



福州大学至诚学院
FUZHOU UNIVERSITY ZHICHENG COLLEGE

高级语言程序设计 (C语言与数据结构)

杨雄

83789047@qq.com





第二章 数据类型、运算符和表达式

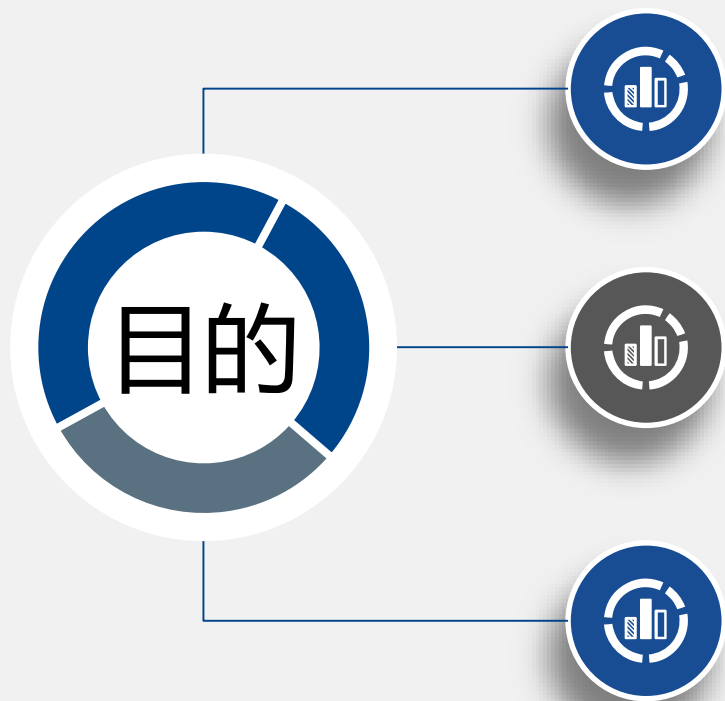
2.1 数据与数据类型

2.2 变量、常量与标准函数

2.3 基本运算符及其表达式



学习目的



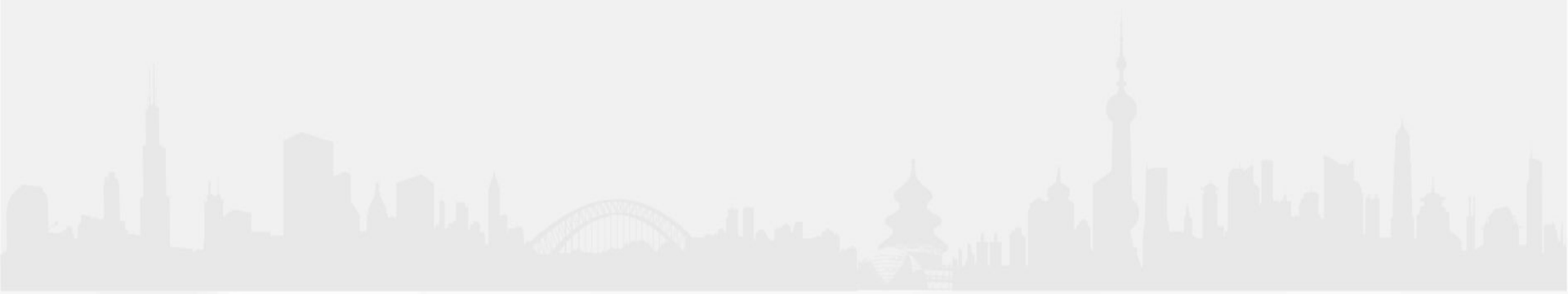
掌握常量与变量的书写和定义

掌握三种基本数据类型

掌握运算符和表达式的使用

Part.1

2.1 数据类型



2.1 数据类型

基本类型

是C语言数据类型的基本型,其值不可再分解为其他类型。

构造类型

一种由单种或多种数据类型构造而成的数据类型。

- 数组
- 结构体
- 共用体
- 枚举类型

指针类型

一种特殊的数据类型,其值为某个量的内存地址。

空类型

一种无返回值函数的数据类型。
`void`

自定义

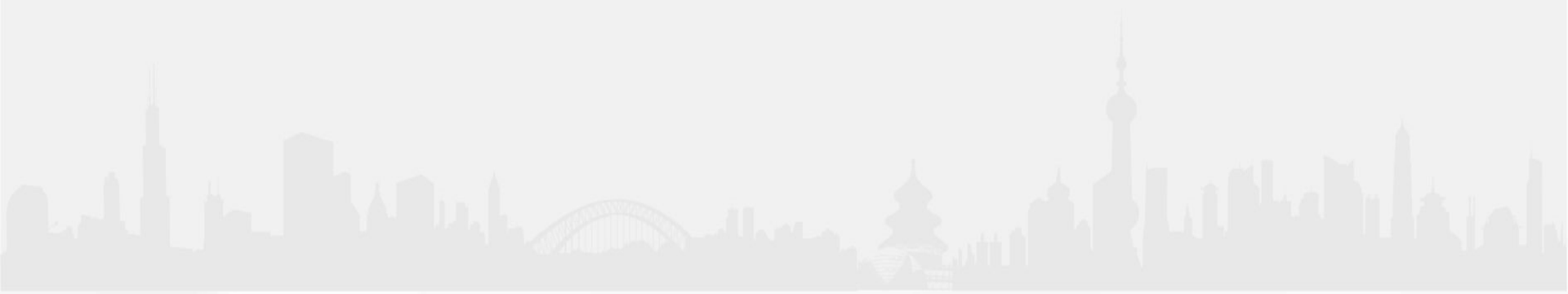
用新的类型名替代已有类型名使用。
用typedef定义

C语言的基本数据类型

- **整型:** 无符号的基本整型、短整型、长整型
- **实型:** 单精度实型、双精度实型 `float, double`
- **字符型:** 表示和存储ASCII字符。 `char`

Part.2

2.2 常量、变量和标准函数





2.2 常量、变量和标准函数

2.2.1 常量

2.2.2 变量

2.2.3 标准函数



- 常量
 - 在程序执行期间其值**保持不变**的量
- C语言有以下几种类型的常量：
 - **整型**常量
 - 实型(**浮点型**)常量
 - **字符**常量
 - **字符串**常量

➤ 整型常量

• 有三种形式:

— **十进制整数**: 如: 11, -13, 0, 65535, ... ✓

1,000 21/3 10^4 54. $10*3$ 01 ✗

— **八进制整数**: 由数字 0 开头

如: 014 -011 0177777

— **十六进制整数**: 由前缀 0x 或 0X 开头

如: 0x11 0XAFBDE ✓ 0X00FFH ✗

➤ 整型常量

— **长整型整数**：用后缀 **L/l** 表示。

012L , **65536L** , **0XCL**

— **无符号整数**：用后缀 **U/u** 表示。

017u , **0xfdbU**

— **无符号长整数**：用后缀 **UL/ul** 表示。

15uL

➤ 实型常量——浮点数

- 实型常量又称实数，指**带有小数部分的数**。C语言中的实数是以**十进制**表示的，有两种表示形式：

- **十进制小数形式**：由数字、数符和**小数点**组成。

例： 0.123, .123, 123. , 0.0 , -2.5 , ...

- **指数形式**：又称科学记数法。用E(或e)表示以10为底的指数。

$0.91 \times 10^{-3} \rightarrow$ 0.91E-03

尾数部分

指数部分

字母E/e前必须有**数字**，而E后面的阶码必须为**整数**。

例：下列**不合法**的指数形式：

①单独的小数点和单独的指数部分

如： $.E-5$ $E10$

②阶码只能是整数，不能带小数点。

如： $1234E1.5$ $2.E$ $6.5e$
 $(2*3)E-3$ $5*E4$

③ 10^{12} 不能写成 $E12$ ，必须写成： $\begin{cases} 1E12 \\ 1.0E12 \end{cases}$

➤ 字符常量

- 字符常量：是用**单引号**括起来的一个字符

如： 'a' , 'A' , '9' , '+' , '?' , ' ' **合法**

"a" , '99999' , " **非法**

- 在内存中, 字符常量以**ASCII码**存储, **一个字符占一个字节**。
- 由于字符常量是按**整数存储**的, 可以像整数一样在程序中参与相关的运算。如:

'a' - 32 ; // 执行结果 $97 - 32 = 65$

'9' - 9 ; // 执行结果 $57 - 9 = 48$

➤ 字符常量 转义字符

– 以 **'\'** 开头的字符序列，有特定的含义。

– 如: **'\\'** 表示输出一个反斜杠符

'\"' 表示输出一个双引号

\\ddd (ddd表示八进制的ASCII码)

\\xhh (hh表示十六进制的ASCII码)

– 例: **'\n'** (回车换行) → **'\12'** → **'\xa'**

'A' → **'\101'** → **'\x41'**

– 注: **'\0'** 或 **'\000'** 是代表ASCII码为0的字符,

即**空字符**(NULL), 表示整数0。

【例2.1】转义字符的应用

```
#include <stdio.h>

void main()
{
    printf( "a\tb\nc\bd\100\x40\n" );
}
```

a_____b
d@@

输出到显示屏

➤ 字符串常量

- 字符串常量：用一对**双引号**括起的字符序列。例：
“CHINA”， “a”， “\$12.5”， “ ”，
“w\x53\\np\103q”
- 字符串长度：字符串中所有**字符的个数**
- 系统**自动**在每个**字符串的末尾**加上一个空字符NULL，即‘\0’
作为字符串的结束。‘\0’是一个ASCII码为0的字符。

例：“CHINA” 在内存中所占的字节为：

C	H	I	N	A	\0
---	---	---	---	---	----

2.2.1 常量

★字符串常量和字符常量的主要区别:

- 字符常量由单引号括起来, 字符串常量由双引号括起来。
- 字符常量只能是一个字符, 字符串常量可以含多个字符。
- 字符常量占一个字节, 字符串常量占的字节数等于字符个数加1 `printf("he said \ " I am a student.\ " \n");`
- 比较输出: `A'` 与 `"A"` 的区别

`'A'`
A
占一个字节

`"A"`
A \0
占两个字节

➤ 符号常量

- 符号常量：用标识符表示的常量

- 格式：

#define 标识符 常量

- 功能： 用该标识符代表后面的常量值

- 例： #define PI 3.1415926

#define STAR '*'

预处理命令**#define**也称为**宏定义**，一个**#define**命令只能定义一个符号常量，用一行书写，**不用分号结尾**。

【例2.2】求半径为r的圆面积和圆周长

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

```
#define PI 3.1415926 //用预处理命令定义符号常量
```

```
void main()
```

```
{ float r, area, l;      //定义变量类型为实型
```

```
    scanf( "%f" , &r);    //输入r的值
```

```
    area=PI*r*r ;
```

```
    l=2*PI*r ;
```

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

```
}
```

➤ 符号常量

• 说明:

- 符号常量名习惯用**大写字母**表示
- 该命令通常**放在文件头**
- 在程序中，符号常量**不允许重新赋值。**
- 例: `#define PI 3.1415926`

`PI=5.6; × 或 scanf("%f " , &PI); ×`



2.2.2 变量



变量概述



整型变量



实型变量



字符型变量

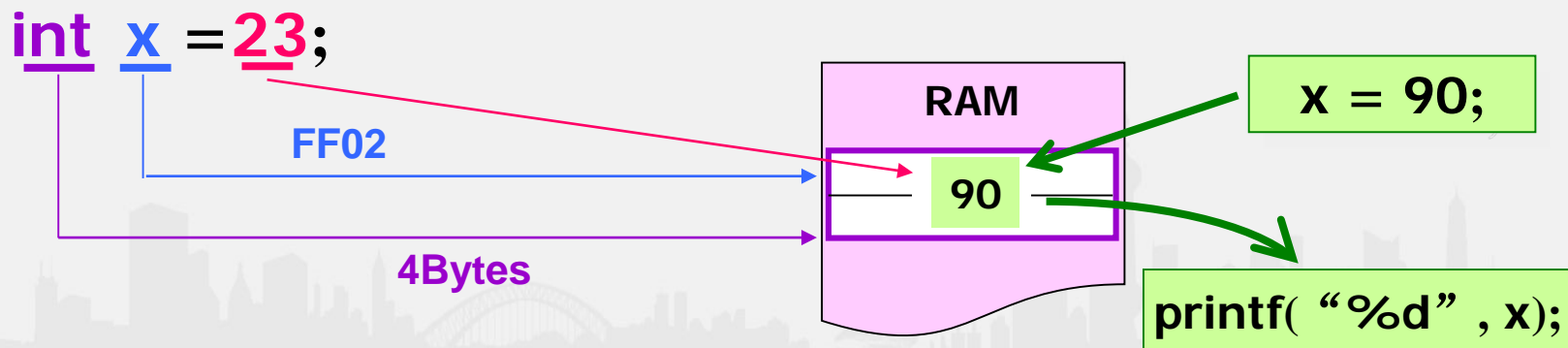


变量的初始化



➤ 变量概述

- 变量: 在程序执行期间其值**可以改变**的量
- 变量**在使用之前必须被声明**
 - 每一个变量有名字，类型，长度和值。
 - 对变量赋值过程是“覆盖”过程，用新值去替换旧值。
 - 从内存中读出变量的值，该变量保持不变。



2.2.2 变量

➤ 变量概述

- 定义变量的形式:

类型说明符 变量名表;

例:

变量类型
(type)

int a, b ; /*定义两个整型变量a和b*/

char c ; /*定义字符型变量c*/

float f1, f2; //定义单精度实型变量f1和f2

C语言中规定，标识符只能由字母、数字和下划线三种字符组成，且第一个字符必须是字母或下划线，而且标识符不能是C语言的关键字。

2.2.2 变量

➤ 整型变量 一用来存放整型数据的变量

整型变量的分类表

类型名称	类型说明符	字节数	数值范围
基本整型	<code>int</code>	4	-2147483648~ 2147483647
短整型	<code>short [int]</code>	2	-32768~ 32767
长整型	<code>long [int]</code>	4	-2147483648~ 2147483647
无符号基本整型	<code>unsigned [int]</code>	4	0~ 4294967295
无符号短整型	<code>unsigned short [int]</code>	2	0~ 65535
无符号长整型	<code>unsigned long [int]</code>	4	0~ 4294967295

与操作系统、编译系统、
机器字长有关。

• 有符号的整数 — 最高位是符号位

- 正整数在内存中以二进制**原码**形式存放。
- 负整数在内存中以二进制**补码**的形式存放。

— 有符号正整数 10

0	00000000	00001010
---	----------	----------

— 有符号负整数 -10

1	11111111	11110110
---	----------	----------

-10的原码

1	00000000	00001010
---	----------	----------

按位取反

1	11111111	11110101
---	----------	----------

加1后得到-10的补码

1	11111111	11110110
---	----------	----------

• 无符号整数

- 无符号整数的**所有二进制位**全部用来存放数值，**不能存放负数**。

无符号整数

65535_u

1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1

RAM

11111111

11111111

例：求50的三次方

```
#include <stdio.h>
void main( )
{ short int x;
  x=50*50*50;
  printf( "%d\n" , x );
}
```

程序运行结果为： -6072 (错)

将以上程序改为：

```
#include <stdio.h>
void main()
{ int x ;
  x=50*50*50 ;
  printf( "%d\n" , x) ;
}
```

运行结果：125000

- 因此, 在定义整型变量时, 要注意数据类型允许的数值范围。

➤ 实型变量

- 用来存放实型数据的变量
- 分单精度型、双精度型和长双精度型三类

变量类型名	变量类型	所占字节数	数的范围	有效数字
单精度实型	float	4	$10^{-38} \sim 10^{+38}$	7
双精度实型	double	8	$10^{-308} \sim 10^{+308}$	16
长双精度型	long double	10	$10^{-4932} \sim 10^{+4932}$	19

2.2.2 变量

【例2.3】实型变量的使用

```
#include <stdio.h>
void main()
{ float f;           // f为单精度实型变量
  double d;          // d为双精度实型变量
  f=33333.3333f;
  d=3333333333333333.333333;
  printf( "f=%f\nd=%f\n" , f, d);
} 输出结果:  f=33333. 332031
              d=3333333333333333.332000
```

无效数字

2.2.2 变量

➤ 字符变量

- 用来存放字符常量的变量

- 例: `char c1, c2, c3 ;`

`c1 = 'A' ; c2 = '\n' ; c3 = 97;`

- 每个字符变量分配一个字节用于存放一个字符。

- 字符数据与整型数据可相互赋值，直接运算。
- 可以把字符变量按整型量输出，也允许把整型量按字符量输出。

c1

0 1 0 0 0 0 1

'A' 的ASCII码(值)为65

c2

0 0 0 0 1 0 1 0

'\n' 的ASCII码(值)为10

【例2.4】字符型变量的使用

```
#include <stdio.h>
void main( )
{ int  a= 'b' ;           //给一个整型变量赋一个字符值
  char c1=97;             //给一个字符变量赋一个整数值
  c1=c1-32;               //将小写字母转换为大写字母
  printf( "%d %d\n" , a , c1);
  printf( "%c %c\n" , a , c1);
}
```

运行结果

98	65
b	A

注意：

- 只能将一个字符常数赋给一个字符变量，**不能**把一个字符串常量赋予字符变量。

例： `char c = "abc" ;` ×

- 在C语言中没有字符串变量

（但可以用一个字符数组来存放一个字符串常量）

➤ 变量的初始化

- 在说明变量的同时给变量赋初值

```
int a;
```

- 格式:

```
a=5; //赋初值
```

类型说明符 变量 = 常数 ;

- 例: `int a=5;` //定义并初始化

```
char c1= 'a' , c2= 'B' ;
```

```
double area, r=23.e-2 ;
```

```
int x=10, y=10, z=10 ;
```

//不能写成`int x=y=z=10;`

2.2.3 标准函数

- C编译系统提供的库函数

- 包括: **输入输出函数、数学函数、字符和字符串函数等。**

- 数学函数

- **sin(x)** **x为弧度, double x, double sin(x)**

- **cos(x)** **cos(x)**

- **exp(x)** e^x

- **log(x)** $\log_e x$

- **log10(x)** $\log_{10} x$

- **fabs(x)** $|x|$

- **pow(x, y)** x^y

- **sqrt(x)** $\sqrt{x}, x \geq 0$

2.2.3 标准函数

【例2.5】 求三角形面积

$$\text{area} = \sqrt{s(s-a)(s-b)(s-c)}, \quad s = (a+b+c)/2$$

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

```
#include <math.h> //预编译命令, 将系统提供的数学函数  
//作为头文件包含到用户源文件中
```

```
void main( )
```

```
{ double a, b, c, s, area;
```

```
scanf("%lf,%lf,%lf", &a,&b,&c);
```

```
s=(a+b+c)/2;
```

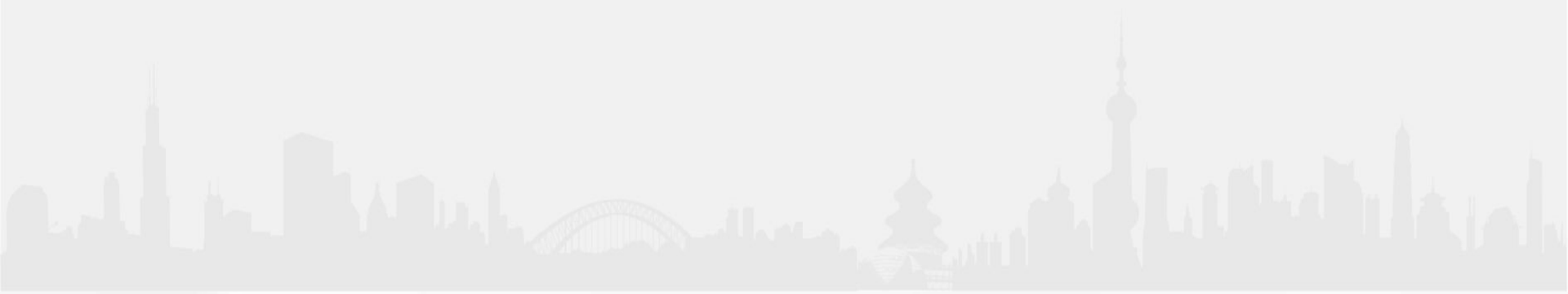
```
area=sqrt(s*(s-a)*(s-b)*(s-c));
```

```
printf("the area is%6.2f\n", area);
```

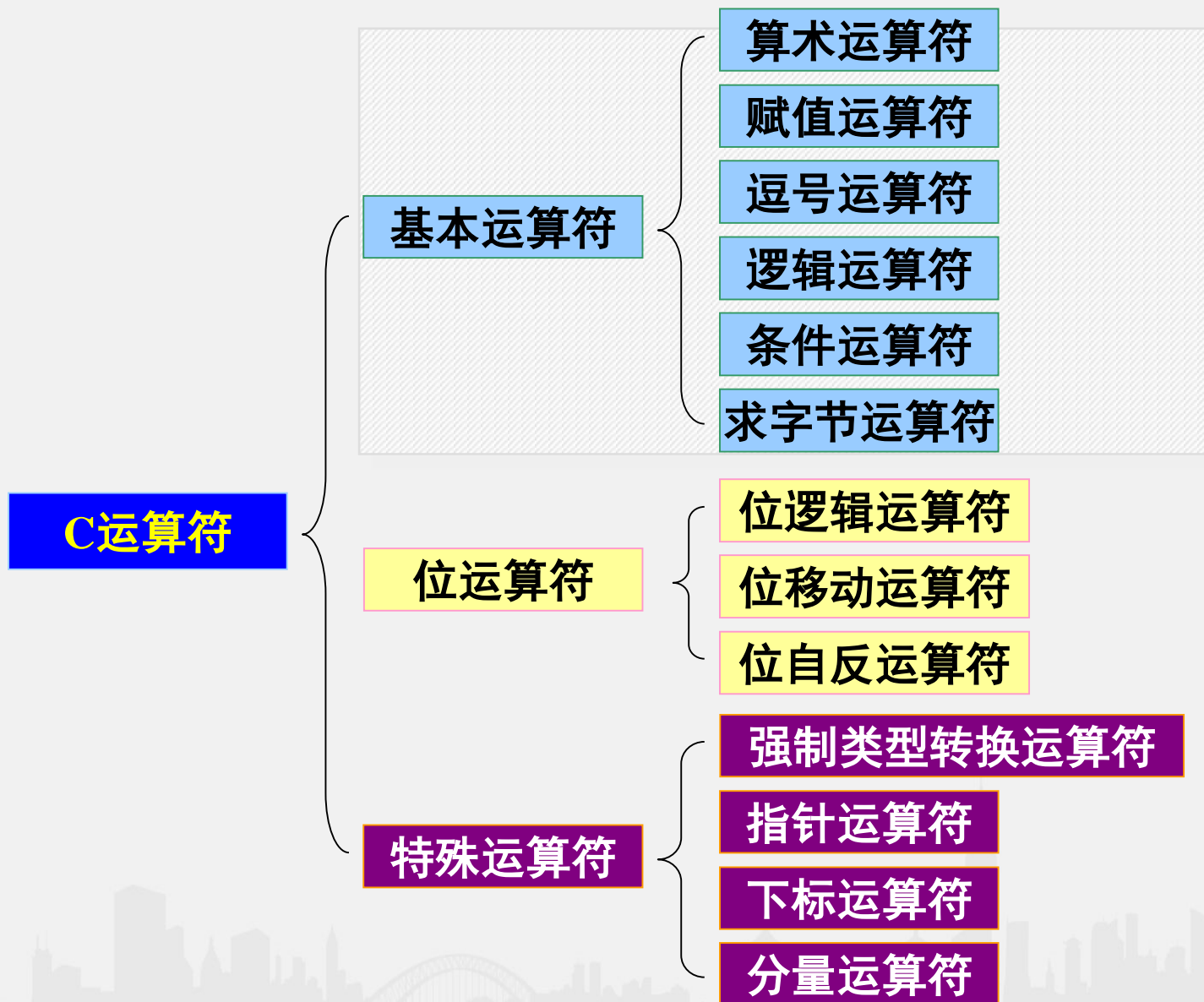
```
}
```


Part.3

2.3 运算符和表达式



2.3 运算符和表达式



2.3 基本运算符及其表达式

2.3.1 算术运算符与算术表达式

2.3.2 不同数据类型的转换与运算

2.3.3 关系运算符与关系表达式

2.3.4 逻辑运算符与逻辑表达式

2.3.5 条件运算符与条件表达式

2.3.6 赋值运算符与赋值表达式

2.3.7 逗号运算符与逗号表达式



2.3.1 算术运算符与算术表达式

➤ 基本算术运算符

- 算术运算符用于各类数值运算。包括**基本算术运算符**、**自增运算符**、**自减运算符**。下表为基本算术运算符。

运算符	运算规则	操作数数目	优先级	结合方向
-	负号	单目	2	右结合
+	加法	双目	4	左结合
-	减法	双目	4	左结合
*	乘法	双目	3	左结合
/	除法	双目	3	左结合
%	求余或模	双目	3	左结合

2.3.1 算术运算符与算术表达式

在C语言中,规定了运算符的**优先级**和**结合性**。

- **优先级:**

C语言中,运算符的优先级共分为**15级**。**1级最高, 15级最低**。表达式求值时, **先做优先级高的操作**。

如: **$d = a + b * c$** ;

当运算符的**优先级别相同**时, 运算次序由**结合性**决定。

- **结合性:** **左结合性**(先左后右)例: **$d = 3 * 5 / 4$** ;

右结合性(先右后左)例: **$d = a = 3$** ;

2.3.1 算术运算符与算术表达式

➤ / 运算符

① 整数相除截去余数, 此运算为整除。

$$5/2=2 \quad (\neq 2.5) \quad 1/3+1/3+1/3=0$$

② 对于浮点数则为通常意义的除法

$$5.0/2.0=2.5 \quad 1./3+1./3+1./3=1$$

2.3.1 算术运算符与算术表达式

➤ % 模运算符

- 用于计算两个数相除后得到的**余数**。

如: $a\%b$ 求a除以b以后得到的余数。

- 运算对象只能是**整型量**, 其结果也是**整型量**。

如: $5\%2=1$ $3\%3=0$ $3\%5=3$

$'A'\%2=1$ $3.0+10.0\%5$ ✕

- 所得结果的符号与运算符**左**侧操作数的符号相同

$-5\%2=-1$ $5\%-2=1$

2.3.1 算术运算符与算术表达式

➤ 自增、自减运算符

含义	运算符	优先级	结合性	功能
自增	++	2	右结合	使变量的值增1
自减	--	2	右结合	使变量的值减1
使用形式	① ++i 或 --i 变量i先自增或自减1, 再引用i.			
	② i++ 或 i-- 先使用变量i, 再自增或自减1.			

例: $x = 10$; $y = ++x$; 此时, $y = 11$

若: $x = 10$; $y = x++$; 则 $y = 10$

在这两种情况下, x都被置为11。

2.3.1 算术运算符与算术表达式

例:

K=3; j=5; i=3;

m=(++k)*j ;

n=(i++)*j ;

m= 20 k= 4

n= 15 i= 4

2.3.1 算术运算符与算术表达式

【例2.7】 自增、自减运算符的使用。

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

```
void main( )
```

```
{ int i=6, a, b;
```

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

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

```
    a=--i ; printf( "%d\n" , a );
```

```
    b=i--; printf( "%d\n" , b );
```

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

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

```
}
```

7
7
7
7
-6

i = 7

2.3.1 算术运算符与算术表达式

注意：

- 自增、自减运算只能用于**变量**, 不能用于常量和表达式。
- 自增、自减运算符是两个+或两个-的一个整体, 中间不能有空格。如果有多于两个+ 或两个-连写的情况, 则编译首先识别**前两个**+或-为增量或减量运算符。
- 表达式 $x+++y$ 等价于 $(x++)+y$
- 自增、自减运算符的运算顺序是**右**结合, 因此对 $-i++$ 应理解为: $-(i++)$, 而 $(-i)++$ 是非法的。

例: $i=3$;
 $\text{printf}(\text{"\%d"}, -i++);$ 输出:**-3**

2.3.1 算术运算符与算术表达式

【例2.8】自增、自减运算符的使用

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

```
void main( )
```

```
{ int i, j, k ;
```

```
    i=1;
```

```
    j=1;
```

```
    k=i+++j ;
```

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

```
}
```

程序运行结果：

i=2, j=1, k=2

k=(i++)+j;

2.3.1 算术运算符与算术表达式

➤ 算术表达式

- 由**算术运算符**和**括号**将运算对象(如常量、变量、函数等)连接起来的一个有值的式子。

例: **'A' *2-sqrt(4)/-d**

- 表达式求值的优先次序:

() → 函数 → ++、-- → *、/、% → +、-
高 —————→ 低

2.3.1 算术运算符与算术表达式

书写问题:

① “/” 号, 如: $\frac{a+b}{c+d} \rightarrow (a+b)/(c+d)$

② “*” 不能省略, 如: $2(a+b) \rightarrow 2*(a+b)$

③ 括号只能使用圆括号, 且成对出现, 不能使用 [] 和 { }。

如: $a[x+b(y+c)] \rightarrow a*(x+b*(y+c))$

2.3.1 算术运算符与算术表达式

例: 与数学式子3乘以x的n次方除以(2x-1)对应的C语言表达式是_____。

A> $3*x^n/(2*x-1)$

B> $3*x**n/(2*x-1)$

✓ C> $3*pow(x,n) *(1/(2*x-1))$

D> $3*pow(n, x) /(2*x-1)$

2.3.2 不同数据类型间的转换与运算

- 在C程序中，当**不同类型的量**进行运算时，要**转换成同一种类型**然后再进行运算。

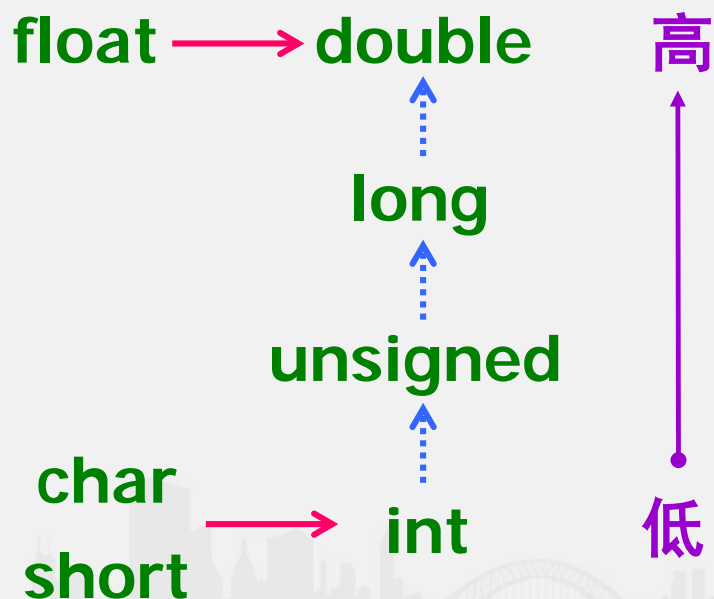
例： $10 + 'a' + 1.5 - 8765.1234 * 'b'$

- 转换方式：
 - 自动转换**：数据类型**自动**由**低级向高级**转换。
 - 强制转换**：将表达式的运算结果**强制转换**成指定的数据类型。

2.3.2 不同数据类型间的转换与运算

➤ 自动类型转换

- 这种类型转换由编译系统**自动完成**
- 转换规则:



例:

```
float f=3.5;
```

```
int n=6;
```

```
long k=21;
```

```
double ss=f*n+k/2;
```

2.3.2 不同数据类型间的转换与运算

➤ 强制类型转换

- 格式:

(数据类型说明符)(表达式)

- 注意:

- 强制转换属单目运算, 运算优先级为2。
- 强制转换得到的是中间结果类型, 原变量类型不变。
- 数据类型说明符和表达式都必须加括号(单个变量除外)
- 例: `int x, y; float z;`
`(float)(x+y);`
`(int)z+x;`
`(float)(5%3);` (将5%3的值转换为float型)

2.3.2 不同数据类型间的转换与运算

【例2.12】强制类型转换的应用

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

```
void main( )
```

```
{ int x=8;
```

```
float f=6.85;
```

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

```
printf( "(int)f=%d, f=%f\n" , (int)f %4, f );
```

```
}
```

X, f的类型
并未改变

(float)x=8.000000, x=8
(int)f=2, f=6.850000

2.3.3 关系运算

- 即比较两个量的大小, 比较的结果为“真”或“假”。

例: $a > 3$

如果 $a=8$, 则结果为“真”; 如果 $a=1$, 则结果为“假”。

1. 关系运算符

含义	运算符	优先级	结合性
小于	<	6	左结合
小于等于	<=		
大于	>		
大于等于	>=		
等于	==	7	
不等于	!=		

2.3.3 关系运算

➤ 关系表达式

- 用关系运算符将两个表达式连接起来的式子。

- 格式:

〈表达式1〉 〈关系运算符〉 〈表达式2〉

其中: 表达式1和表达式2可以是任意表达式。

- 例: $5 > (4 < 5)$ 值为: 1
- $'a' > 'b' + 3$ 0
- $(a=4) >=(b=6)$ 0
- $5 > 2 > 7 > 8$ 0

2.3.3 关系运算

例2.13

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

```
void main( )
```

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

```
    a=b==c ;
```

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

```
    a== ( b=c++*2 ) ;
```

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

```
    a= b>c>=100 ;
```

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

```
}
```

输出结果:

1 , 10 , 10

1 , 20 , 11

0 , 20 , 11

➤ 关系运算的不足

- 关系表达式只能表达简单的关系，如：

$\text{sum} \geq 1500$

$y \neq z$

即只能对一个条件进行测试。

而以下关系：

$0 < x < 5$

则不能用关系表达式表示。

$x > 0 \&\& x < 5$

2.3.4 逻辑运算符与逻辑表达式

➤ 逻辑运算符

C逻辑运算符



含义	运算符	优先级	结合性
逻辑非	!	2	右结合
逻辑与	&&	11	左结合
逻辑或		12	左结合

2.3.4 逻辑运算符与逻辑表达式

● 逻辑运算真值表

逻辑非!

a	!a
1	0
0	1

逻辑与&&

a	b	a&& b
1	1	1
1	0	0
0	1	0
0	0	0

特点: 全真为真,
其余为假。

逻辑或||

a	b	a b
1	1	1
1	0	1
0	1	1
0	0	0

特点: 全假为假,
其余为真。

2.3.4 逻辑运算符与逻辑表达式

➤ 逻辑表达式

- 用逻辑运算符将表达式连接起来的式子。
- 形式:

[<表达式1>] <逻辑运算符> <表达式2>

表达式1和表达式2可以是任何表达式

如: $5 > 3 \&\& 2 || 4 - !' a'$

逻辑表达式的值
整数 1: true
0: false

- 判断时, 0 代表“假”, 非0即表示“真”。

2.3.4 逻辑运算符与逻辑表达式

将下面的条件用C语言的逻辑表达式表示

例1: $1 \leq x \leq 10$ 且 $x \neq 7$

$x \geq 1 \ \&\& \ x \leq 10 \ \&\& \ x \neq 7$

例2: y 能被4整除但不能被100整除, 或 y 能被400整除。

$(y \% 4 == 0 \ \&\& \ y \% 100 != 0) || (y \% 400 == 0)$

2.3.4 逻辑运算符与逻辑表达式

● 运算顺序:

()→**!**→**算术运算**→**关系运算**→**&&**→**||**→**赋值运算**

例: 若 $a=2$, $b='a'$, $c=5$, $f=3.0$

$c > 3 \&\& 8 < 4 - !0$ 0

$f/3 \&\& a - b$ 1

$!(b - 'a')$ 1

$(a=7) > 6 \&\& (b=-1) > 6$ 0

2.3.4 逻辑运算符与逻辑表达式

●逻辑与和逻辑或运算符具有短路能力

— 逻辑与: (表达式1) && (表达式2) && ...

只有表达式1的值为“真”时才求表达式2的值

如: $x=y=-1;$

$++x \ \&\& \ ++y;$

$/* \ x=0 \ \ y=-1 \ */$



2.3.4 逻辑运算符与逻辑表达式

● 短路表达式

— 逻辑或: (表达式1) || (表达式2) || ...

只有表达式1为假时才判断表达式2的值

例:

```
void main()  
{ int num=3;  
  5>4||(num=0);  
  printf( "num=%d\n" , num);  
}
```

输出: num=3

2.3.5 条件运算符与条件表达式

- 条件运算符: **?** :

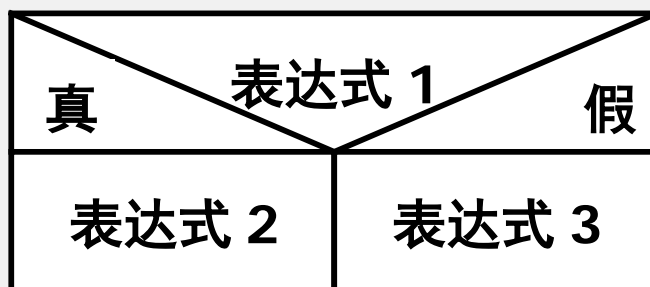
- 三目运算符、**右结合**、13级。

- 条件表达式:

<表达式1> ? <表达式2> : <表达式3>

例: `max = a > b ? a : b ;`

- 执行过程:



2.3.5 条件运算符与条件表达式

□ 说明:

- 在程序中常把条件表达式的值赋给某个变量, 如:

①将ch中字母转换为小写字母

```
char ch=getchar( );
```

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

②将x 的绝对值赋给 y

```
y= x>=0 ? x : -x ;
```


2.3.5 条件运算符与条件表达式

□ 说明:

- 在条件表达式中, 各表达式的类型**可以不同**, 此时, 条件表达式值的类型为表达式2和表达式3中较高的类型。

如:

$x > y ? 1 : 1.5$

条件表达式的值为浮点型

2.3.5 条件运算符与条件表达式

□ 说明:

- 条件运算符可以嵌套, 这种嵌套是右结合的。

例:

```
int a=15, b=20, c=25, d=30, e ;
```

```
e= a>b ? c : (c>d ? b : d);
```

```
e=30
```

2.3.5 条件运算符与条件表达式

例2.15 写出运行结果。

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

```
void main( )
```

```
{ int  a=1, b=1, c=1;
```

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

```
    printf( "(1)%d\n" , a>b ? a : b );
```

```
    printf( "(2)%d\n" , a>c ? a-- : c++);
```

```
    (a>=b>=c) ? printf( "a\n" ) : printf( "b\n" );
```

```
    printf( "(4)%d, %d, %d\n" , a, b, c );
```

```
}
```

运行结果：

(1)2

(2)3

b

(4)2, 2, 4

2.3.5 条件运算符与条件表达式

例2.16：任意输入三个数，输出最大值。

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

```
void main()
```

```
{ int a , b , c , max ;
```

```
    scanf( "%d%d%d" , &a, &b, &c);
```

```
    max= a>=b ? (a>=c ? a : c ) : (b>=c ? b : c );
```

```
    printf( "最大数为: %d\n" , max);
```

```
} 程序运行如下：
```

25 -7 48✓

最大数为: 48

2.3.6 赋值运算符和赋值表达式

- 赋值运算符分**两种**:
 - 简单赋值运算符: **=**
 - 复合赋值运算符: 如 **+=**、**-=**、***=**、**/=**、**%=** 等
 - 优先级**14**, **右**结合性。
- 赋值表达式: 由赋值运算符连接起来式子
 - 作用: 将表达式的值赋给变量

2.3.6 赋值运算符和赋值表达式

➤ 简单赋值运算符和赋值表达式

- 简单赋值运算符为 “=” ; 由 “=” 连接的式子为 (简单) 赋值表达式。
- 格式: $\langle \text{变量} \rangle = \langle \text{表达式} \rangle$
- 作用: 把右边表达式的值, 赋给左边的变量。

例: $k = i + + + - j$ (i=2, j=3, k的值为 4)

$x = (a = 5) + (b = 8)$ (x的值为 13)

$d = e = 2.1$

- 执行顺序是: 先计算, 再赋值。

2.3.6 赋值运算符和赋值表达式

例：指出下列赋值表达式的错误

(1) $\underline{-s} = x + y$

(2) $\underline{5} = n$

(3) $a = \underline{a*3} = 2$

(4) $w = 3.64E\underline{2.1}/5.87$

(5) $\underline{\cos(\beta)} = a/\text{sqrt}(a*a + b*b)$

(6) $\underline{k} = 12.3\%4$

2.3.6 赋值运算符和赋值表达式

- 赋值运算的**类型转换**，规则为：

- 实型数据赋给整型变量时，只取整数部分。

如： `int i=1.5*2/2;` (**i值为1**)

- 整型数据赋给实型变量时，以浮点形式取值。

如： `float x=1/4+1/4+1/4;` (**x的值为0.000000**)

- 字符型数据赋给整型变量时，整型变量的高位补的数与char的最高位相同，低八位为字符的ASCII码值。


如： `K= 'A'` (**k值为65**) k: 0000 0000 **0**100 0001

- 整型数据赋给字符型时，只把低8位赋给字符变量。

2.3.6 赋值运算符和赋值表达式

例2-13:

```
void main()  
{ short int x=300;    x: 0000000100101100  
  char y;  
    y=x;  
    printf( "y=%d y=%c\n" , y, y );  
}
```



输出结果: **y=44**

2.3.6 赋值运算符和赋值表达式

➤ 复合赋值符及表达式

- 复合赋值运算符是在简单赋值运算符前加双目运算符构成。

共**10**种： $+=$ 、 $-=$ 、 $*=$ 、 $/=$ 、 $\%=$ 等。

例：

$c=c+3$ \longrightarrow $c+=3$

$x*=y+7$ \longrightarrow $x=x*(y+7)$

设 x, y, z 的初始值分别为：10, 20, 30

$x+=y+=z*z$ \longrightarrow $x=x+(y=y+z*z)$ 结果为：930

设 t 的值为5

$t+=t-=t*t$ \longrightarrow $t=t+(t=t-t*t)$ 结果为：-40

2.3.6 赋值运算符和赋值表达式

例2-14

```
void main()  
{ int a, b, c;  
  a=b=c=5;  
  printf(" a=%d, b=%d, c=%d\n" , a,b,c);  
  a=(b=4)+(c=6) ;  
  printf(" a=%d, b=%d, c=%d\n" , a,b,c);  
  a+=b+c ;  
  printf(" a=%d, b=%d, c=%d\n" , a,b,c);  
  a+=a-=a*=a ;  
  printf(" a=%d\n" ,a) ;  
}
```

运行结果:

```
a=5, b=5, c=5  
a=10, b=4, c=6  
a=20, b=4, c=6  
a=0
```

2.3.7 逗号运算符和逗号表达式

- **逗号运算符运算符：** “ , ”
 - 优先级**15**级（最低），**左**结合
- **用逗号运算符将各表达式连接起来的式子为逗号表达式。**
 - **格式：**
<表达式1> , <表达式2> , ... , <表达式n>
 - **如：** $a = 3 * 5, a * 10, a + 8$
 - **求解过程：** 先求表达式1的值, 再求表达式2的值, 最后求表达式n的值, **表达式n的值作为整个逗号表达式的值。**

2.3.7 逗号运算符和逗号表达式

例: 逗号表达式的应用

```
#include <stdio.h>

void main( )
{ int  a=2, b=4, c, y;
  y=(c=a*b , b+c) ;
  printf(“y=%d\n”, y );
}
```

输出结果:

y=12

2.3.7 逗号运算符和逗号表达式

例:

1) $(a=3*5, a*4), a+5$ **20**

2) $a=5, a*=a, a+5$ **30**

3) $\text{int } a, b;$
 $a=2, b=5, a++, b++, a+b;$ **9**

4) $\text{int } x=10, y=3, z;$
 $\text{printf}(\text{"\%d\n"}, z=(x\%y, x/y));$ **3**

2.3.7 逗号运算符和逗号表达式

说明：

(1) 用一个逗号表达式语句，可代替多个赋值语句，如

a=0; b=1; c=2; 可写成: **a=0, b=1, c=2 ;**

(2) 在变量说明和函数参数表中逗号只是起分隔符作用

```
printf( “%d,%d,%d” , a , b ,c );
```

```
printf( “%d,%d,%d” , (a, b, c) , b ,c);
```