n 为 float 类型，m 为 int 类型，则以下能实现将 n 中的数值保留小数点后两位，第三位进行四舍五入运算的表达式是(　　)。

A. n=(n * 100+0.5)/100.0　　B. m=n * 100+0.5, n=m/100.0

C. n=n * 100+0.5/100.0　　D. n=(n/100+0.5) * 100.0

29. 若变量均已正确定义并赋值，以下合法的 C 语言赋值语句是(　　)。

A. x=n%2.5　　B. x=5=4+1

C. x+n=i　　D. x=y==5

30. 以下运算符中优先级最高的是(　　)。

A. <　　B. +　　C. &&　　D. !

31. putchar 函数可以向终端输出一个(　　)。

A. 字符或字符变量值　　B. 字符变量值

C. 字符串　　D. 整型变量表达式

32. 已知字符'A'的 ASCII 码是 65，字符变量 c1 的值是'A'，c2 的值是'D'，运行语句 printf("%d, %d", c1, c2-2);后，输出结果是(　　)。

A. A, B　　B. A, 68　　C. 65, 66　　D. 65, 68

33. 以下不正确的叙述是(　　)。

A. 在 C 语言中，逗号运算符的优先级最低

B. 在 C 语言中，APH 和 aph 是两个不同的变量

C. 若 a 和 b 类型相同，运行赋值表达式 a=b 后，b 中的值将放入 a 中，而 b 中的值不变

D. 从键盘输入数据时，整型变量只能输入整型数值，实型变量只能输入实型数值

34. 以下程序的运行结果是(　　)。

```
#include <stdio.h>
int main()
{   unsigned int x = 0xffff;
    printf("%u\n", x);   }
```

A. -1　　B. 65535　　C. 32767　　D. oxFFFF

35. 设 x, y 为整型变量，x=11, y=5，则语句 printf("%d, %d\n", x++, --y);的输出结果是(　　)。

A. 11, 5　　B. 12, 4　　C. 11, 4　　D. 12, 5

36. 以下程序的运行结果是(　　)。

```
#include <stdio.h>
void main()
{ char x = 'd';     printf("%c\n", x + 10);}
```

A. n　　B. d　　C. d+10　　D. 100

37. 有定义语句：int x, y; scanf("%d, %d", &x, &y);，若变量 x 得到值 11，y 得到值 12，下面四组输入中错误的是(　　)。(**多选**)

A. 11 12 <回车>　　B. 11，12 <回车>

C. 11，<回车> 12 <回车>　　D. 11 < Tab > 12 <回车>

38. 运行以下程序，若从键盘上输入数据，使 m＝123，n＝456，p＝789，则正确的输入是(　　)。

```
#include <stdio.h>
int main()
{ int m, n, p;
  scanf("m = %dn = %dp = %d", &m, &n, &p);
  printf("%d%d%d\n", m, n, p);
}
```

A. m＝123n＝456p＝789　　B. m＝123 n＝456 p＝789

C. m＝123，n＝456，p＝789　　D. 123 456 789

39. 假设变量均已正确定义，若要通过 scanf("%d%c%d%c", &a1, &c1, &a2, &c2); 语句为变量 a1，a2 赋数值 10 和 20，c1，c2 赋字符 X 和 Y，以下所示输入形式中正确的是(　　)。

A. 10 X　20 Y <回车>　　B. 10 X20 Y <回车>

C. 10 X <回车>　　D. 10X <回车>

20 Y <回车>　　20Y <回车>

40. 若要求从键盘上读入含有空格的字符串，应使用的函数是(　　)。

A. getc()　　B. gets()　　C. getchar()　　D. scanf()

41. 对以下程序，当输入 a <回车>后，下面叙述正确的是(　　)。

```
#include <stdio.h>
int main()
{  char c1 = '1', c2 = '2';
   c1 = getchar();    c2 = getchar();
   putchar(c1);      putchar(c2);  }
```

A. 变量 c1 被赋予字符 a，c2 被赋予回车符

B. 程序等待用户输入第 2 个字符

C. 变量 c1 被赋予字符 a，c2 仍然是原字符 2

D. 变量 c1 被赋予字符 a，c2 无确定值

42. 设 a、b 和 c 都是 int 型变量，且 a＝3，b＝4，c＝5 则下面的表达式中，值为零的表达式是(　　)。

A. 'a' && 'b'　　B. a <＝b

C. a || b＋c&&b－c　　D. !((a < b)&&! C|| 1)

43. 设 ch 是 char 型变量，其值为 A 字符，如下表达式 ch 的值是字符(　　)：

```
ch = (ch >= 'A'&& ch <= 'Z') ? ( ch + 32):ch;
```

A. A　　B. a　　C. Z　　D. z

44. 若 x 和 y 都是 int 型变量，x＝100，y＝200，且有程序段 printf("%d", (x,y));，则输出结果是(　　)。

A. 200　　B. 100

C. 100,200　　D. 输出格式不够,输出不确定的值

45. C 语言中最简单的数据类型包括(　　)。

A. 整型、实型、逻辑型　　B. 整型、单精度型、双精度、字符型

C. 整型、字符型、逻辑型　　D. 整型、实型、逻辑型、字符型

46. C 语言中,运算对象必须是整型的运算符是(　　)。

A. %　　B. /　　C. %和 /　　D. **

47. 数学关系 x≥y≥z,它对应 C 语言表达式是(　　)。

A. (x>=y)&&(y>=z)　　B. (x>=y) AND (y>=z)

C. (x>=y>=z)　　D. (x>=y) & (y>=z)

48. 设有"int x=10,y=3,z;",则语句 printf ("%d\n",z=(x%y,x/y));的输出结果是(　　)。

A. 1　　B. 0　　C. 4　　D. 3

49. C 语言表达式 10 !=9 的值是(　　)。

A. true　　B. 非零值　　C. 0　　D. 1

50. 合法的 C 语言字符常量是(　　)。

A. '\t'　　B. "A"　　C. 65　　D. A

二、填空题

1. 字符变量以(　　)类型表示,它在内存中占(　　)位。

2. 在一个 C 语言源程序中,多行注释的分界符分别为(　　)和(　　)。

3. C 语言中的标识符只能由三种字符组成,它们是(　　)、(　　)和(　　),且第一个字符必须为(　　)。

4. 若 a 是 int 型变量,则运行表达式 a=25/3%3 后 a 的值为(　　)。

5. C 程序中数据有常量和变量之分,其中,用一个标识符代表一个常量的,称为(　　)常量。C 语言规定在程序中对用到的所有数据都必须指定其数据类型,对变量必须做到先(　　),后使用。

6. C 语言中,用关键字(　　)定义单精度实型变量,用关键字(　　)定义双精度实型变量,用关键字(　　)定义字符型变量。

7. 在 C 语言中,以 16 位 PC 机为例,一个 char 型数据在内存中所占的字节数为(　　);一个 int 型数据在内存中所占的字节数为(　　),则 int 型数据的取值范围为(　　)。一个 float 型数据在内存中所占的字节数为(　　);一个 double 型数据在内存中所占的字节数为(　　)。

8. 设 C 语言中的一个基本整型数据在内存中占 2 字节,若欲将整数 135791 正确无误地存放在变量 a 中,应采用的类型说明语句是(　　)。

9. C 的字符常量是用(　　)引号括起来的 1 个字符,而字符串常量是用(　　)引号括起来的字符序列。

10. C 语言中,用'\'开头的字符序列称为转义符。转义符'\n'的功能是(　　);转义符'\r'的功能是(　　)。

11. 若有定义 char c='\010';,则变量 c 中包含的字符个数为(　　)。

12. C语言中，& 作为双目运算符是表示的是(　　)，而作为单目运算符时表示的是(　　)。

13. 在C语言的赋值表达式中，赋值号左边必须是(　　)。

14. 自增运算符＋＋、自减运算符－－，只能用于(　　)，不能用于常量或表达式。＋＋和－－的结合方向是“自(　　)至(　　)”。

15. 在C语言中，格式化输入操作是由库函数(　　)完成的，格式化输出操作是由库函数(　　)完成的。

16. 写出下列数所对应的其他进制数(D表示十进制数，B表示二进制数，O表示八进制数，H表示十六进制数)。

32_D＝(　　)$_B$＝(　　)$_O$＝(　　)$_H$

17. 写出下列数所对应的其他进制数。

75_D＝(　　)$_B$＝(　　)$_O$＝(　　)$_H$

18. 假设已指定i为整型变量，f为float变量，d为double型变量，e为long型变量，有式子10＋'a'＋i＊f－d/e，则结果为(　　)型。

19. 若有定义int x＝3，y＝2；float a＝2.5，b＝3.5；，则表达式(x＋y)%2＋(int)a/(int)b的值为(　　)。

20. 5/3的值为(　　)，5.0/3的值为(　　)。

21. 若有以下定义，int m＝5，y＝2；，则运行表达式y＋＝y－－＝m＊＝y后的y值是(　　)。

22. 若a是int型变量，则表达式(a＝4＊5，a＋2)，a＋6的值为(　　)。

23. 若x和n均为int型变量，且x的初值为12，n的初值为5，则运行表达式x%＝(n%＝2)后，x的值为(　　)。

24. 表达式8/4＊(int)2.5/(int)(1.25＊(3.7＋2.37))值的数据类型为(　　)。

25. 假设m是一个三位数，从左到右用a，b，c表示每一位的数字，若从左到右数字是bac的三位数，则在C语言中用m表示bac的表达式是(　　)。

26. 运行语句(a＝3.0＋5，a＊4)，a＋＝－6；后，变量a及表达式的值分别为(　　)和(　　)。

27. 以下语句的运行结果是(　　)。

```
int a = 1;
printf (" % d\\ % s\\ % s", a, "abc", "def");
```

28. getchar()函数的作用是(　　)。

29. 运行以下程序后，用户输入123456abc，输出结果为(　　)。

```
# include < stdio. h >
void main()
  {   int a, b;
      char  c;
      scanf (" % 3d % 2d % 3c", &a, &b, &c);
      printf(" % d,  % d,  % c", a, b, c);
  }
```

30. 以下程序的运行结果是(　　)。

```
# include < stdio. h >
void main()
{    int i = 10;
     { i++;
       printf (" % d", i ++);
     }
     printf (" % d\n", i );
}
```