

第2章 基本数据类型和表达式

2.1 C语言的基本语法单位

2.2 数据和数据类型

2.3 基本数据类型

2.4 运算符和表达式

2.1 C语言的基本语法单位

1. 基本符号

按照C99的规定，C语言的基本符号集包括：

(1) 26个大写字母

(2) 26个小写字母

(3) 10个数字字符

(4) 29个图形字符：! " # % & ' () * + , - . / : ; < = > ?
[\] ^ _ { | } ~

值得注意的是上面符号均是半角符号，非全角符号，编码时需留意输入法的当前状态

2. 关键字 **用户不能用关键字作标识符！！**

关键字是程序设计语言保留下来并被赋予特定语法含义的单词或单词缩写，用来说明某一固定含义的语法概念，程序中只能使用关键字的规定作用。（类似于自然语言中具有特定含义的动、名词）

C99 中的37个关键字，常用的有：

♥ 与数据类型有关的：

char int float double signed unsigned
short long void struct union typedef
enum sizeof

♥ 与存储类别有关的：

auto extern register static

♥ 与程序控制结构有关的：

do while for if else switch case
default goto continue break return

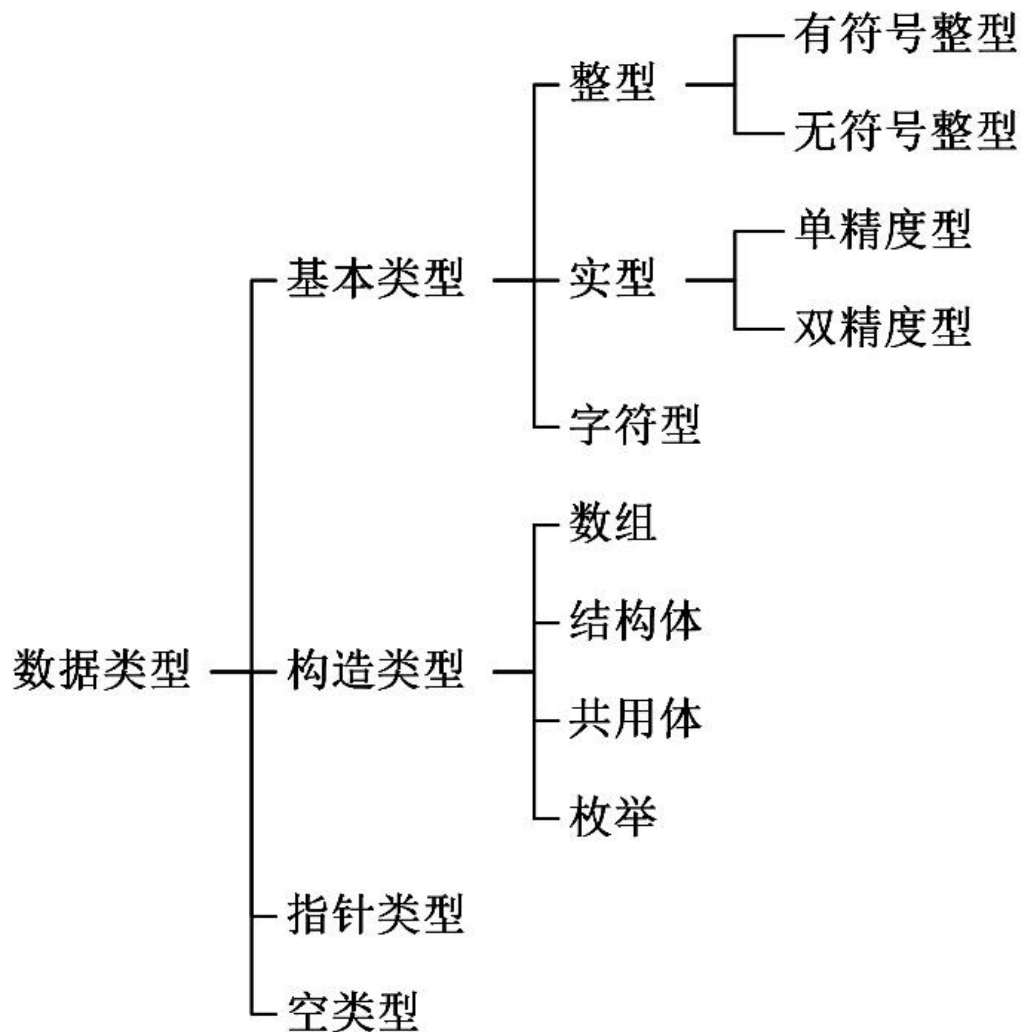
就是用来标识变量名、符号常量名、函数名、类型名、文件名等的有效字符序列。(类似于自然语言中各种事物的名字)

例如:

不合法标识符，MULTIPLY 2 * b.
8

△

2.2 C语言的基本语法单位



规定:在程序中用到的数据,必须指定数据类型。

2.3 基本数据类型

- 1. 常量与符号常量

常量是在程序运行过程中其值不能被改变的量。例如，100、0、-13为**整型常量**，3.14、-0.5、3.8e2为**实型常量**，'A'、'3'为**字符常量**。

为了增加程序的可读性，程序中经常用一个标识符来代表一个常量，即给常量命名，其语法格式如下：

#define **标识符** **常量**

例2-1： 阅读程序。程序功能是计算半径为10的圆的周长。

程序如下：

必须先定义后使用，
习惯用大写字母表示。

```
• #include <stdio.h>
• #define PI 3.14 //宏定义，符号常量PI代表实数3.14
• int main()
• {
•     double r, perimeter;
•     r=10.0;
•     perimeter=2*PI*r;
•     printf("perimeter=%f\n",perimeter);
•     return 0;
• }
```

常用这种方式简化程序调试，但是必须在程序的开头定义要使用的符号常量。称为宏定义。

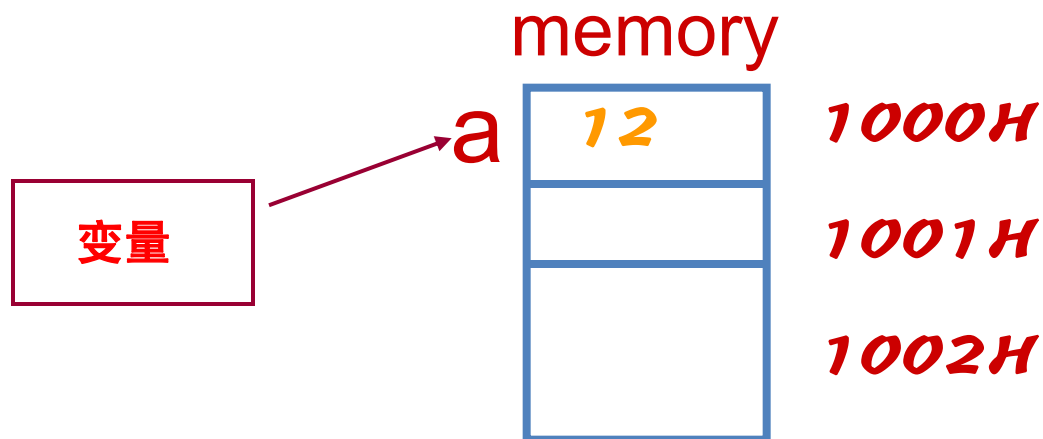
• 2. 变量

变量是**程序运行过程中其值可以改变的量**。C语言规定，程序中使用变量时，必须“**先定义，再赋值，后使用**”。

变量定义是指在程序中确定**变量名**和**数据类型**。程序编译时根据数据类型为变量分配不同大小的**存储单元**。**编写程序时通过变量名来存、取存储单元**。

变量定义语句一般位于函数体的声明部分，其语法形式如下：

数据类型关键字 变量名1，变量名2，… …，变量名n；



例如：

```
int num, count, i;    //定义三个int类型变量
float radius, area;   //定义两个float类型变量
char ch;              //定义一个char类型变量
```

为了给变量设定一个初始值，可以采用如下形式：

```
int num=100, count=0, i=0;
float radius=0, area=0;
char ch= 'A';
```

不可以用下面的写法对几个变量同时赋同一个初值

```
int i=j=k=0;
```

但下面的做法是允许的：

```
int i,j,k;
i=j=k=0;
```

ANSI C标准没有规定各类数据所占内存的字节数，只要求long型数据不短于int型，short型不长于int型。

根据整型数据所占内存空间大小分为：short、int、long

根据二进制数最高位的意义分为：signed、unsigned

不同的整数类型表示数值的范围不同，如下表所示：

类型名称		类型关键字		字节数	位数	数值范围
		完整写法	省略写法			
有符号	短整型	short int	short	2	16	$-2^{15} \sim 2^{15}-1$
	基本整型	int		2	16	$-2^{15} \sim 2^{15}-1$
				4	32	$-2^{31} \sim 2^{31}-1$
	长整型	long int	long	4	32	$-2^{31} \sim 2^{31}-1$
无符号	短整型	unsigned short int	unsigned short	2	16	$0 \sim 2^{16}-1$
	基本整型	unsigned int	unsigned	2	16	$0 \sim 2^{16}-1$
				4	32	$0 \sim 2^{32}-1$
	长整型	unsigned long int	unsigned long	4	32	$0 \sim 2^{32}-1$

如果程序在计算过程中，数值超出类型表示的数值范围，会产生溢出错误，得到错误结果；另一方面，一味用占空间大的整型变量，变量数多时，空间浪费问题需要重视。

整型常量

- 在C语言源程序中，可以用十进制、八进制、十六进制这三种制式的数来表示各种类型的整型字面值常量
- 形式：

符号位	制式前缀	该进制数字(字母)序列	后缀
-----	------	-------------	----
- 例：十进制整数：1234567891、-1234567891、93U、123Lu、9072U
- 八进制：正号+可以，负号-必须
- 十六进制：正号+可以，负号-必须
- N进制数转为十进制数：十六进制数字加字母：0~9、A~F(或a~f)
- 十进制数转为N进制数的方法：将十进制数作为下一次被除数，再除以N取余，直到被除数为0，所求余数按相反顺序输出
- 例：072=7*8+2=58

$$\begin{array}{r} 8 \overline{) 58} \\ \underline{8 \times 7} \\ 0 \end{array}$$

2
7 ↑

所以：58=072

整型变量

- 变量定义实质上是为数据创建变量空间，需要指明类型和空间的名称即变量名。C语言通过定义语句来完成这项工作。
- 变量名代表内存中的存储单元，变量的类型决定存储单元的大小。

例：//定义三个int类型的整型变量，并初始化c为10

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

//定义一个long类型的整型变量

```
long len;
```

//定义一个无符号的short类型的整型变量

```
unsigned short x;
```

整型数据的存储格式

- 所有数据在计算机中均以二进制数的形式存在于内存中
- 整数的二进制数有3种编码方式：**原码、反码、补码**，为了将减法统一到加法运算中，C语言中有符号整数采用了补码。
- 正数：原码、反码、补码形式统一，最高位为0表示是正数，余下的二进制码是其数值二进制表示
- 例：short 型变量 x 的值为 -1，其补码为：11000001，则其原码、反码、补码分别为：
0的原码有两种形式：
0的反码有两种形式：
00000000 00000000 0001
0的补码只有一种形式：
00000000 00000000
- 负数的原码：最高位为1，其余位是其相反数相同
- 负数的反码：最高位为1，其余位是其原码各位取反
- 负数的补码：最高位为1，其余位是其反码最低位加1

-97的补码

11111111 10011111

整型数据的存储格式

- 同一个二进制数码序列，当表示的整数是有符号数时，最高位代表正负；当表示的整数是无符号数时，默认为正数，无符号位，最高位也是数值位。
- 例：10000000 00000000
 - 当表示无符号数时，此位代表数值本身，即 $2^{15}=32768$
 - 当表示有符号数时，此位代表负数。其绝对值为后面各位取反再加1，得到：10000000 00000000，即 $2^{15}=32768$ ，因此该数的值为：-32768

例2-2：阅读程序

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int n1=4294967295;
```

```
    unsigned n2=4294967295;
```

```
    int m=2147483647;
```

```
    printf("作为有符号int类型输出: n1=%d,n2=%d\n", n1,n2);
```

```
    printf("作为无符号int类型输出: n1=%u,n2=%u\n", n1,n2);
```

```
    printf("m =%d\n",m);
```

```
    m=m+1;
```

```
    printf("m+1=%d\n",m);
```

```
    return 0;
```

```
}
```

作为有符号int类型输出:

n1=-1, n2=-1

作为无符号int类型输出:

n1=4294967295,

n2=4294967295

m =2147483647

m+1=-2147483648

2.实型数据。也称浮点型，是C语言中常用的基本类型。
根据实型数据精度以及所占字节数的不同，划分为单精度浮点型、双精度浮点型和长精度浮点型。
C99规定的实型数据类型，如下表所示：

类型名称	类型关键字	字节	有效数字	数值范围（绝对值）
单精度	float	4	6~7	$0, 1.2 \times 10^{-38} \sim 3.4 \times 10^{38}$
双精度	double	8	15~16	$0, 2.3 \times 10^{-308} \sim 1.7 \times 10^{308}$
长双精度	long double	16	17~18	$0, 3.4 \times 10^{-4932} \sim 3.4 \times 10^{4932}$

实型变量

- 实数类型的关键字为：float、double、long double，定义字符变量的格式：

实数类型关键字 实型变量名1 [, 字符变量名2, ...];

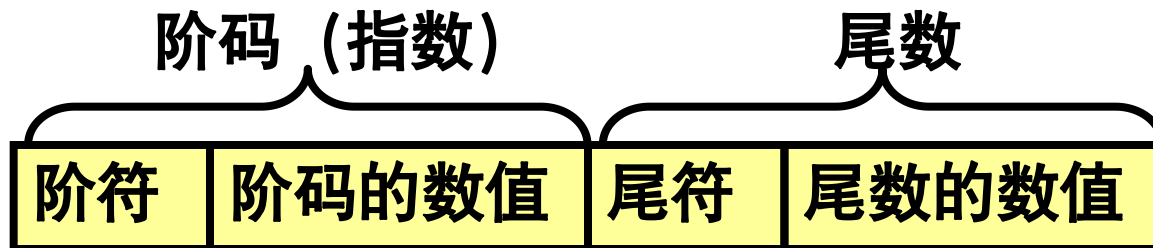
- 例如：

float x, y; //定义两个float类型的实型变量

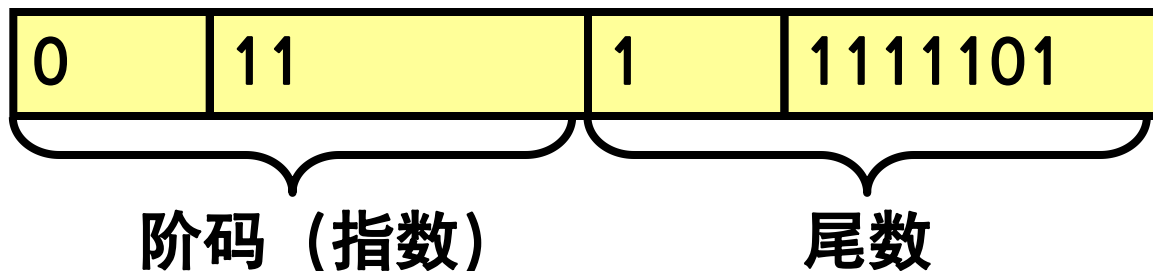
double area; //定义一个double类型的实型变量

实型数据的存储格式

- 实型数据二进制数码序列包含有两个信息——实数的尾数、指数，也就是说，尾数和指数都需要占用一定的存储空间。不同的实型所占存储空间大小不同，分配给这两部分的比例也不一样，于是有了精度与范围上的差别。（思考：整型数据没有精度问题）
- 实型数在内存的存储格式示意（忽略每一部分具体二进制位数）



- 例：-111.1101B 在内存中的存储形式，先转化成：
- $-0.1111101B \times 2^{11B}$ (保证尾数的小数点后第一位非0)



附：IEEE 754标准（美国电气电子工程师协会IEEE，1985）

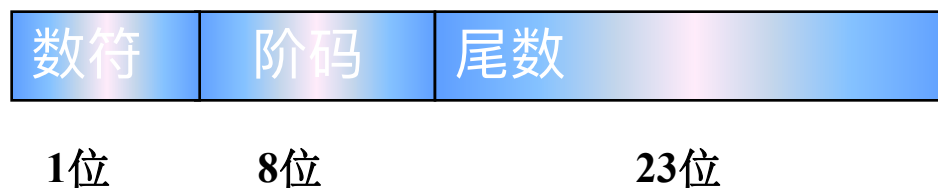
将实数分成两类：

浮点数（Float）和双精度数（Double）。

浮点数用32位表示：数符1位，阶码8位，尾数23位

双精度数用64位表示：数符1位，阶码11位，尾数52位
为了处理负指数的情况，实际数据的指数存储时数值加上127 (2^7-1) 后进行存储。尾数只存储小数部分。

单精度浮点数机内存储格式（占4个字节，32位）：



单精度实数的**精度**取决于小数部分的23位二进制数位所能表达的数值位数，将其转换为十进制，最多可表示7位十进制数字，所以单精度实数的有效位是7位。

例2-3: 阅读程序

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    float f = 12345.67899;
```

```
    double d = 12345.67899;
```

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

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

```
    return 0;
```

```
}
```

f=12345.67871

1

d=12345.6789

90

3.字符型数据

- C语言中采用ASCII字符集中的字符表示字符型字面值常量，每个字符对应一个唯一的整数编码，占**1字节**。
- 字符数据在程序中是采用编码形式进行存储和表示的，目前大多数系统采用的是ASCII码，其基本字符集包括了127个字符。详细的字符及其编码对照见附录A。

字符常量

这种表示通常用于

这种表示通常用于**控制字符**。

控制字符不可直接显示，而

可以用来表示任何字符，实际上是用该字符**ASCII码**的八进制和十六进制数来表示

- 在C语言源程序中，字符常量用单引号将字符值括起，有两种表示方法：

(1) 用单引号括起的一个字符

如：'A'、'9'、'%'

(2) 用单引号括起的以反斜杠开头的转义字符

如：'\n'、'\a'、'\t'

- 两种通用的转义字符表示：

(1) '\ddd'：1到3位八进制数所代表的字符

(2) '\xhh'：1到2位十六进制数所代表的字符

例如字符A常量就有3种等效的表示：'A'、'\101'和'\x41'

例如换行符有3种等效的表示：'\n'、'\12'和'\xA'

常用的转义字符

表示形式	含义说明	ASCII码
<code>\0</code>	空字符	0
<code>\a</code>	响铃	7
<code>\b</code>	退格，当前输出位置回到前一位	8
<code>\t</code>	水平制表，当前输出位置跳到下一个制表位	9
<code>\n</code>	换行，当前输出位置跳到下一行的开始位置	10
<code>\v</code>	垂直制表，当前输出位置跳到下一个Home位置	11
<code>\f</code>	换页，当前输出位置跳到下一页的开始位置	12
<code>\r</code>	回车，当前输出位置跳到当前行的开始位置	13
<code>\"</code>	双引号	34
<code>\'</code>	单引号	39
<code>\\</code>	反斜线 (\) 字符	92

例2-4： 阅读程序

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    printf("Welcome to \"C Language\"!\n");
```

```
    printf("It\tisn\t an easy job.\b\x20");
```

```
    printf("to learn C.\n");
```

```
    return 0;
```

```
}
```

Welcome to "C Language"!
It isn't an easy job to learn
C.

字符变量

- 字符类型的关键字为：char，定义字符变量的格式：
char 字符变量名1 [, 字符变量名2, ...];

例如：

```
char c1, c2;      //定义两个char类型变量c1和c2
```

```
char digit='9';
```

```
    //定义char类型变量digit，并初始化值为字符'9'
```

字符型数据的存储格式

- 128个标准字符和数值0~127一一对应，这种对应使字符型数据在内存中以1字节的整数二进制形式存储
- 因此在C语言中，字符型数据可以当作整型数据使用，而0~127之间的整数也可以当作字符数据使用
- 需要熟记的几个字符的ASCII码：
 - 'A' -- 65,其他的大写字母字符依序增加 例. 'E' --69
 - 'a' -- 97,其他的小写字母字符依序增加 例: 'f' --102
 - '0' -- 48,其他的数字字符依序增加 例: 'B'+32='b'
 - ' ' -- 32,空格字符的ASCII码 例: '8'-48=8或'8'-'0'=8;
8+48='8' 或 8+'0'='8'
- 熟练掌握以下两组转换：
 - 对应大小写字母字符的转换：大写字母-32=小写字母 或 小写字母+32=大写字母
 - 对应数字字符与整数数字的转换：数字字符-48=整数数字 或 整数数字+48=数字字符

例2-5: 阅读程序

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    char c='A';
```

```
    int n=65;
```

```
    printf("c=%c, n=%c\n",c,n);
```

```
    printf("c=%d, n=%d\n",c,n);
```

```
    c=c+32;    //计算c对应的小写字母的ASCII码
```

```
    printf("c as char=%c,c as int=%d\n",c,c);
```

```
    return 0;
```

```
}
```

c=A, n=A

c=65, n=65

c as char=a, c as

int=97

例： ASCII码字符‘a’、’A’、’1’ 存储为： 97、 65、 49

‘a’的ASCII值为97

内存中存储形式



0 1 1 0 0 0 0 1

整型数97

内存中存储形式

0 0 0 0 0 0 0 0

0 1 1 0 0 0 0 1

在ASCII范围以内,整型数据与字符型数据可以通用，整型变量和字符型变量可以相互赋值，字符型数据可以直接与整型数据进行算术运算。

计算字符'A'与整型数据25的和

```
main()  
{  
    char a;  
    int b;  
    a=' A' ;  
    b=a+25;  
    printf( "%c,%d,%c,%d" ,a,a,b,b);  
}
```

程序运行结果： A, 65, Z, 90

字符串常量

字符串常量是用双引号括起来的一个或多个字符。

例: "a"

串长 1

"This is C string"

串长 16

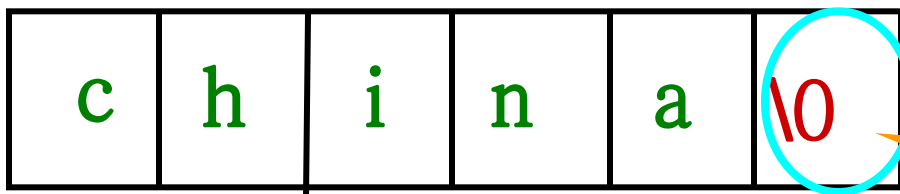
" " (空格)

串长 1

" " (不含空格)

串长 0

字符串常量中的字符依次存储在内存中的一块连续区域，末尾自动添加\0作为字符串的结束标志。 n 个字符组成的字符串常量，占内存空间为 $n+1$ 个字节。

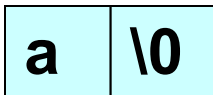


字符串结束标记
“空”字符

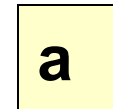
字符常量和字符串常量的区别:

在程序中，可以用字符常量或字符串常量表示单个字符，例如 'a'，或 "a"，两者的区别如下：

- (1) 字符串"a"在内存中占2个字节，而字符'a' 在内存中占1个字节。



字符串"a"



字符'a'

- (2) 不能将字符串赋给一个字符变量。

例： `char ch1,ch2;`

`ch1='a';` `/*正确*/`

`ch2="a";` `/*错误*/`

2.4 运算符和表达式

C语言中的运算符非常丰富，主要包括：

算术运算符：+、-、*、/、%、++、--

关系运算符：>、>=、<、<=、==、!=

逻辑运算符：!、&&、||

位运算符：<<、>>、~、^、&、|

赋值运算符：=、复合赋值运算符

条件运算符：?:

逗号运算符：,

指针运算符：&、*

求字节数运算符：sizeof

强制类型转换运算符：(类型关键字)

分量运算符：->、.

数组下标运算符：[]

其他运算符：括号()、函数调用等

学习运算符不需要死记硬背，伴随各章节的学习逐个掌握即可

使用运算符时，需要考虑运算符的以下特征：

(1) 运算符连接的运算对象的个数

如果一个运算符只能对一个运算对象进行运算，则称为单目运算符。依次类推，可以连接两个运算对象的称为双目运算符，连接三个运算对象的称为三目运算符。

(2) 运算符的优先级

当一个表达式中同时出现多个运算符时，优先级决定了运算的先后次序。例如大家都熟知的算术运算“先乘除，后加减”的优先级规定。C语言对其种类繁多的运算符共规定了15个级别的优先级。

(3) 运算符的结合性

当多个具有相同优先级的运算符连续出现在表达式中时，运算符的结合性规定了运算的先后次序。结合性分为“左结合”和“右结合”两种，左结合是指优先级相同的运算符按从左到右的顺序运算，右结合是从右到左运算。

1.基本算术运算符。

C语言提供5个基本算术运算符和两个单目的正、负运算符

运算符	运算说明	结合性	优先级	表达式示例
+	正号，结果是运算数本身	右结合	14	+a、+12.5
-	负号，结果是运算数的相反数		14	-a、-3
*	乘	左结合	13	a*2
/	除		13	a/2
%	模，结果是两个运算数相除的余数		13	a%2
+	加		12	a+2
-	减		12	a-2

注意： / 是除法运算符。当两个整数相除时为整除。
9/2=4； -9/2=-4；（若有一个是负数,采取**向零取整**）

%是取余数运算符,只能作用于两个整数。运算结果的符号与被除数的符号一致。

9%2=1； -9%2=-1； 9%-2=1

2. 自增运算符和自减运算符。

自增和自减运算符是C语言中比较有特色的单目运算符，它们的运算对象只能是一个数值类型的变量。其作用是使变量自身的值增加1或减少1，结合性为**右结合**（ $-a++$ 等价于 $-(a++)$ ）。两种运算符语法形式分别有两种：

$++$ 变量 或 变量 $++$

$--$ 变量 或 变量 $--$

运算符位于**变量（不能用于常量或表达式）**前面时，先将变量的值加1（或减1），变量的新值作为表达式的运算结果。运算符位于变量后面时，变量的原值作为表达式的运算结果，然后再将变量的值加1（或减1）。

例2-6：阅读程序

```
#include <stdio.h>
```

```
int main()
```


```
{
```

```
    int i=10,j=10;
```

```
    printf("%d,%d\n", i++, ++j);
```

```
    return 0;
```

```
}
```



10,1
1

例2-6中，第5行输出语句中，表达式“i++”的结果是变量i自增1之前的值，而表达式“++j”的值是变量j自增1之后的值，因此会产生相应的输出结果。

例: 将下列数学表达式写成符合C语言规则的表达式.

$$\frac{a + b + c}{\sqrt{a} + b(\sin x + \sin y + \sin z)}$$

$(a+b+c)/(\text{sqrt}(a)+b*(\sin(x)+\sin(y)+\sin(z)))$

表达式必须书写在一行，其中sqrt(a)和sin(x)、sin(y)、sin(z)都是数学函数的引用，表达式中用了三层括号，以保证表达式的运算顺序。

强调: 对C语言表达式的理解和掌握，除了要严格遵循表达式构成的规则，还要加强对表达式含义的理解，掌握运算符的优先级和结合规则。在此基础上才能灵活地运用表达式，有效地对实际问题进行描述。

常用的数学库函数：

平方根函数： `sqrt(x)`, 计算`sqrt(4.0)`的值为2.0

绝对值函数： `fabs(x)`

幂函数： `pow(x.n)`, 计算 x^n

指数函数： `exp(x)`, 计算 e^x

以e为底的对数函数`log(x)`, 计算 $\ln x$

**调用数学函数时，要求在源文件中包含头文件`math.h`。
参见附录C.1**

3.赋值运算符和赋值表达式

C语言采用“=”作为赋值运算符，其赋值表达式的语法形式如下：
变量=表达式

进行赋值运算时，有时会出现“=”右边的表达式的结果的数据类型与左边的变量数据类型不一致的情况，此时以左边变量的数据类型为准，把右边表达式结果转换为左边变量的类型。例如：

```
int a; double b;
```

```
a=9.0/2;
```

//右边表达式结果是double类型的4.5，赋值时转换为int类型，a为4

```
b=10;
```

//右边是int型常量10，赋值时转换为double类型，d得到10.0

复合赋值运算符和表达式

为了使程序源代码书写简洁和进行代码效率优化，C语言允许将5种双目算术运算符和5种双目位运算符放在“=”左边构成10种复合赋值运算符

复合赋值运算	作用说明	复合赋值运算	作用说明
$x+=y$	$x=x+y$	$x<<=y$	$x=x<<y$
$x-=y$	$x=x-y$	$x>>=y$	$x=x>>y$
$x*=y$	$x=x*y$	$x\&=y$	$x=x\&y$
$x/=y$	$x=x/y$	$x^{\wedge}=y$	$x=x^{\wedge}y$
$x\%=y$	$x=x\%y$	$x =y$	$x=x y$

例：int a=1, b=2, c=2; double x=1.5,y=2,z=2;

c*=a-b;

2*(1-2)=-2

z-=x+y;

2.0-(1.5+2.0)=-1.5

c的值?

int型

z的值?

double型

4.逗号运算符(左结合，是所有运算符中级别最低的运算符)

逗号运算符“,”可以将两个表达式连接起来，实际程序经常使用多个逗号将多个表达式顺序连接起来，其基本语法形式如下：

表达式1, 表达式2, 表达式3,, 表达式n

例如：

//从左到右依次给三个变量赋值，逗号表达式的结果是3

x=1, y=2, z=3

//表达式运算后，表达式的值是2，变量x的值是8

x=5+3, 5-3

//该表达式是赋值表达式，将逗号表达式的结果2赋值给变量x

x=(5+3, 5-3)

例如：a=3*5,a*4

a=3*5,a*4,a+5

x=(a=4%3, a+1, a*10)

a等于15，表达式的值60

a等于15，表达式的值20

a等于1，表达式的值10

5. sizeof运算符

sizeof运算符是C语言特有的一种运算符，其语法形式如下：

sizeof(运算对象)

该运算符的作用是获得运算对象占用内存空间的字节数，结果是整数类型。其中，运算对象可以是数据类型关键字、常量、变量和表达式。

例2-7：阅读程序

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    float price;
```

```
    printf("int类型占用%d个字节.\n", sizeof(int));
```

```
    printf("变量price占用%d个字节.\n", sizeof(price));
```

```
    printf("实数3.14占用%d个字节.\n", sizeof(3.14));
```

```
    printf("字符串\"AB\"占用%d个字节.\n", sizeof("AB"));
```

```
    printf("表达式9+5的结果占用%d个字节.\n", sizeof(9+5));
```

```
    return 0;
```

```
}
```

int类型占用4个字节.

变量price占用4个字节.

实数3.14占用8个字节.

字符串"AB"占用3个字节.

表达式9+5的结果占用4个字节.

6.混合运算与数据类型转换

整型、**实型**和**字符型**可以混合运算

混合运算与数据类型转换C语言程序中允许表达式中存在不同数据类型数据之间的运算，由于不同的数据类型的存储方式不同，因此表达式在进行混合数据运算时要先进行数据类型的转换，把不同的数据类型转换成相同的数据类型后，再进行运算。C语言提供“**自动类型转换**”和“**强制类型转换**”两种类型转换方式。

自动类型转换

转换规则如下:

- (1) 如果其中一个操作数是long double类型, 则把另一个也转换为long double类型;
- (2) 否则, 如果其中一个操作数是double类型, 则把另一个也转换为double类型;
- (3) 否则, 如果其中一个操作数是float类型, 则把另一个也转换为float类型;
- (4) 否则, 如果其中一个操作数是unsigned long类型, 则把另一个也转换为unsigned long类型;
- (5) 否则, 如果其中一个操作数是long类型, 则把另一个也转换为long类型;
- (6) 否则, 如果其中一个操作数是unsigned int类型, 则把另一个也转换为unsigned int类型;
- (7) 否则, 两个操作数都转换为int类型。

例1:

已知: `int i; long e; float f; double d;`

`f = 10 + 'a' + i * f - d / e`

f的类型?

int double double

double

double

注意: 运算过程中的类型转换**不是**变量本身数据类型的转换

例2：已知三角形的底为15，高为20，计算其面积。

```
include <stdio.h>
main()
{
    int c=15,h=20,s;
    s=1/2*15*20;
    printf("s=%d",s);
}
```

运行结果：

强制类型转换

C语言程序中，如果有需要可以对数据进行强制类型转换，也称为显式类型转换。强制类型转换的语法形式如下：

(数据类型关键字)表达式

例如：定义double a=1.5, b=2.7; int k=1;

(1) 表达式(double)k/5运算时，先把k的值转换为double，因此整个表达式的结果是double类型，其值为0.2。

(2) 表达式(int)a+b运算时，把a的值强制转换为int类型的1，然后1又自动转换为double类型的1.0，整个表达式的结果是double类型，其值为3.7。

(3) 表达式(int)(a+b)运算时，先计算2个数的和是double类型的4.2，再把和强制转换为int类型，整个表达式的结果是int类型，其值为4。

7.位运算（选讲）

位运算是指对二进制的位（bit）进行的运算。C语言是一种适合开发系统软件的语言，而系统软件开发中常常需要处理二进制位的问题。

运算符	名称	功能说明
~	取反	单目，操作数每个二进制位都取反，即0变1，1变0
&	按位与	双目，两个操作数对应二进制位都是1时，结果为1
	按位或	双目，两个操作数对应二进制位有一个是1时，结果为1
^	按位异或	双目，两个操作数对应二进制位相同结果为0，否则为1
<<	左移	双目，操作数的二进制位全部左移n位
>>	右移	双目，操作数的二进制位全部右移n位

位运算

位运算是针对二进制数的运算，通常只适用于整数数据。

C语言中提供的位运算符有：

~、<<、>>、&、^、|

1.按位取反运算符 ~

形式：~A

功能：把A的各位都取反，（即0变1，1变0）

例如：int A=179

A	0	0	0	0	0	0	0	0	1	0	1	1	0	0	1	1
~A	1	1	1	1	1	1	1	1	0	1	0	0	1	1	0	0

2. 按位与运算符 &

形式: **A&B**

功能: 对A的各位与B的对应位进行比较, 如果两者都为1,
A&B对应位上的值为1, 否则为0。

例如: int A=179 (二进制 10110011)

int B=169 (二进制 10101001)

A	1	0	1	1	0	0	1	1
B	1	0	1	0	1	0	0	1
A&B	1	0	1	0	0	0	0	1

3. 按位或运算符|

形式： $A \mid B$

功能：对A的各位与B的对应位进行比较，如果两者中有一个为1， $A \mid B$ 对应位上的值为1，否则为0。

例如：int A=179 (二进制10110011)

int B=169 (二进制10101001)

A	1	0	1	1	0	0	1	1
B	1	0	1	0	1	0	0	1
$A \mid B$	1	0	1	1	1	0	1	1

4. 按位异或运算符 ^

形式: $A \wedge B$

功能: 对A的各位与B的对应位进行比较, 如果两者不同,
 $A \wedge B$ 对应位上的值为1, 否则为0。

例如: `int A=179` (二进制 10110011)

`int B=169` (二进制 10101001)

A	1	0	1	1	0	0	1	1
B	1	0	1	0	1	0	0	1
A&B	0	0	0	1	1	0	1	0

5. 左移运算符 <<

形式: $A \ll n$ (其中 n 为一个大于0的整型表达式)

功能: 把A的值向左移动 n 位, 右边空出的 n 位用0填补。

当左移时移走的高位中全都是0时, 相当于对A作 n 次乘以2的运算。

例如: $\text{int } A=27$ (二进制00011011)

A	0	0	0	1	1	0	1	1
$A \ll 3$	1	1	0	1	1	0	0	0

6.右移运算符>>

形式： $A \gg n$ （其中 n 为一个大于0的整型表达式）

功能：把A的值向右移动n位，左边空出的n位用0填补。

相当于对A作n 次除以2的运算。

例如： $\text{int } A=179$ （二进制10110011）

A	1	0	1	1	0	0	1	1
$A \gg 3$	0	0	0	1	0	1	1	0

课后习题

- 教材课后习题1、2
- 预习下一章

本章結束