现代制造学院 | 香港铸业学院

C 语言

授课主题

授课主题的详细内容

教学目标

教学目标的详细内容

教学重点、难点与解决方案

重点：

难点：

学时分配

技术学院
INGNAN
I C

目 录

数据类型、运算符、表达式

01

PART ONE

C 语言的数据类型

- 类型
  - 基本类型
    - 整型
      - 短整型 short int
      - 整型 int
      - 长整型 long int
    - 实型
      - 单精度 float
      - 双精度 double
    - 字符型 char
    - 枚举型 enum
  - 构造类型
    - 数组类型
    - 结构体类型
    - 共用体类型
  - 指针类型
  - 空类型

常量和符号常量

1、常量： 在程序运行过程中，其值不能被改变的量。

每种数据类型都有常量，也都有变量。如整型常量、浮点型常量．例如：

12、4 、-67 或 3.14 5.6 或 ’a’ 、 ’d‘

从上可以看出常量可以是不同类型的．

2、常量分为两种直接常量和符号常量。

直接常量直将数值直接使用，如： a=b*10;

当用一个标志符代表一个常量时，称为符号常量．

例：

```
#define PRICE 30
main()
{
 int num, total;
 num=10;
 total=num* PRICE;
 printf("total=%d",total);
}
```

上例中的 PRICE　就称为符号常量；凡在程序中出现的 PRICE　都代表 30 。

```
PRICE=PRICE+5; /* 正确否 */
```

一般情况下，符号常量名用大写、变量用小写。

3、符号常量的声明和使用

一般放在程序的前端，与 #include <...> 在一块：

#define 符号常量名 值

如： #define Zero 0

好处：

含义清楚 (#define Second 60)

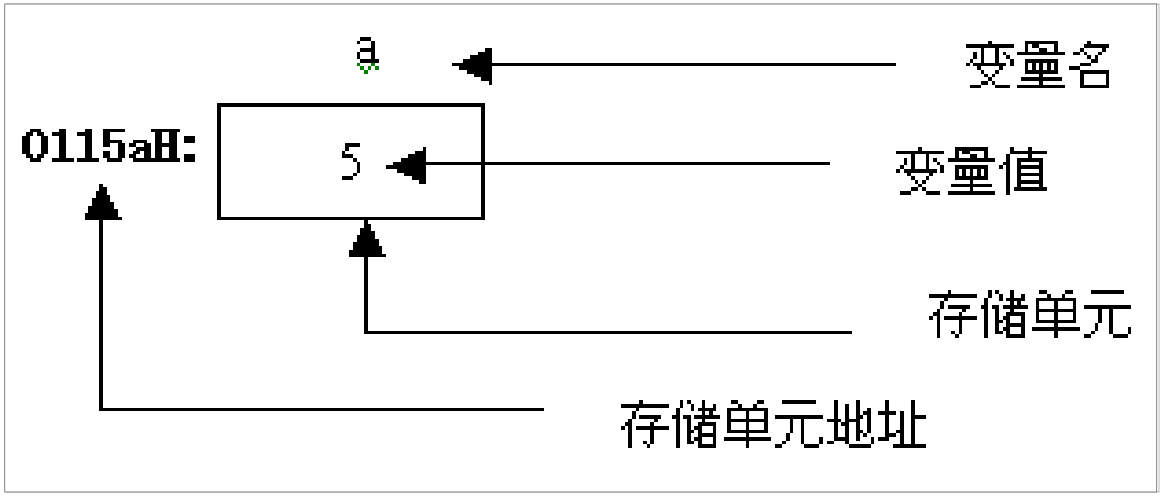
一改全改 (#define PI 3.14)

2.2 变量

- 在程序运行过程中，其值可以改变的量称为变量。具有三个基本要素：名字、类型和值。
- 变量的名字是用标志符来表示的。标志符只能由字母、数字和下划线以及一系列字符组成，不能以数字开头，关键词不能作为变量名，大小写敏感

- 变量类型就是变量所存储数据的类型，其目的是：指定一个变量的类型不仅决定了该变量存储在内存中所占的空间的大小，而且也规定了该变量的合法操作，以及检测该变量的正确性
- 给变量赋值，有两种途径。一种是在定义的同时用赋值运算符“ =”给变量赋值，称之为变量的初始化，它是系统在编译时完成的。另一种在程序执行后，执行语句动态地改变变量的值

注意：变量名与变量值的区别



在C语言中，要求对所有用到的变量作强制定义，即“先定义、后使用”。

因为：

1、 编译程序不能翻译未定义变量。

2、 编译程序在编译时根据变量类型确定存储单元的数量并分配空间

3、编译程序在编译时根据变量类型进行语法检查。例，整型变量a、b可以进行“求余”运算 a % b；若把a、b定义为实数，则上述运算非法。

```
main() {
  int i, studentNo;
  i=5;
  studentno=5;      /* 错在哪里 */
  j=3;                /* 错在哪里 */
}
```

一条变量说明语句由数据类型和其后的一个或多个变量名组成。变量说明的形式如下：

类型 〈变量表〉;

变量表是一个或多个标识符名， 每个标识符之间用”,”分隔。

例如：

```
int i
int i, j, k;
char c, str[5], *p;
```

2.3 标识符

所谓标识符是指常量、变量、语句标号以及用户自定义函数的名称。

C 语言标识符的定义十分灵活。作为标识符必须满足以下规则：

1. 所有标识符必须由一个字母 (a~z, A~Z) 或下划线 (_) 开头；

2. 标识符的其它部分可以用字母、下划线或数字 (0~9) 组成；

3. 大小写字母表示不同意义， 即代表不同的标识符；

4. 标识符只有前 32 个字符有效 (Turbo C)

5. 标识符不能使用 Turbo C 的关键字。

下面举出几个正确和不正确的标识符：

正确 ---------- 不正确

smart------ 5smart
_decision-- ----------bomb?
key_board------ ------key.board
FLOAT---------- ---float

三、整型数据

3.1. 整型常量

整型常量即整常数按不同的进制区分， 整型常数有三种表示方法：

十进制数： 以非 0 开始的数 如 :220, -560, 45900

八进制数： 以 0 开始的数 如 :06; 0106, 05788

十六进制数： 以 0X 或 0x 开始的数 如 :0X0D, 0XFF, 0x4e

```
main(){
  int i;
  i=11;
  printf("i=%d\n",i);
  i=011;
  printf("i=%d\n",i);
  i=0x11;
  printf("i=%d\n",i);
  i=0X11;
  printf("i=%d\n",i);
}
```

程序的输出结果是多少？

注意：

- 可在整型常数后添加一个 "L" 或 "l" 字母表示该数为长整型数，如 22L, 0773L, 0Xae4l 。
- 另外，　所有整数的缺省类型是 int ，可在整型常数后添加一个 "L" 或 "l" 字母表示该数为长整型数，　如　22L,0773L, 0Xae4l 。
- 若加上一个 "u" 或 "U" 字母表示该数为无符号整型数，如 27u, 0400u, 0xb8000000u 。
- 若加上一个 "ul" 或 "UL" 字母表示该数为无符号长整型数，如 27ul, 0400UL, 0xb8000000UL 。
- 当整数的值超出 int 类型所能表示的范围时称为整数溢出。

3.2、整型变量

1．整型变量的分类：加上不同的修饰符，整型变量有以下几种类型；

C 语言标准没有规定整型变量在计算机内存中所占的字节数，它与具体的机器和操作系统有关，在 16 位的计算机中，int 占 2 字节，在 32 位系统中占 4 字节

	所占位数	数的范围
[signed] int	16	-32768~+32767
[signed] short	16	-32768~+32767
[signed] long	32	-2147483648~+2147483647
unsigned int	16	0~65535
unsigned short	16	0~65535
unsigned long	32	0~4294967295

2. 整型变量的定义

可以用下列语句定义整型变量：

```
int a, b;
/*a 、 b 被定义为有符号短整型变量 */
unsigned long c;
/*c 被定义为无符号长整型变量 */
unsigned short c,d;
/* 指定变量 c 、 d 为无符号短整型 */
long e,f;  /* 指定变量 e 、 f 为长整型 */
```

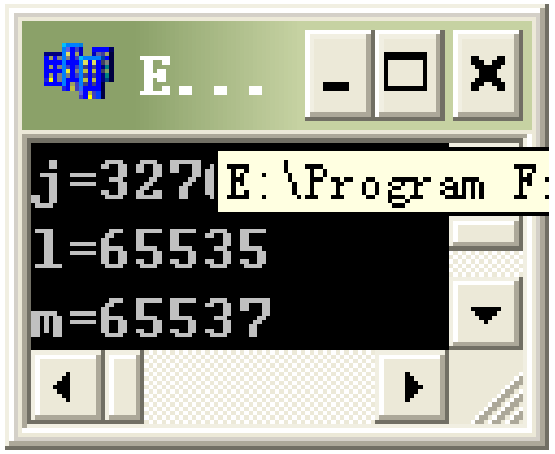
程序中在函数的开头部分定义变量。

```
main()
{
  int a,b,c,d;  /* 指定变量 a 、 b 、 c 、 d 为整型 */
  unsigned u;   /* 指定变量 u 为无符号整型 */
  a = 12; b = -24; u = 10;
  c = a + u; d = b+u;
  printf("a+u = %d, b+u = %d\n", c,d);
}
```

程序运行显示： a+u = 22, b+u = -14

3. 整型变量的溢出

```
main(){
 int i,j;
 unsigned k,l,m;
 i=32767;
 j=i+1;
 l=65535u;
 m=l+2;
 printf("j=%d\n",j);
 printf("l=%u\n",l);
 printf("m=%u\n",m);
}
```



象这类问题体现了 c 语言灵活性所带来的副作用。
这种错误系统不给提示，由程序员自己控制。

四、实型数据

4.1 实型常量

实数（ real ）又称为浮点数（ float ），有两种表达方式：

普通 (十进制) 方式： 0.123 、 .123 、 123.0 、 123. 、 0.0 （必须有小数点）

指数方式： 123e3 或 123E3 、 123E-2

注意：字母 e(或 E) 之前必须有数字

e 后面指数必须为整数

规范化指数形式： e 前的小数部分中，小数点左边只能有一位非 0 数字。如下面例子应为： 1.23e5

错误： 0.123e6

4.2 实型变量

精确程度

		占内存单元	有效数字	取值范围
单精度实型变量	float	4 字节(32 位)	6-7	10^{-37}-10^{38}
双精度实型变量	double	8 字节（64 位）	15-16	10^{-307}-10^{308}
长双精度实型变量	long double	80 位	18-19	10^{-4931}-10^{4932}

例：

```
float x,y;    /* 指定 x 、 y 为单精度实型变量 */
double z;     /* 指定 z 为双精度实型变量 */
```

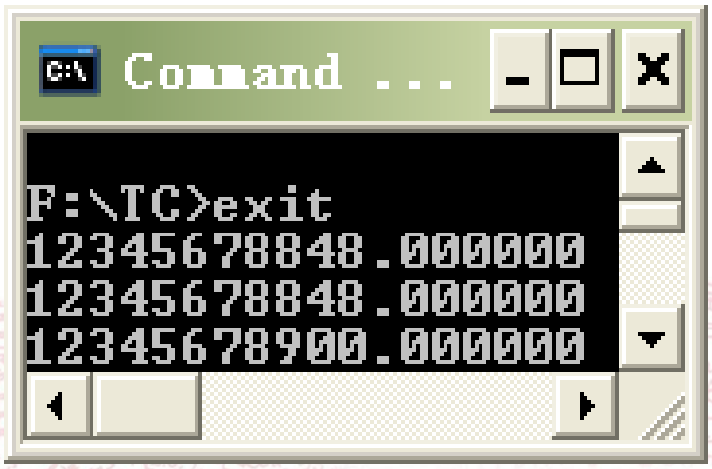
实型常量不分 float 和 double 。

问题：如何用图示的方法表示这些变量的内存空间？

4.3 实型变量的误差与精度

```
main(){
  float a,b;
  a=123456.789e5;
  b=a+20;
  printf("%f\n",a);
  printf("%f\n",b);
}
```

注意输出结果




```
main(){
  float a;
  double b;
  a=111111.111;
  b=111111.111;
  printf("%f\n",a);
  printf("%f\n",b);
}
```

注意输出结果

五、字符型数据

5.1 字符常量

用单引号 (撇号) 括起来的一个字符。
如：'a' 、 'x' 、 'D' 、 '?' 、 '$' 。
注意，‘a’ 和 ‘A’ 是不同的字符常量。
因为 C 语言区分大小写。

以“ \” 开头的字符序列，称为“转义序列”。
“\” 使其后面的字符变为另外的意义。见下表。

字符形式	功能
\n	换行符
\t	横向跳格：跳到下一个输出区（每一输出区为8个字符位置）
\b	退格
\r	回车（回到本行起始字符位置）
\f	走纸换页
\\	反斜杠字符\
\'	单引号(撇号)'
\"	双引号
\ddd	1~3位八进制数所代表的字符。如\101表示'A'(65)
\xhh	1~2位十六进制数所代表的字符。如\x41表示'A'(65)

转义序列主要用来控制打印机和屏幕输出。

例： printf("\n sum is %d\n\n",sum);

比较下面两句的区别：

```
printf(“this is a “test””);
/* 出错 */
printf(“this is a \“test\””);
/* 输出 :this is a “test”*/
```

5.2 字符变量

```
char c1,c2; /* 定义 c1 、 c2 为字符变量 */
c1 = 'a'; c2 = 'b';
```

字符变量在内存中占一字节。

问题：如何用图示的方法表示这些变量的内存空间？

5.3 字符数据在内存中的存储形式及其使用方法

字符在内存中以 ASCII 码存放。

字符	ASCII
'A'	0x41(65)
'B'	0x42(66)
'a'	0x61(97)
'a'	0x62(98)
'0'	0x30(48)
'1'	0x31(49)

```
main()
{
  char c1,c2;
  c1 = ‘A’; c2 = ‘B’;
  printf("%c %c",c1+32,c2+32);
}
```

该例的输出是：　　a b

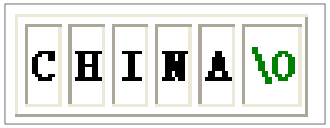
```
main()
{
 char c1,c2;
 c1 = 'a'; c2 = 'b';
 c1 = c1 - 32; c2 = c2 - 32;
 printf("%c %c",c1,c2);
}
```

C 语言允许字符和整数之间进行运算

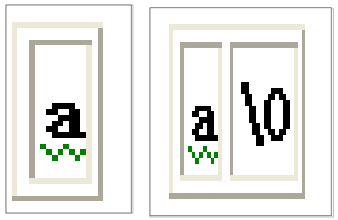
5.4 字符串常量

字符常量： 单引号括起来的一个字符。
字符串常量：双引号括起来的字符序列（ 0~N 个字符）。如：
"How do you do.",
"CHINA", "a", "$123.45"
字符串常量在内存中的存放：
每一个字符均以其 ASCII 码存放，且最后添加一个“空字符”（二进制 00000000 ，记为 NULL 或 \0 。字符‘0’在内存中存 0x30 即 00100000 ）。

例：字符串常量“ CHINA” 存放在内存中的情况是：（6 字节存储器空间，不是 5 字节）



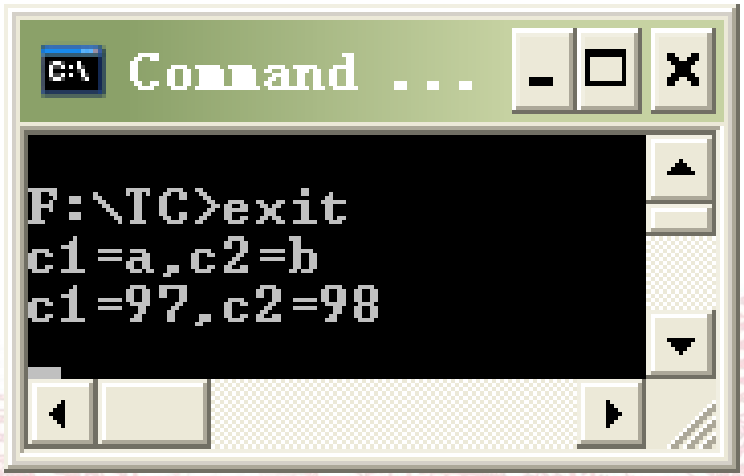
因此，字符 'a' 和字符串 "a" 的区别是：
字符 ‘ a’ ： 1 字节（值为 97 ）
字符串 "a" ： 2 字节（值为 97,0 ）




```
char c;
c = 'a'; /* 字符'a' */
c = "a"; /* 字符串"a" */
```

5.5 字符与整型的相互转换

```
Void main( )
{ char  c1,c2;
  c1='a'; c2='b';
  printf ("c1=%c,c2=%c \n",c1,c2);
  printf ("c1=%d,c2=%d \n",c1,c2);
}
/* 问题：结果如何 */
```



基本数据类型（int,float,double,char,void）

数据类型	类型说明	长度（位）	数据长度
bit	位	1	0， 1
char	字符	8	— 128~127
unsigned char	无符号字符	8	0~255
signed char	有符号字符	8	— 128~127
int	整型	16	— 32768~32767
short int	短整型	16	— 32768~32767
unsigned int	无符号整型	16	0~65535
signed int	有符号整型	16	— 32768~32767
long int	长整型	32	—2147483648~2147483647
unsigned long int	无符号长整型	32	0~4294967295
signed long int	有符号长整形	32	—2147483648~2147483647
float	浮点数(实数)	32	0.175e-38~0.402e38
double	双精度浮点	32	0.175e-38~0.402e38
void	空	0	没任何数据

六、变量赋初值

为什么要给变量赋初值？

因为分配的内存空间可能有一个随机数据。如果不赋初值，默认初始值就是这个随机数据。

1. 定义的同时给变量赋初值

如： int a=3; float b=3.1415;

char c='x';

2. 定义变量时，可以对其中的一部分变量赋初值 . 如： int a=3, b, c, d=8;

3. 把一个值赋给不同变量时，要分别进行。

如不应写为： int a=b=c=3;

而应写为： int a=3,b=3,c=3;

4. 初始化是在程序运行时，执行本函数时赋以初值的。

即不是编译时赋以初值的。如：

int a=3; 相当于： int a; a=3;

int a,b,c=5; 相当于： int a,b,c; c=5;

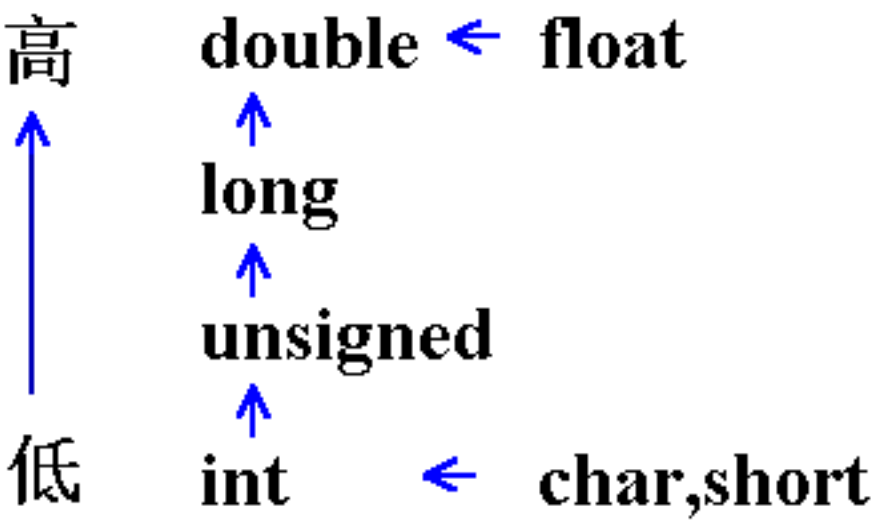
七、各类数值型数据间的混合运算

在 C 语言中，整、实、字符型数据间可以混合运算，如：

10 + 'a' + 1.5 - 8765.1234 * 'b‘

1、不同类型数据间进行运算时，要转换成同一类型

转换过程中，低类型向高类型靠拢，然后进行运算，不同类型之间转换如下图所示，横向左箭头为必转，纵向箭头表示有条件类型转换。


高
低
double
float
long
unsigned
int
char,short

例：

char 数据在运算前必定转换为 int 数据；

int 数据可以转换为 unsigned 数据；

float 数据在运算前必定转换为 double 数据 (即使两个 float 相加) ；

int 数据可以转换为 double 数据 (若另外一个运算数据为 double) ，(直接转换，不需先转换为 unsigned ，long ，再转换为 double) 。

```
int i;
float f;
Double d;
Long e;
10 + 'a' + i*f - d/e
```

上式的运算次序是：

① 进行 10+'a' 运算。先将 'a' 转换为整数 97 ，运算结果为整数 107 。

② 进行 i*f 运算。先将 i 和 f 均转换为 double 类型（实运算过程总是使用 double 类型，仅在把结果存入存储器时，才可能使用 float 类型），运算结果是 double 类型。

③ 整数 107 与 i*f 的积相加。先将整数 107 转换为 double 类型（ 107.0 ），运算结果是 double 类型。

④ 进行 d/e 运算。先将 long e 转换为 double 类型，运算结果是 double 类型。

⑤ 进行③ -④ 运算，结果为 double 类型。

02

PART TWO

算术运算符和算术表达式

C运算符简介

C语言运算符表

运算符按照优先级大小由上向下排列，在同一行的运算符具有相同优先级。第二行是所有的一元运算符。

运算符	解释	结合方式
() [] -> .	括号（函数等），数组，两种结构成员访问	由左向右
! ~ ++ -- + - * & (类型) sizeof	否定，按位否定，增量，减量，正负号， 间接，取地址，类型转换，求大小	由右向左
* / %	乘，除，取模	由左向右
+ -	加，减	由左向右
<< >>	左移，右移	由左向右
< <= >= >	小于，小于等于，大于等于，大于	由左向右
== !=	等于，不等于	由左向右
&	按位与	由左向右
^	按位异或	由左向右
|	按位或	由左向右
&&	逻辑与	由左向右
||	逻辑或	由左向右
? :	条件	由右向左
= += -= *= /= &= ^= |= <<= >>=	各种赋值	由右向左
,	逗号（顺序）	由左向右

算术运算符和算术表达式

1、基本的算术运算符

+ ：加法或正值运算符，如： 2+3 、 +5

- ：减法或负值运算符，如： 8-3 、 -6

* ：乘法运算符， 如： 3*5

/: 除法运算符，则可以是整或实型数据。如 5/3

% ：求余运算符，两侧必是整型数据。如 7%4

注： -5/3 = -1 （余 -2 ） 或 -2(余 +1) 。

2、算术表达式和运算符的优先级与结合性

算术表达式： 如：a*b/c-1.5+’a’

在C语言中，运算符共有15个优先级，其中算术运算符的优先级是：

*、/、%（3级） +、-（4级）

算术运算符的结合方向：从左向右

注意：对于运算符的结合方向，只需记住三种运算符是自右向左的，其余都是自左向右的。这三种运算符是：
单目运算符、
三目运算符 (only one) 、
赋值运算符 (不计算好如何赋值？)

例： a = b + c
由于赋值运算符 = 为右结合，
先执行右边的 b+c ，再赋值给 a 。

强制类型转换运算符

一般形式：（类型名）（表达式）

如：（ double)a　将 a 转换为 double 型

(int)(x+y)　将 x+y 的值转换为整型

(float)(5%3)　将 5%3 的值转换成单精度实型

说明：1.（ int ）(x+y) 与 (int)x+y　意义不同

2. 类型转换后得到一个中间值，而原来变量本身的类型没有发生变化。

3.(int)x 不要写成 int(x);

/*C 语言中不行， c++ 允许 */

```
main( )
{
 float x , i;  x=3.6;  i=(int) (x);
 printf ("x=%f, i=%f\n",x,i);
}
```

运算结果： x=3.600000, i=3.000000

思考：程序运行期间， x,i 的类型和值发生变化没有？

自增、自减运算符

++ ：增 1 运算符，使变量值增 1 。
记忆： ++ 在前，先加； + + 在后，后加
-- ：减 1 运算符，使变量值减 1 。

如：
++i ,--i ：在使用变量 i 之前，先使变量 i 加（减） 1 。
i++,i-- ：在使用变量 i 之后，使变量 i 值加（减） 1 。

i=i+1 的执行代码：

```
lod   ；将存储器中变量调到栈顶
lit 1  ；将常量 1 调到栈顶
add   ；栈顶两个数相加
sto   ；将栈顶数据存到存储器
```

i++ 的执行代码：

```
lod   ；将存储器中变量调到栈顶
inc   ；栈顶数据自加
sto   ；将栈顶数据存到存储器
```

```
main( ) {
 int i,j; i=3;j=++i;
 printf (“i=%d,j=%d\n”,i,j);
}
```

运算结果： i=4,j=4

```
main( ) {
 int i,j; i=3;j=i++;
 printf (“i=%d,j=%d\n”,i,j);
}
```

运算结果： i=4,j=3

说明：

1.++,- - 运算符只能用于变量，而不能用于常量和表达式。

如 a++,b-- 是正确的，而 5++,(x+y)- - 却是错误的。

2.++ ， - - 运算符的结合方向是从右向左。

```
main( ) {
 int i,j;
 i=3;
 i++;
 j=i;
 printf (“i=%d,j=%d\n”,i,j);
}
```

运算结果？

```
main( ) {
 int i; i=3;
 printf (“i1=%d\n”,-i++);
 printf (“i2=%d\n”,i);
}
```

注意：结合性自右向左（单目运算符）

运算结果： i1=-3 i2=4

1) 、无需深究的问题
表达式中的子表达式的求值顺序各编译系统是有差别的。

```
main( ) {
 int  i=3,j=3,k,q;
 k=(i++)+(i++)+(i++);
 q=(++j)+(++j)+(++j);
 printf (“i=%d,j=%d,k=%d,q=%d\n”,i,j,k,q);
}
```

TC 运算结果： i=6,j=6,k=9,q=18

VC 运算结果： i=6,j=6,k=9,q=16

2) 在 C 语言中运算符的确定

在由多个字符组成的表达式中，应尽可能多地从左向右将若干个字符组成一个运算符。

如： i+++j

其结合性是： (i++)+j 不是： i+(++j) 。

3)printf 函数输出实参的顺序

如： int i=3; printf (“%d,%d\n”,i,i++);
有的系统按从左到右的顺序求值，输出结果是： 3,3
而 Turbo C 是按从右到左顺序求值，输出结果是： 4,3

结论：不写别人甚至自己都看不懂的程序，也不写那些不知道系统会怎样运行的程序 .

03

PART THREE

赋值运算符和赋值表达式

赋值运算符

“=” 是赋值号，也是赋值运算符

功能：将赋值号右边表达式的值赋给赋值号左边的变量，赋值号同时含有计算的功能。

如： a=3; b=x*y;

a,b 变量中原来不管存放什么值，执行赋值语句后，新值将取代旧值

类型转换

(1) 实型数据赋给整型变量时，舍去实型数据的小数部分。
如：

int i; i=3.56; 结果 i 的值为 3

(2) 整型数据赋给实型变量时，数值不变，但以浮点形式存放于内存。 如：

float a=23;

先将 23 转换成 23.000000 ，然后送 a 中。

double b=23;

先将 23 转换为 23.00000000000000, 然后送 b 中

(3)double 型数据赋给 float 变量

截取其前面的 7 位有效数字，存放到 float 单元，应注意数值范围不能溢出

如： float f;

```
double d=123.45678e65;
f=d;
```

由于数据溢出， f 将得到错误的值

float 数据赋给 double 变量时，数值不变，有效位扩展到 16 位。

(4) 字符型数据赋给整型变量

(5)int, short, long 型数据赋给 char 变量，则将其低 8 位赋给 char 型变量

(6)long 型数据赋给 int 型变量，将 long 型数据的低 16 位赋给 int 型变量

(7) 将 unsigned int 型数据赋给 long 型变量时：

① 将 unsigned int 型数据送到 long int 型变量的低 16 位，long int 的变量高 16 位补 0

② 若无符号数据赋给相同长度的带符号的变量时，则原样赋给。

要注意数据的有效位占据符号位，若最高位为 1 ，则赋值后新变量的值成负。

(8) 将带符号的数赋给长度相同的无符号变量原样赋给

```
main(  ){
  unsigned  int  a;
  int  b= -1;
  a=b;
  printf (“a=% u,b=%d\n”,a,b);
}
```

运算结果： a=65535,b= -1

复合的赋值运算符

在赋值号前加其它运算符，可以构成复合运算符。

如： a=a+b 等价于 a+=b

x=x*(y+8) 等价于 x*=y+8

x=x%3 等价于 x%=3

为了便于记忆，将赋值号左边移到赋值号右边，赋值号左边再补上变量名。

如： a+= b a+ 复制到 = 右边，等号左边补操作数 a

a=a+b 其中 a 是变量， b 是表达式

x*= y+8 -> x=x*(y+8) 而不是 x=x*y+8

C 语言中，有十个二元运算符： + 、 - 、 * 、 / 、 % 、

《、》、 & 、 ^ 、 | 可与赋值号一起构成复合运算符；

其优点是：简化程序，提高编译效率。

赋值表达式

由赋值运算符将一个变量和一个表达式连接起来的式子。

(1) 一般形式：< 变量 >< 赋值运算符 >< 表达式 >

a=5;

(2) 赋值表达式求解过程：

计算赋值号右边表达式的值，然后赋给左边的变量。

如： a=3+5

在 C 语言中，表达式又可以是赋值表达式，如

a=(b=5);

其中 b=5 是赋值表达式，其值是 5 ，因此 a 的值是 5 ，整

个表达式的值是 5 。

(3) 赋值运算符的结合顺序：从右向左

因此： a=(b=5) 与 a=b=5 是等价的。

赋值表达式的例子：

a=b=c=5

表达式的值为 5 ， a,b,c 的值为 5 。

a=5+(c=6)

表达式的值为 11 ， a 的值为 11 ， c 的值为 6 。

a=(b=4)+(c=6)

表达式的值为 10 ， a 的值为 10,b 为 4,c 为 6 。

a=(b=10)/(c=2)

表达式的值为 5 ， a 的值为 5,b 为 10,c 为 2 。

(4) 赋值表达式也可以包含复合的赋值运算符

```
main(   ) {
 int  a=12, b;
  b=(a+=a-=a*a);
  printf (“a=%d, b=%d\n”,a,b);
}
```

运算结果： a= -264,b= -264

a + = a- = a*a

a=a-a*a

12-12*12

-132

a=a+(-132)

-132+(-132)

-264

逗号运算符和逗号表达式

逗号运算符：，又称“顺序求值运算符”；优先级最低

逗号表达式：用逗号将两个表达式连接起来的式子。
形式：表达式 1, 表达式 2

求解过程：先求表达式 1 的值，再求表达式 2 的值，整个表达式的值是表达式 2 的值。
注：表达式 1 和表达式 2 又可以分别由若干个逗号表达式组成。因此，逗号表达式又可扩展为：
表达式 1，表达式 2, …… 表达式 n

如：

3*5 ， 6 + 8　　表达式 = 14

(a=3*5 ， a*4)　变量 a = 15 表达式 = 60

```
main( ){
 int  a,b;
 b=((a=3*5,a*4),a+5);
 printf (“a=%d,b=%d\n”,a,b);
}
```

运算结果： a=15,b=20

1. 逗号运算符是一个顺序求值运算符

```
main( ) {
  int x,a=1,b=2,c=3;
  x=a,b,c;     printf (“x=%d\n”,x);
}
```

运算结果： x=1

```
main( ) {
 int  a=1,b=2,c=3,x;
  x=(a,b,c);    printf (“x=%d\n”,x);
}
```

结果： x=3

2. 并非所有出现“，”的地方都是逗号运算符，如函数中的参数分隔符

```
main(  ){
int a=1,b=2,c=3;
printf (“%d,%d,%d\n”,a,b,c);
printf (“%d,%d,%d\n”,(a,b,c),b,c);
}
```

运算结果： 1 ， 2 ， 3

3 ， 2 ， 3

04

PART FOUR

总结

SOC 达成度的总体评价

效果

教学反思与小结

效果

感谢各位聆听

Thanks for Listening

现代制造学院 | 香港铸业学院

模具设计与制造专业