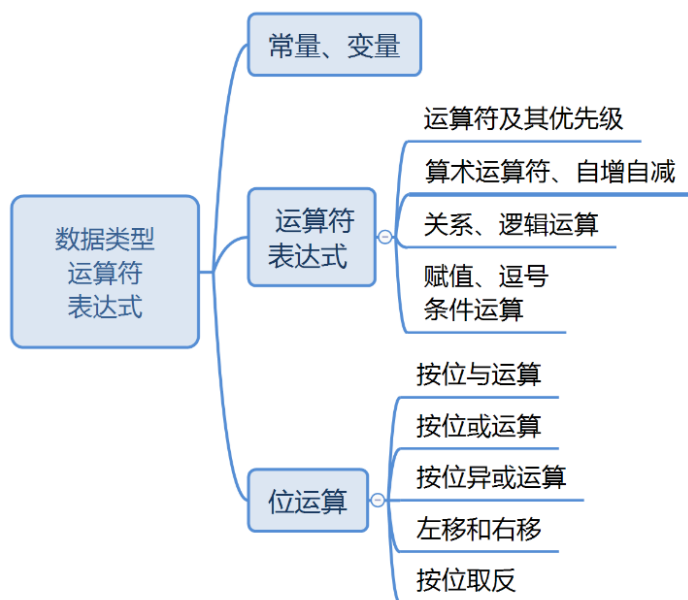




专题二 数据类型、运算符和表达式

【内容预览】



2.1、常量和变量

【知识清单】

2.1.1、常量

1. 整型常量

- (1) 十进制整数，即按日常接触的数字形式正常表达。如 123、-120。
- (2) 八进制整数，以 0 开头的数字进行表达。如 0123 表示八进制数 123，即 $(123)_8$ ，其值为 $1 \times 8^2 + 2 \times 8^1 + 3 \times 8^0$ ，等于十进制数 83。-011 表示八进制数-11，即十进制数-9。
- (3) 十六进制整数，以 0x 开头的数字进行表达。如 0x123，代表十六进制数 123，即 $(123)_{16}$ ，其值为 $1 \times 16^2 + 2 \times 16^1 + 3 \times 16^0$ ，等于十进制数 291。-0x12 等于十进制数-18。
- (4) 整型常数后缀：u 或 U(unsigned)、l 或 L(long)、u/U 与 l/L 的组合(如：ul、lu、Lu 等)。例：100u; 123u; 0x123l;

2. 浮点型常量

- (1) 十进制小数形式：例如 99.9。
- (2) 指数形式的一般形式为:

[±][整数部分][.][小数部分][e/E±n][后缀]

(3) 浮点常数后缀：**f 或 F**（单精度浮点数）、**l 或 L**（长双精度浮点数）。(注：**因浮点型常数总是有符号的，故没有 u 或 U 后缀**)。例：1.23e5f; 1.23l; -123.45f;

3. 字符型常量

- (1) 一种是**普通字符**，即用单引号括起来的一个字符，如'9'，'y'，'?'。字符常量在储存在计算机的储存单元中时，是以其代码（一般采用 ASCII 代码）储存的。如： 字符'9'的 ASCII 码为 57，即存储时，计算机存储的是 57。
- (2) 另一种是**转义字符**，即特殊字符常量。转义字符是 C 语言中表示字符的一种特殊形式，其含义是将反斜杠后面的字符转换成另外的意义。

表 2-1 转义字符列表

转义字符	含义	ASCII 码（16/10 进制）
\0	空字符(NULL)	00H/0
\n	换行符(LF)	0AH/10
\r	回车符(CR)	0DH/13
\t	水平制表符(HT)横向跳转到下一制表位置	09H/9
\a	响铃(BEL)	07/7
\b	退格符(BS)	08H/8
\f	换页符(FF)	0CH/12
\'	单引号 (')	27H/39
\"	双引号 (")	22H/34
\\	反斜杠 (\)	5CH/92
\ddd	任意字符	三位八进制
\xhh	任意字符	二位十六进制

注意：ddd 和 hh 分别为八进制和十六进制的 ASCII 码。如'\101'字符表示'A'，'\x41'也表示字母'A'。

4. 字符串常量与字符常量的区别

- (1) 字符常量由**单引号**括起来，字符串常量由**双引号**括起来。如：字符'9'，字符串"9"（注意与数字 9 的区别）
- (2) 字符常量只能是单个字符，字符串常量则可以有一个或者多个字符。
- 例如：字符 'a', 'b' 字符串"ab"
- (3) **不能把字符串常量赋予给字符变量。**
- 例如：char a="abcd";是错误的

(4) 字符常量占用一个字节的内存空间，但是字符串所占用的字节数等于其中的字符数加一，增加的一个字节存放的是'\0'，'\0'是字符串结束的标志，需要占用一个字节的内存。例如："ab"占用 3 个字节内存。

2.1.2、变量

1. 整型变量

在 C 语言中，整型用 int 来表示，同时可以用 long, short 来修饰 int，称为长整型和短整型。又可以用 signed 和 unsigned 来修饰 int，称为带符号整型，和无符号整型。

2. 浮点型变量

C 语言中比较常用的两种浮点数的类型是

float 单精度型（占用 4 个字节）

double 双精度型（占用 8 个字节）

注意：浮点数均为有符号浮点数，因此不能用 signed 或者 unsigned 来修饰。

3. 字符变量

字符变量类型说明符是 char。字符值是以 ASCII 码值的形式存放在内存单元之中的。因此，可以把字符变量看成是整型量。C 语言允许对整型变量赋以字符值，也允许对字符变量赋以整型值。同理，输出时两者也可以互相交换。

例如：#include<stdio.h>

```
void main()
{
    char ch='a';
    int i=ch;
    printf("%c ASCII is %d\n",ch,ch);
    printf("%c ASCII is %d\n",i,i);
}
```

输出结果为:

```
a ASCII is 97
a ASCII is 97
```

表 2-2 字符类型

类型名	说明	字节	值的范围
short [int] /signed short [int]	有符号短整型	2 byte	-32768~32767
unsigned short [int]	无符号短整型	2 byte	0~65535
int /signed [int]	有符号整型	4 byte	-2147483648~2147483647
unsigned [int]	无符号整型	4 byte	0~4294967295
long [int]/signed long [int]	有符号长整型	4 byte	-2147483648~2147483647

unsigned long [int]	无符号长整型	4 byte	0~4294967295
char/signed char	有符号字符型	1 byte	-128~127
unsigned char	无符号字符型	1 byte	0~255
float	单精度浮点型	4 byte	$-3.4 \times 10^{-38} \sim 3.4 \times 10^{38}$ 有效数位 7 位
double	双精度浮点型	8 byte	$-1.7 \times 10^{-308} \sim 1.7 \times 10^{308}$ 有效数位 15 位

注：[]中的关键字在实际编程中可以省略。

【解题技巧】

例 2.1.1 #include<stdio.h>

```
void main()
{
    char ch;
    ch='\44';
    printf("ch is %c\n",ch);
}
```

输出结果是：

正解：ch is \$

分析：ch='\44'是把 ASCII 码值为'\44 '即 36 的字符赋给 ch，对应的字符为\$。所以输出的结果是 ch is \$。

2.2、运算符和表达式

【知识清单】

2.2.1、运算符及其优先级

表 2-3 运算符及其优先级

优先级	运算符	名称或含义	使用形式	结合方向	说明
1	[]	数组下标	数组名[常量表达式]	左到右	
	()	圆括号	(表达式)/函数名(形参表)		
	.	成员选择（对象）	对象.成员名		
	->	成员选择（指针）	对象指针->成员名		
	++	后置自增运算符	++变量名		单目运算符
	--	后置自减运算符	--变量名		单目运算符
2	-	负号运算符	-表达式	右到左	单目运算符
	(类型)	强制类型转换	(数据类型)表达式		
	++	前置自增运算符	变量名++		单目运算符
	--	前置自减运算符	变量名--		单目运算符
	*	取值运算符	*指针变量		单目运算符
	&	取地址运算符	&变量名		单目运算符
	!	逻辑非运算符	!表达式		单目运算符

	~	按位取反运算符	~表达式		单目运算符
	sizeof	长度运算符	sizeof(表达式)		
3	/	除	表达式/表达式	左到右	双目运算符
	*	乘	表达式*表达式		双目运算符
	%	余数（取模）	整型表达式/整型表达式		双目运算符
4	+	加	表达式+表达式	左到右	双目运算符
	-	减	表达式-表达式		双目运算符
5	<<	左移	变量<<表达式	左到右	双目运算符
	>>	右移	变量>>表达式		双目运算符
6	>	大于	表达式>表达式	左到右	双目运算符
	>=	大于等于	表达式>=表达式		双目运算符
	<	小于	表达式<表达式		双目运算符
	<=	小于等于	表达式<=表达式		双目运算符
7	==	等于	表达式==表达式	左到右	双目运算符
	!=	不等于	表达式!= 表达式		双目运算符
8	&	按位与	表达式&表达式	左到右	双目运算符
9	^	按位异或	表达式^表达式	左到右	双目运算符
10		按位或	表达式 表达式	左到右	双目运算符
11	&&	逻辑与	表达式&&表达式	左到右	双目运算符
12		逻辑或	表达式 表达式	左到右	双目运算符
13	?:	条件运算符	表达式 1? 表达式 2: 表达式 3	右到左	三目运算符
14	=	赋值运算符	变量=表达式	右到左	
	/=	除后赋值	变量/=表达式		
	=	乘后赋值	变量=表达式		
	%=	取模后赋值	变量%=表达式		
	+=	加后赋值	变量+=表达式		
	-=	减后赋值	变量-=表达式		
	<<=	左移后赋值	变量<<=表达式		
	>>=	右移后赋值	变量>>=表达式		
	&=	按位与后赋值	变量&=表达式		
	^=	按位异或后赋值	变量^=表达式		
	=	按位或后赋值	变量 =表达式		
15	,	逗号运算符	表达式,表达式,...	左到右	从左向右顺序运算

注：单目运算符是指所需运算量为一个的运算符；双目运算符指所需运算量为两个的运算符。

2.2.2、算术运算符

- 算术运算符包括+（加）、-（减）、*（乘）、/（除）、%（余数）。
- 1. +、-、*、/运算符的运算量可以是任何整型或浮点型常量、变量、有返回值的函数及其表达式。
 - 2. **x/y 时，y 的值不能为 0。**
 - 3. %求余数运算符要求**运算量必须是整型，且%后面的量不为 0，且余数的正负取决于被除数。**
例如：4%2 结果为 0
 -15%2 结果为-1
 8%0 错误；
 - 4. 当双目运算符的两个运算量的类型相同时，它们的运算结果类型与运算量类型相同。
但是两个运算量类型不同时，**运算前会按照一般的转换规则转换为相同类型。**
例如：15.5+5 运算时将 5 转换为浮点数 5.0，结果为 20.5。
 - 5. 强制类型转换是靠强制类型转换运算符来实现数据类型转换的，一般类型为：

(类型名) 表达式

例如： float x=6.5;
 int y;
 y=(int)x;
结果： y=6;x=6.5;

2.2.3、自增和自减运算

++（自增），--（自减）都是单目运算符，都有前置和后置两种不同的形式，两种形式单独使用时并没有区别，但是代入表达式中的时候，**前置式先进行自增或自减运算，再代入表达式运算，而后置式先用原值进行运算，之后再自增或者自减运算。**（可参考基础篇第 1 题）

注意：**自增和自减运算只能用于变量，不能用于常量。**

表 2-4 前置式和后置式

++i	先执行 i+1，再使用 i	int i=0; y=++i;结果: y=1
i++	先使用 i，再执行 i+1	int i=0; y=i++;结果: y=0

2.2.4、关系运算符

关系运算符都是双目运算符，它用来比较两个运算量之间大小关系。

C 语言中的关系运算符主要包括：

<（小于）,<=（小于等于）,==（等于）,>=（大于等于）,>（大于）,!（不等于）。

2.2.5、逻辑运算符

C 语言中的逻辑运算符包括&&（逻辑与）、||（逻辑或）、！（逻辑非，单目运算符）。逻辑运算符&&、||的两个表达式类型可以不同，运算时也不需要类型转化，只要遵循非 0 视为真，0 值视为假。运算结果也只有 0 和 1 两种情况。

注意：**&&只要左边为逻辑 0，右边的表达式计算机将不会计算，同理，||左边为逻辑 1，右边不计算。**

例如: `int a=1,b=0,c;`

`c=++a||++b;`

`b=!b;`

结果: **c=1, a=2, b=1;**

2.2.6、赋值运算符

1. 基本赋值运算符“=”是一个双目运算符, 它的一般表达式为:

左值表达式=右值表达式

注意: 在进行赋值运算前, 会自动把右值表达式的值类型自动转换为左值表达式的值类型。右→左

2. 在赋值运算符“=”前加上其他运算符, 便构成了复合赋值运算符。如果用 op 来代表加在“=”之前的运算符, 则复合赋值运算符可以表示为“op=”。一般形式为:

左值表达式 op=右值表达式

例如: `int i=0; i+=1; //i+=1` 表示 `i=i+1` 结果: `i=1`。

2.2.7、条件运算符

条件运算符 (?:) 是 C 语言中唯一的三目运算符, 它的一般形式为:

表达式 1? 表达式 2: 表达式 3

它的运算过程是: 判断表达式 1 的值, 如果非 0, 执行表达式 2, 如果为 0, 则执行表达式 3。

例如: `int a,b;`

`(a>b)?a:b`

当 `a>b` 成立时, 条件表达式的值为 `a`, 否则, 条件表达式的结果为 `b`。

`(a==b)?0: ((a>b)?-1:1)`

这是一个嵌套的表达式, 先算内层表达式, 再算外层表达式的值。

2.2.8、逗号运算符

逗号运算符是双目运算符, 用它可以构成逗号表达式。一般形式为:

表达式 1, 表达式 2.....表达式 n

逗号运算的每个表达式分开进行运算, 从表达式 1 按顺序一直求到表达式 n, 最后的表达式的值就是该逗号表达式的值。

例如: `int i=1,j=0,k;`

`k= (i++,j++, i+j) ;`

最后 `k` 的结果为 3。

【解题技巧】

例 2.2.1 已知 `int i=10,j=5;` 求

(1) `++i-j--`

(2) `i*i*=j`

(3) `i=3/2*(j=3-2)`

正解：(1) 6 (2) 50 (3) 1

分析：(1) 考察的是对前置后置自增自减运算的理解，++i 结果是 11，但是 j-- 虽然单独运算时结果式 4，但是代入表达式时，j-- 用 j 的原始值 5 计算后，再自减运算，因此，答案是 6。

(2) $i*=j$ 的解果是 $i=50$ ， $i=i$ ，所以结果是 50。

(3) 因为 $(j=3-2)$ 的值就是表达式 $3-2$ 的值，原式为 $i=3/2*1$ ，“/” 整数相除，只取整数，所以 $i=1$ 。

例 2.2.2 设 $\text{int } a=12$ ，则执行完语句 $a+=a-=a*a$ 后，a 的值是_____。

A)0

B)264

C)144

D)-264

正解：D

分析：先执行 $a-=a*a$ ，计算出结果为 $a=-132$ ，再执行 $a+=a$ ，由于此时 $a=-132$ ，所以计算结果是 $a=-264$ 。

例 2.2.3 以下程序的运行的结果是

```
(1) int a=10,b=-5,c;
    c=(b>0)?a+b:a-b;
(2) int b,a=10;
    b=(a++,a%3);
```

正解：(1) 15 (2) 2

分析：(1) 由条件运算符的规则可以知道， $b<0$ ，执行 $c=a-b=15$ (2) 由若干表达式构成的逗号运算符，从左至右依次计算，取最后一个表达式值为最终的值。

例 2.2.4 `#include<stdio.h>`

```
void main()
{
    int x,y;
    float x1,y1;
    x=15;
    y=6;
    x1=15.0;
    y1=6.0;
    printf("x=%d,y=%d\n",x,y);
    printf("x+y=%d\n",x+y);
    printf("x-y=%d\n",x-y);
    printf("x*y=%d\n",x*y);
    printf("x/y=%d...%d\n",x/y,x%y);
    printf("x1/y1=%f\n",x1/y1);
}
```

写出运行结果：

正解: $x=15, y=6$
 $x+y=21$
 $x-y=9$
 $x*y=90$
 $x/y=2. \dots 3$
 $x1/y1=2.500000$

分析: 解题的关键就是理解算术运算符用法和使用的注意点。(输出函数的格式见章节 1.3)

例 2.2.5 设变量 x,y,z 均为 `double` 类型且已正确赋值, 下面不能正确表示数字式子 $x/y/z$ 的 C 语言表达式是()

- A) $x/y*z$
- B) $x* (1/ (y*z))$
- C) $x/y*1/z$
- D) $x/y/z$

正解: A
分析: 首先可以确定变量 x, y, z 都是 `double` 型, 因此不存在类型转换的问题, 所以这道题只要按照正确的数学逻辑就可以解出答案为 A。

2.3、位运算

【知识清单】

2.3.1、按位与“&”、按位或“|”、按位异或“^”

表 2-5 位运算符运算规则

符号	运算规则	例如
按位与运算符“&”	$0 \& 0 = 0, 0 \& 1 = 0,$ $1 \& 0 = 0, 1 \& 1 = 1。$ 即同为 1 的位, 结果为 1, 否则结果为 0。	3 的内部表示为 00000011 5 的内部表示为 00000101 则 3&5 的结果为 00000001
按位或运算符“ ”	$0 0 = 0, 0 1 = 1,$ $1 0 = 1, 1 1 = 1。$ 即只要有 1 个是 1 的位, 结果为 1, 否则为 0。	23 的内部表示为 00010111 35 的内部表示为 00100011 23 35 结果为 00110111
按位异或运算符“^”	$0 \wedge 0 = 0, 0 \wedge 1 = 1,$ $1 \wedge 0 = 1, 1 \wedge 1 = 0。$ 即相应位的值相同的, 结果为 0, 不相同的结果为 1。	13 内部表示为 00001101 35 内部表示为 00100011 13^35 结果为 00101110

2.3.2、二进制左移运算符“<<”右移运算符“>>”

1. 左移运算符 (<<)

二进制左移运算把数据向左移动若干位时，移出左边界的所有位都将丢失，右侧新增加的位为 0。

例如: `int a=4,a<<2` 的结果为 16。

因为变量 `a` 在内存中的二进制表示为 `00000100`，向左移动两位并在右端补 0 后的二进制值为 `00010000`，对应结果为 16。

向左移动一位等同于乘以 2，向左移动两位等同于乘以 4，因此，可以得到一个规律：在变量可以表示的范围内，向左移动 n 位等同于乘上 2 的 n 次方。

2. 右移运算符 (>>)

二进制右移若干位时，移出右边界的所有位都即将丢失，左侧的新位的补充将遵循下面的原则：

- (1) 对于无符号数，右移时左侧的新位一律补 0，称为“逻辑右移”。
- (2) 对于有符号数，若符号位是 0，则左侧新位一律补 0；若符号位是 1，则左侧一律补 1，称为“算术右移”。

例如：变量 a 是无符号数，a=8，其二进制表示为 00001000，右移两位且左侧新位补 0 结果为 00000010，所以 a>>2 的结果为 2。

变量 b 是有符号数，b=-10，其二进制表示为 11110110。因为符号位为 1，所以变量 b 右移一位且在左侧新位补 1 后的结果为 11111011，所以 b>>1 的结果为 -5。

2.3.3、按位取反运算符“~”

用于求整数的**二进制反码**，即分别将操作数**各二进制位上的 1 变为 0，0 变为 1**。

【精选习题】

答案 P110

基础篇

- 以下选项中是正确的整型常量是()
A)1234 B)'1234' C)1,234 D)"1234"
- 有以下程序

```
void main()  
{ int i=10,j=1;  
    printf("%d,%d\n",i++,++j); }
```

执行后输出结果是 ()
A) 9,1 B) 10,1 C)10,2 D)9,2
- 若有定义：int a=8,b=5,c;执行语句 c=a/b+0.4;后的值为 ()
A)2.0 B)2 C)1 D)1.4
- 已定义c为字符型变量，则下列语句中正确的是 ()
A)c=97 B)c="a" C)"97" D)'97'
- 已知大写字母 A 的 ASCII 码是 65，小写字母 a 的 ASCII 码是 97，则用八进制表示的字符常量'\101'是 ()
A)字符 e B)非法的常量 C) 字符 a D) 字符 A

6. 若变量 `c` 为 `char` 类型, 能正确判断出 `c` 为数字字符的表达式是()。
- A) `'0'<=c<='9'` B) `c>='0' || c<='9'` C) `c>=0 && c<=9` D) `c>='0' && c<='9'`
7. 假设所有变量均为整型, 则表达式 (`a=2, b=5, b++, a+b`) 的值是 ()
- A) 6 B) 2 C) 8 D) 7
8. 下面 4 个选项中, 均是不合法的转义字符的选项是 ()
- A) `'\011'`、`'\f'`、`'\'` B) `'\abc'`、`'\101'`、`'\xlf'` C) `'\1011'`、`'\'`、`'\aa'` D) `'\'`、`'\\'`、`'\xf'`
9. 设有 C 语言序列: `int x=-5; x+=x-=x*x;` 执行该语句序列后, 变量 `x` 的值是 ()
- A) 0 B) -15 C) -26 D) -60

提高篇

1. 若以下选项中的变量已正确定义, 则正确的赋值语句是 ()
- A) `x1=26.8%3;` B) `1+2=x2;` C) `x3=0x12;` D) `x4=1+2=3;`
2. 设有以下定义:
- ```
#define d 2
int a=0; double b=1.25; char c='A';
```
- 则下面语句中错误的是 ( )
- A) `a++;`      B) `b++;`      C) `c++;`      D) `d++;`
3. 以下选项中合法的实型常量是 ( )
- A) `2E0`      B) `1.3E`      C) `E-3`      D) `5E2.0`
4. 判断下列语句输出结果 ( )
- ```
printf("%d", (10>20?50:(60,70)));
```
- A) (B) 60 C) 70 D))
5. 设有: `int a=1, b=2, c=3, d=4, m=2, n=2;` 执行 (`m=a>b`) && (`n=c>d`) 后 `n` 的值为 ()
- A) 3 B) 4 C) 2 D) 1
6. 执行以下语句后 `b` 的值为 ()
- ```
int a, b, c;
a=b=c=1;
++a || ++b && ++c;
```
- A) 错误      B) 0      C) 2      D) 1
7. (重庆大学 2012) 设有如下的变量定义: `int i=8, k, a, b; unsigned long w=5; double x=1.42, y=5.2;` 则在以下的表达式中, 符合 C 语言语法的表达式是 ( )。
- A) `a+=a-(b=6)/(a=2)`      B) `x%(-5)`  
C) `a=a*9=2`      D) `y=double(i)`