C语言程序设计

计算机科学与技术学院 王志军

顺序结构程序设计

计算机的输入输出

- **三种基本结构**：顺序结构、选择结构、循环结构
- **顺序结构**：程序的执行顺序从上到下，逐行执行

变量与常量

- **变量**：在程序运行过程中，其值可以被改变的量
  - int x; scanf ("%d", &x);
- **常量**：在程序运行过程中，其值不能被改变的量
  - printf("%d", 1+2);

常量

- 字面常量：直接写在程序中
  - printf("%d",3 * 2);
- 符号常量：使用define预编译指令定义
  - 写在源程序文件的前端，后面没有’；’
  - 常量的名称一般用大写字符
  - #define MAX 100

例：输入一个整数作为圆的半径，求出圆的周长和面积。

- 例：

```
#include <stdio.h>
int main (){

    int r;
    printf("请输入一个整数：");
    scanf ("%d", &r);
    printf("周长：%f\n", 2*3.14*r);
    printf("面积：%f", 3.14*r*r);

    return(0);
  }
```

- 说明
  - 2*3.14*r 中的 * 表示 乘
  - %f：输出一个小数
  - 3.14->3.14159？有500个？

例：输入一个整数作为圆的半径，求出圆的周长和面积。

- 例：

```
#include <stdio.h>
#define PI 3.14159
int main (){

    int r;
    printf("请输入一个整数: ");
    scanf ("%d", &r);
    printf("周长: %f\n", 2*PI*r);

    return(0);
  }
```

- 说明
  - #define PI 3.14159 定义一个常量
  - 一次定义，多次使用，修改初值方便
  - 符号常量必须"从一而终"
  - 在程序中，不能试图更改常量的值
    - PI=3.14 错误

变量

- 先定义，后使用
- 变量须要指定数据类型
- 变量定义的一般格式
  - 数据类型 变量名1[，变量2，…，变量名n];
  - 变量名：合法标识符，通常使用小写字母
  - 数据类型： 决定存放何种数据、如何存
- 同一函数体或分程序中，不能定义两个同名的变量

- 例

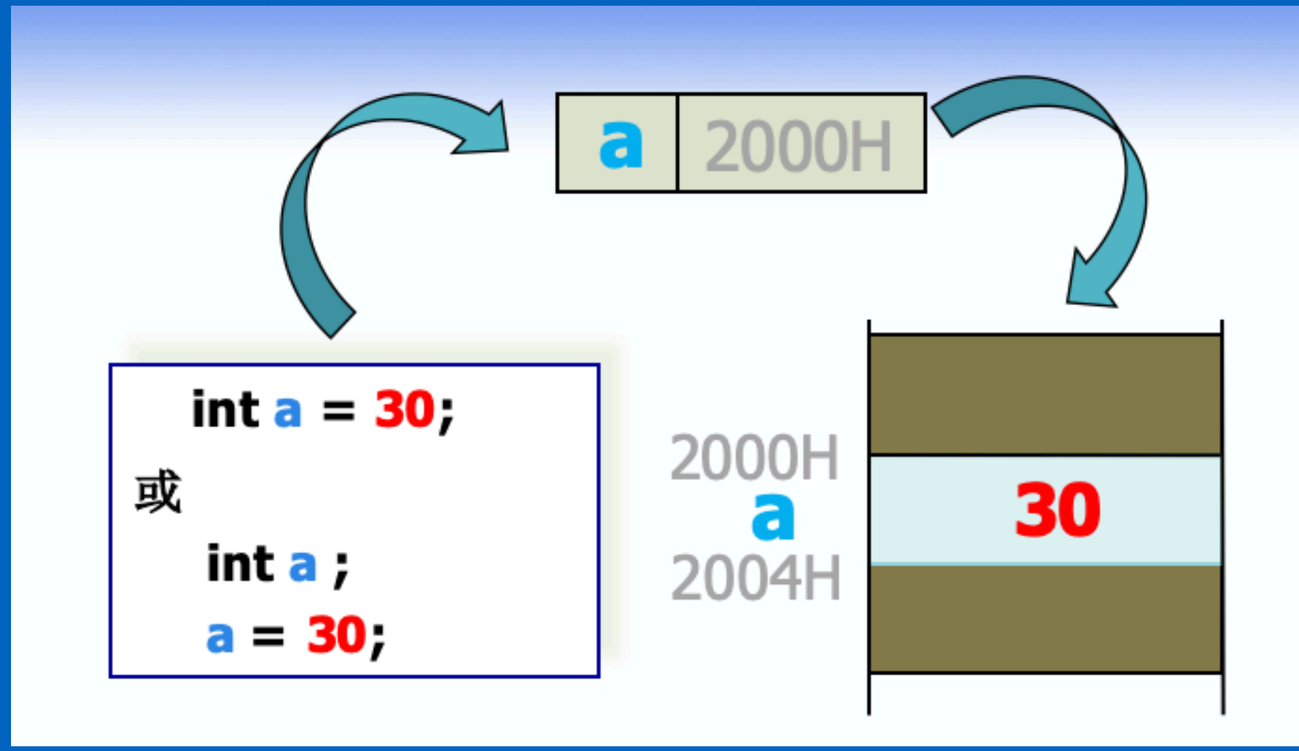
```
int a;
a=30;
printf("a=%d\n", a);
```

- 例

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

变量的本质

- 变量代表内存中的一个存储单元
- 变量名称实际上是存储单元地址的别名
- 变量的值是存储单元中存放的数据
- 从变量中取值，实际上是通过变量名找到相应的内存地址，从该存储单元中读取数据。



变量命名：标识符

- 标识符只能由字母、数字和下划线组成
- 第一个字符必须为字母或下划线
  - int 学号？
  - `sum, _total, month, Student_name, lotus_1_2_3, BASIC, li_ling`
  - M.D.John, ￥123, 3D64, a>b
- 区分 大写字母和小写字母
  - int sum,Sum;
- 命名时应"见名知意"，即选有含意的英文单词 （或其缩写）作标识符,增强可读性
  - int s;
  - int sum;
  - int sumScore;

变量初始化

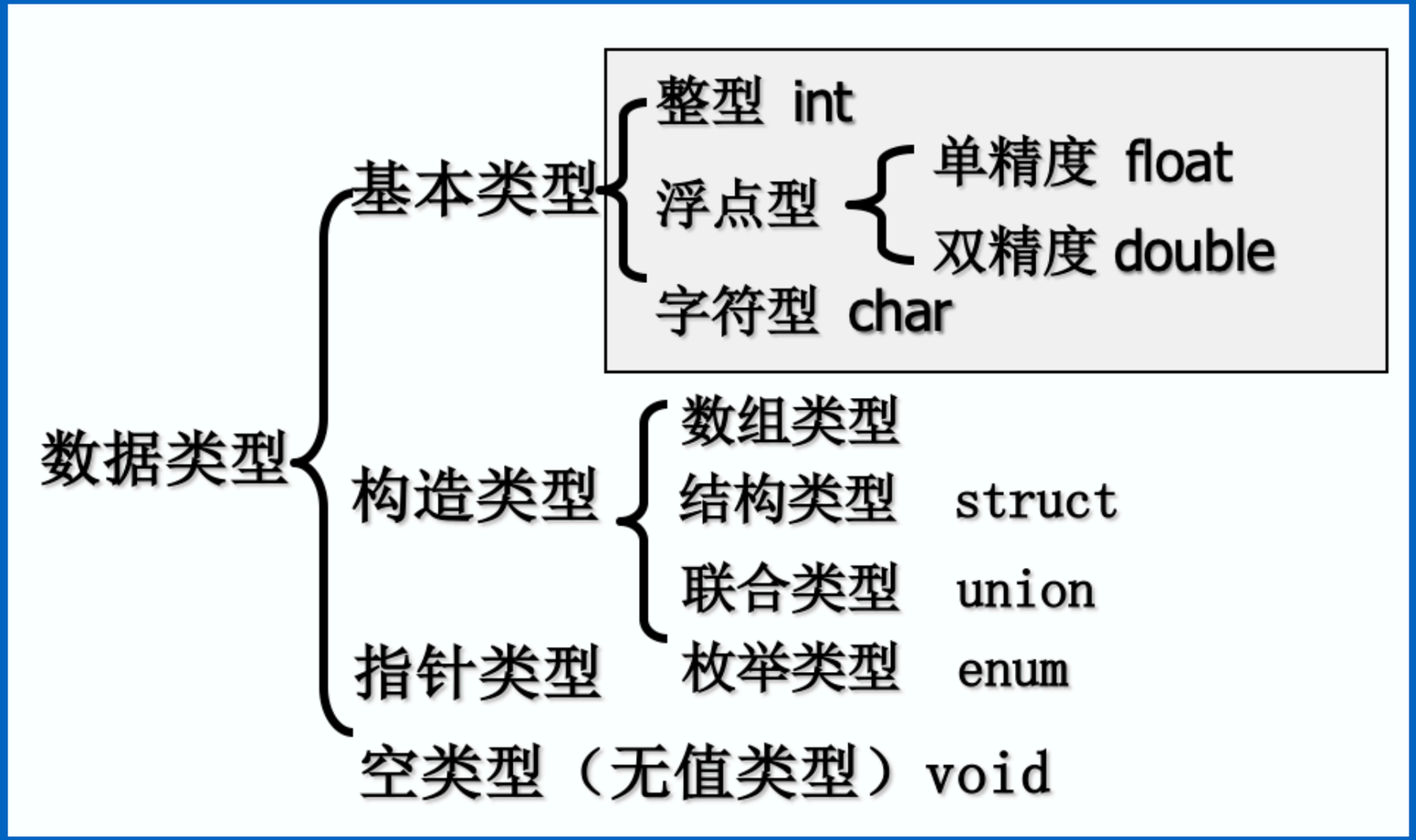
```c
#include <stdio.h>
int main (){

    int a, b,c;
    a=30;
    b=2;
    c=a*b;
    printf("a*b=%d\n", c);

    return(0);
}
```

```c
#include <stdio.h>
int main (){

    int a = 30, b=2,  c;
    c=a*b;
    printf("a*b=%d\n", c);

    return(0);
}
```

变量初始化

- 可以对被定义的变量的一部分赋初值：
  - int a,b,c=3；
  - 定义 a 、 b 、 c 为整型变量, 但只对 c 初始化， c的初值为3, b 和 c 必须在后续程序中赋值
- 对几个变量赋以同一个初值：
  - int a=3,b=3,c=3;
  - 定义 a , b , c 为整型变量， 初值都是 3
  - 不能写成：int a=b=c=3;

C中的数据类型



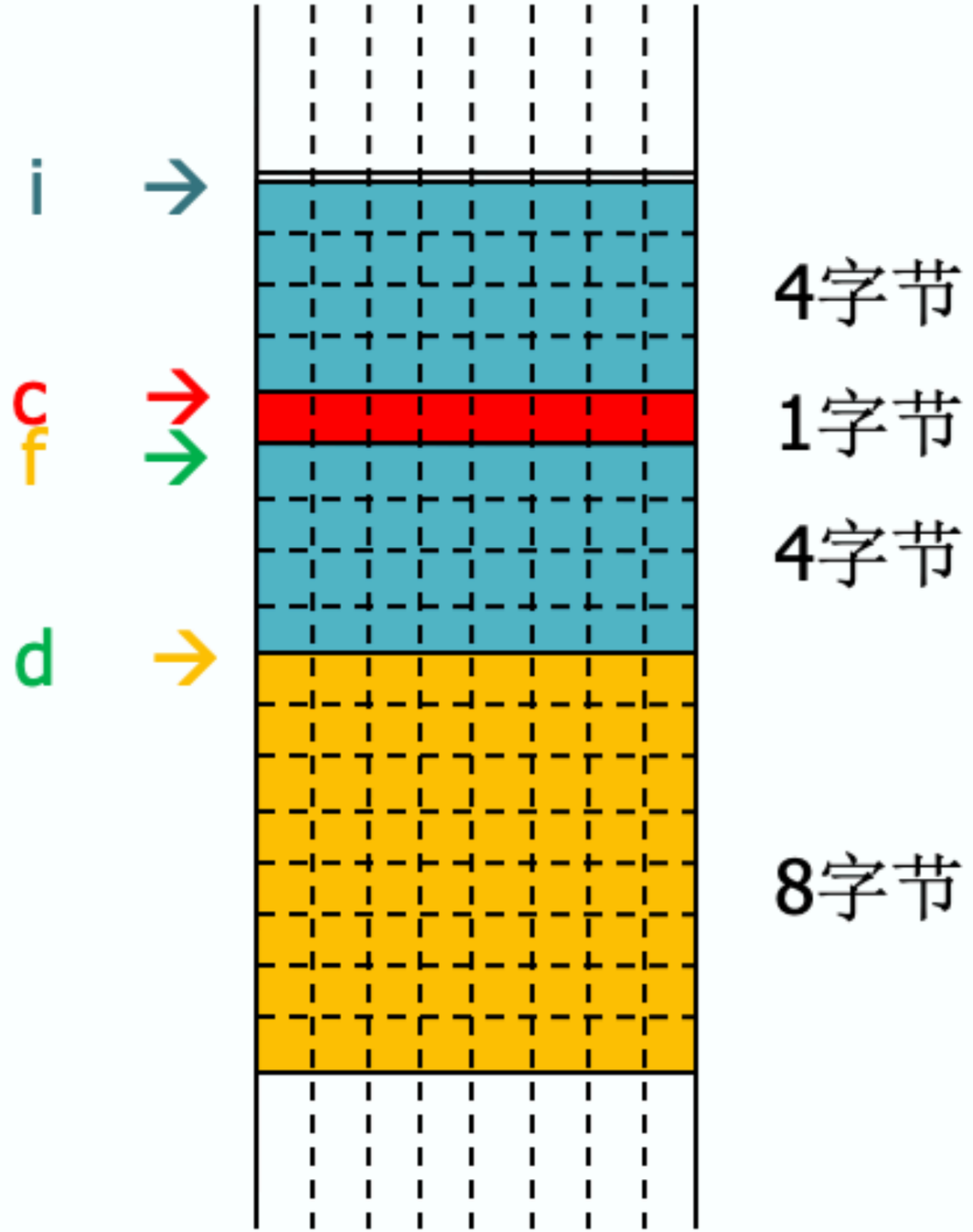
数据类型举例

- 整数 100，125，-100，0
- 浮点型 3.14 ， 0.125，-3.789
- 字符型 'a', 'b'，'2'
- 字符串型 "a" , "ab" ， "1232"

数据类型的存储长度

- 整型(int)：4字节，
- 字符型(char)：1字节
- 单精度 (float)：4字节
- 双精度 (double)：8字节

```
int i = 65;
char c='A’;
float f=3.14;
double d=3.14;
```



sizeof运算符

```c
#include <stdio.h>
int main (){

  int i;
  i=sizeof(char);
  printf("%d\n",i);

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

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

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

  return(0);
}
```

整型（int）

- 整型常量有三种表示方法：
  - 十进制：65
  - 八进制：加前缀0 0123、0701
  - 十六进制：加前缀0x（或0X）0x41, 0x5abc

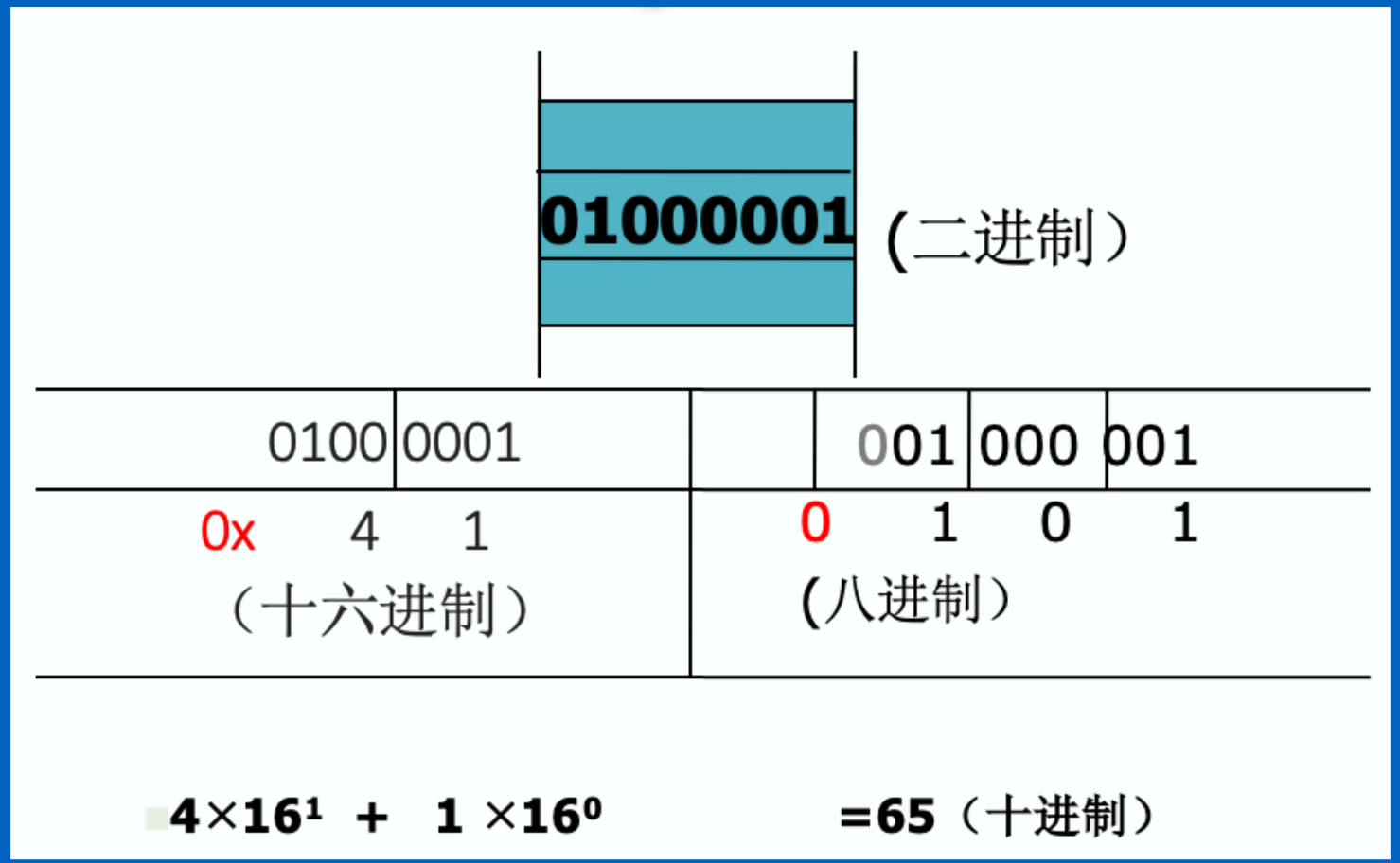
- 例

```
#include <stdio.h>
int main (){

  int a;
  scanf ("%d",&a);
  printf("%d, %d , %d",a+10, a+010, a+0x10);

  return(0);
}
```

C中的八进制和十六进制整数

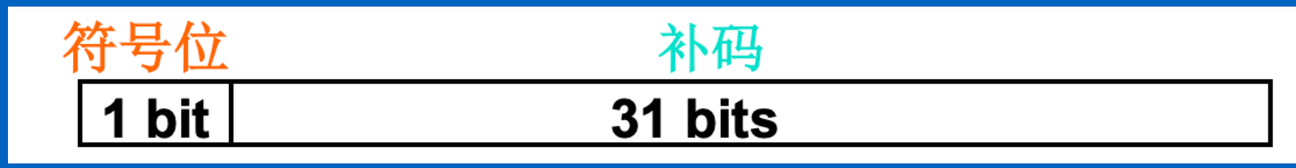
- 八进制：以 0 起始
  - 010 和 10不同
- 十六进制：以 0x 或 0X 起始
  - A~F或a~f代表10~15
  - 0x11,0x05,0xFA,0xFF
  - 存储地址
  - 按位控制

八进制和十六进制都是二进制的简记形式



整型（int)

- int 表示范围
  - -2^{31}～$2^{31}-1$
  - -2147483637～+2147483647
- 为什么?



浮点型(float, double)

- 即实型
  - 小数形式
  - 指数形式
- 小数形式
  - 3.14，45.67，0.893，*.409*

- 例

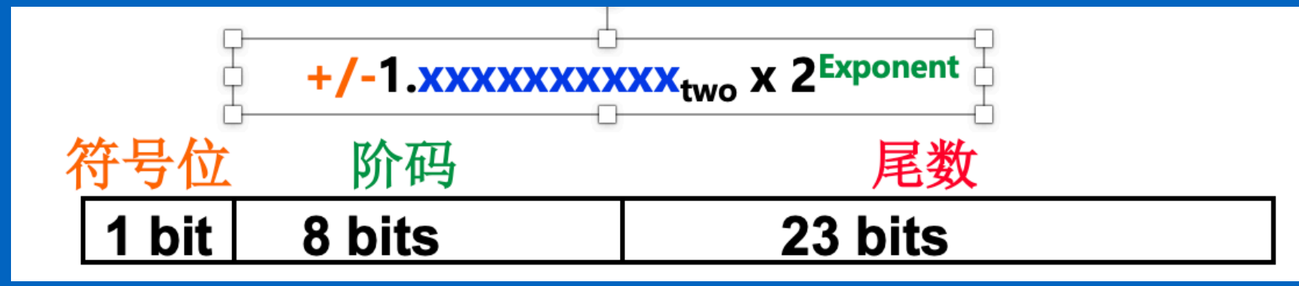
```
#include <stdio.h>
int main()
{
  float f=3.14;
  double d=3.14;
  printf("%f, %f",f, d);

  scanf("%f, %lf",&f, &d);
  printf("%e, %e",f, d);

  return(0);
}
```

例：输入一个整数作为圆的半径，求出圆的周长和面积。

```c
#include <stdio.h>
#define PI 3.14159
int main (){

    float l,s,r ;
    printf("输入半径: ");
    scanf("%f",&r);
    l=2*PI*r;
    s=PI*r*r;
    printf("l=%.2f,s=%.2f\n",l,s);

    return(0);
}
```

浮点型(float, double)

- float
  - 32位(bits)
  - 表示范围：10^{37}～10^{38}
  - 有效数字：7位
- double
  - 64位(bits)
  - 表示范围：-10^{307}～10^{308}
  - 有效数字：15位
- IEEE 754：
  - 1985年完成浮点数标准的制定
  - UC Berkeley 数学教授 William Kahan，1989年图灵奖



浮点数的指数形式

- 科学计数法
  - 123.456 -> 1.23456e3
  - 12e4，1.2e12，0.9e-3，.3e4
- 输出格式符：%e
- 字母e(或E)之前必须有数字，且e后面须为整数
  - 以下不合法：E3，2.1e3.5，.e3，e

字符型(char)

- 单个字符的三种表示方式
  - 直接表示方式：如 'A'， '9'
- 转义字符表示方式
  - '\n'：回车并换行
  - '\'': 单撇号本身
  - ' \ \ ' : 斜杠本身
  - '\t' 横向跳格
  - '\r' 回车
- ASCII码表示方式
  - '\ddd' ddd表示1到3位八进制数字, 如 '\101'
  - '\xhh' hh表示1到2位十六进制数字，如 '\x41'

ASCII 码		字符	ASCII 码		字符	ASCII 码		字符	ASCII 码		字符
十进制	十六进制		十进制	十六进制		十进制	十六进制		十进制	十六进制	
032	20		056	38	8	080	50	P	104	68	h
033	21	!	057	39	9	081	51	Q	105	69	i
034	22	"	058	3A	:	082	52	R	106	6A	j
035	23	#	059	3B	;	083	53	S	107	6B	k
036	24	$	060	3C	<	084	54	T	108	6C	l
037	25	%	061	3D	=	085	55	U	109	6D	m
038	26	&	062	3E	>	086	56	V	110	6E	n
039	27	'	063	3F	?	087	57	W	111	6F	o
040	28	(	064	40	@	088	58	X	112	70	p
041	29	)	065	41	A	089	59	Y	113	71	q
042	2A	*	066	42	B	090	5A	Z	114	72	r
043	2B	+	067	43	C	091	5B	[	115	73	s
044	2C	,	068	44	D	092	5C	\	116	74	t
045	2D	-	069	45	E	093	5D	]	117	75	u
046	2E	.	070	46	F	094	5E	^	118	76	v
047	2F	/	071	47	G	095	5F	_	119	77	w
048	30	0	072	48	H	096	60	`	120	78	x
049	31	1	073	49	I	097	61	a	121	79	y
050	32	2	074	4A	J	098	62	b	122	7A	z
051	33	3	075	4B	K	099	63	c	123	7B	{
052	34	4	076	4C	L	100	64	d	124	7C	|
053	35	5	077	4D	M	101	65	e	125	7D	}
054	36	6	078	4E	N	102	66	f	126	7E	~
055	37	7	079	4F	O	103	67	g	127	7F	⌂

字符的输出

```
#include <stdio.h>
int main()
{  char c1, c2, c3 ;
   c1 = 'A’;
   c2 = '\101’;
   c3 = '\x41’;

   printf("%c,%c,%c", c1, c2, c3);//输出 A,A,A
   printf("%d,%o,%x",c1, c2, c3);//输出 65,101,41

   int i=65;
   printf ("%d,%c",i,i);/*不规范，但能够输出 65,A*/

   // 65、'A’、'\101’、'\x41在内存中存储的都是 0100 0001（二进制）

   return 0;
}
```

字符的+和-：移位

```
#include <stdio.h>
int main()
{
    printf("%c,%c",’A’,’A’+1);
    printf("%c,%c",’A’,’A’+32);
    printf("%d",'f’-’d’);
    return 0;
}
```


ASCII 码		字符	ASCII 码		字符	ASCII 码		字符	ASCII 码		字符
十进制	十六进制		十进制	十六进制		十进制	十六进制		十进制	十六进制	
032	20		056	38	8	080	50	P	104	68	h
033	21	!	057	39	9	081	51	Q	105	69	i
034	22	"	058	3A	:	082	52	R	106	6A	j
035	23	#	059	3B	;	083	53	S	107	6B	k
036	24	$	060	3C	<	084	54	T	108	6C	l
037	25	%	061	3D	=	085	55	U	109	6D	m
038	26	&	062	3E	>	086	56	V	110	6E	n
039	27	'	063	3F	?	087	57	W	111	6F	o
040	28	(	064	40	@	088	58	X	112	70	p
041	29	)	065	41	A	089	59	Y	113	71	q
042	2A	*	066	42	B	090	5A	Z	114	72	r
043	2B	+	067	43	C	091	5B	[	115	73	s
044	2C	,	068	44	D	092	5C	\	116	74	t
045	2D	-	069	45	E	093	5D	]	117	75	u
046	2E	.	070	46	F	094	5E	^	118	76	v
047	2F	/	071	47	G	095	5F	_	119	77	w
048	30	0	072	48	H	096	60	`	120	78	x
049	31	1	073	49	I	097	61	a	121	79	y
050	32	2	074	4A	J	098	62	b	122	7A	z
051	33	3	075	4B	K	099	63	c	123	7B	{
052	34	4	076	4C	L	100	64	d	124	7C	|
053	35	5	077	4D	M	101	65	e	125	7D	}
054	36	6	078	4E	N	102	66	f	126	7E	~
055	37	7	079	4F	O	103	67	g	127	7F	⌂

字符串

- 一对双引号括起来的字符序列
  - "program" 、"This is a string"
- 每个字符串都有一个结束符'\0’。
- 在C中，字符串不是一种基本数据类型，也没有字符串变量
- 输出方式：
  - printf("Hello, world!\n");
  - printf("%s","Hello, world!\n");

- 字符与字符串的区别
- 'a’和"a"不同

```
char c;
c='a';     //可以
C="a";     //不可以 'a''\0'
```

运算符：分为单目、双目、三目运算

- 算术运算符：+ - * / % ++ --
- 关系运算符：< <= == > >= !=
- 逻辑运算符：！ && ||
- 位运算符：<< >> ~ | ^ &
- 赋值运算符：= += -= *= /= %=
- 条件运算符：? :
- 逗号运算符：，
- 指针运算符：* &
- 求字节数： sizeof
- 强制类型转换：（类型）
- 分量运算符：. ->
- 下标运算符：[]
- 其它：( ) -

表达式

- 变量、常量和函数以及运算符按一定规则组合起来的算式
- 表达示求值
  - 其结果有一个确定的值
  - 其结果有一个确定的类型
- 合法与不合法
  - 1+2
  - a+b //若a与b已知的变量
  - 8%3.5

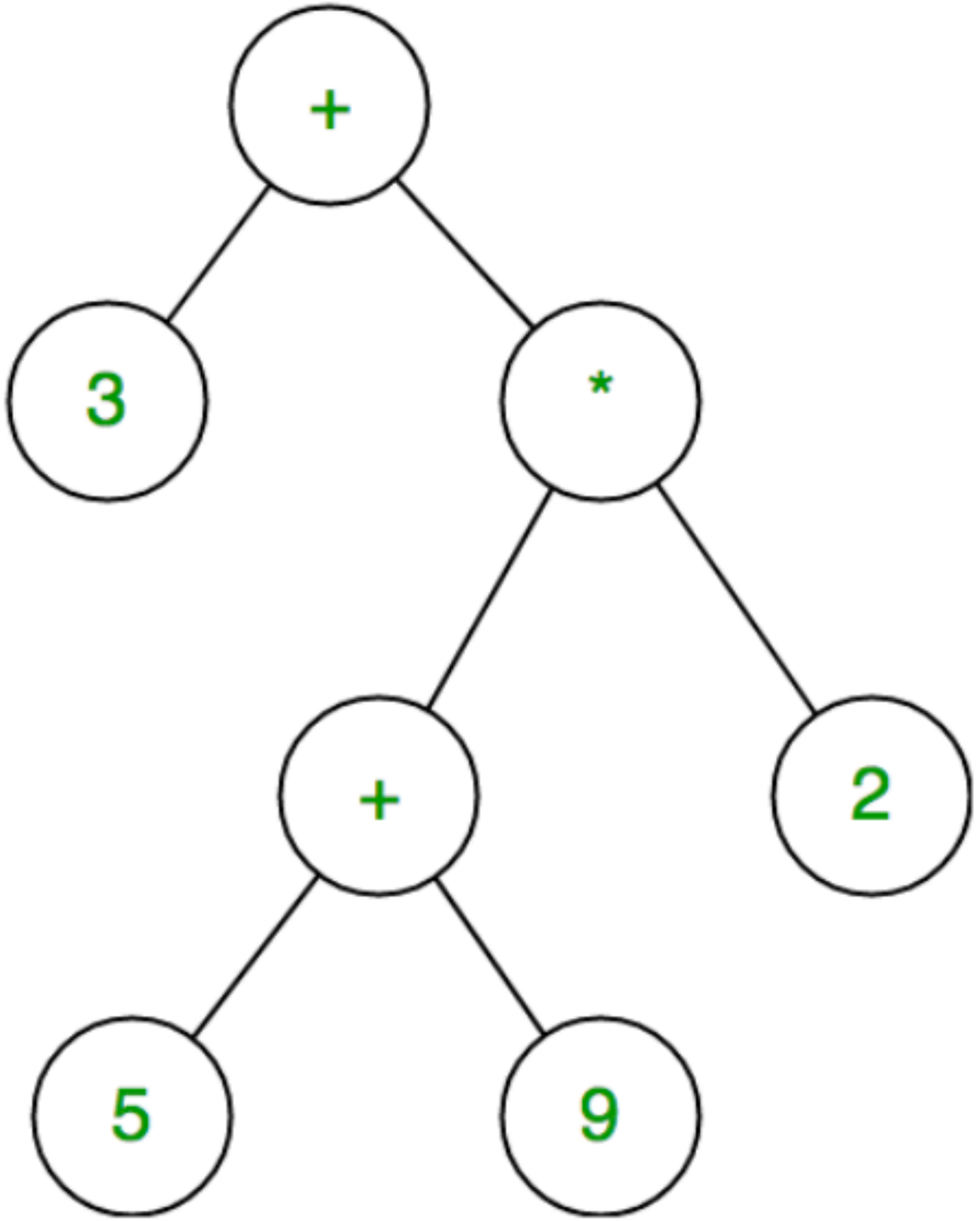
运算符的优先级与结合性

运 算 符	结合性
() [] -> .	从左至右
! ~ ++ -- + - * & (*type*) sizeof	从右至左
* / %	从左至右
+ -	从左至右
<< >>	从左至右
< <= > >=	从左至右
== !=	从左至右
&	从左至右
^	从左至右
|	从左至右
&&	从左至右
||	从左至右
?:	从右至左
= += -= *= /= %= &=	从右至左
^= |= <<= >>=	从左至右
,	

注：一元运算符+、-、&与*比相应的二元运算符+、-、&与*的优先级高。

运算符的优先级

- 优先级列表一般规律
  - 单目运算：优先级最高
  - 算术运算 优先级 高于 赋值运算
  - 算术运算中，* / 优先级 高于 + -
- 括号改变优先级
  - 1+2*3/4-5
  - (1+2)*3/(4-5)
  - x=1+2 //x=(1+2)
- 表达示求值
  - 3 + ((5+9)*2)



算术运算符

- 算术运算: 从左向右结合

  printf("%d",1+2);

- 运算符：
  - ＋：加法运算符，或正值运算符。如: 3 ＋ 5 、＋ 3
  - －：减法运算符，或负值运算符。如: 5 － 2 、－ 3
  - *：乘法运算符，如: 3 * 5
  - /：除法运算符，如: 5 / 3 。注意：整数相除，结果为整数
  - %：模运算符，或称求余运算符，%要求两侧均应为整型数据，如: 7 % 4 的值为 3

算术运算符

```
#include <stdio.h>
int main()
{    int  a;
     a=_________________;
     printf("%d", a);
     return(0);
}
```

- 若：
  - 19%5
  - 12%4
  - 3%9
  - 7/3
  - 5%-3
  - -5%3
  - 'd’-2
  - 'f’-’d’

自增 ++、自减 --

- 作用：使变量值加1或减1
- 种类：
  - 前置 ++i, --i (先执行i+1或i-1，再使用i值)
  - 后置 i++,i-- (先使用i值,再执行i+1或i-1)
- 只能用于整型变量、字符变量
- 不能用于常量或表达式
  - 不可以：5++；(a+b)++;

自增 ++、自减 -- 的特点

```c
#include <stdio.h>
int main()
{
    int  a=0;
    a++;  // 与 a = a+1 采用不同的指令
    printf("%d", a);
    return 0;
}
```

前置的作用

```c
#include <stdio.h>
int main()
{
    int  a=0, b=1;

    b=++a;

    printf("%d, %d", a,b);

    return 0;
}
```

```c
#include <stdio.h>
int main()
{
    int  a=0, b=1;

    a++;
    b=a;

    printf("%d, %d", a,b);

    return 0;
}
```

后置的作用

```c
#include <stdio.h>
int main()
{
    int  a=0, b=1;

    b=a++;

    printf("%d, %d", a,b);

    return 0;
}
```

```c
#include <stdio.h>
int main()
{
    int  a=0, b=1;

    b=a;
    a++;

    printf("%d, %d", a,b);

    return 0;
}
```

分析技巧：折分成多个语句

```c
#include <stdio.h>
int main()
{
    int  k,j,m,n;
    k=1; j=1;

    m=(k++)+(j++);

    n=(++k)+(++j);
    printf("%d,%d,%d,%d",k,j,m,n);
    return 0;
}
```

```c
#include <stdio.h>
int main()
{
    int  k,j,m,n;
    k=1; j=1;

    m=k+j;
    k++;
    j++;

    n=(++k)+(++j);
    printf("%d,%d,%d,%d",k,j,m,n);
    return 0;
}
```

分析技巧：折分成多个语句

```
#include <stdio.h>
int main()
{
    char  c='g’,d;
    int m=2;

    d=(m++)+(--c);

    printf("%d, %c, %c\n",m,c,d);
    return 0;
}
```

```
#include <stdio.h>
int main()
{
    char  c='g’,d;
    int m=2;

    --c;
    d=m+c;
    m++;

    printf("%d, %c, %c\n",m,c,d);
    return 0;
}
```

赋值运算

- 从右向左结合
- 赋值运算符: =

```
int x=0;
x=1+2;
printf("%d",x);
//printf("%d",x=1+2);
```

- 复合赋值运算符: +=， -=, *=， /=， %=

赋值运算：从右向左结合

```c
#include <stdio.h>
int main()
{
    int  a=0, b=0;
    a=b=2;
    //int  a=b=2;  错误
    printf("%d, %d", a,b);

    return 0;
}
```

```c
#include <stdio.h>
int main()
{
    int  a=0, b=0;
    b=2;
    a=b;
    printf("%d, %d", a,b);

    return 0;
}
```

复合赋值运算：从右向左结合

```c
#include <stdio.h>
int main()
{
    int  a=0, b=1;

    a+=b;

    printf("%d, %d", a,b);

    return 0;
}
```

```c
#include <stdio.h>
int main()
{
    int  a=0, b=1;

    a=a+b;

    printf("%d, %d", a,b);

    return 0;
}
```

复合赋值运算：从右向左结合

```c
#include <stdio.h>
int main()
{
    int  a=0, b=1;

    b*=a+4;

    printf("%d, %d", a,b);

    return 0;
}
```

```c
#include <stdio.h>
int main()
{
    int  a=0, b=1;

    b=b*(a+4);

    printf("%d, %d", a,b);

    return 0;
}
```

复合赋值运算：从右向左结合

```c
#include <stdio.h>
int main()
{
    int a=4, b=2, c=1;

    printf("%d", c*=a+=b*=a/=3);

    return 0;
}
```

```c
#include <stdio.h>
int main()
{
    int a=4, b=2, c=1;

    a=a/3;
    b=b*a;
    a=a+b;
    c=c*a;

    printf("%d", c);

    return 0;
}
```

数据长度运算符 sizeof

- 求数据类型/变量在内存中所占的字节数
- 用法：sizeof(<类型说明符>/<变量名>)
- 例：

```
float a=20.0;
printf("%d",sizeof(a));
```

数据类型转换

- 不同数据类型的运算量，在运算过程中可以进行转换
- 自动类型转换：
  - 低精度类型转换为高精度类型
- 强制转换
  - 使用数据类型转换运算符 ( )

自动类型转换

- - 运算或赋值时可能发生
- 一般地，低精度类型转换为高精度类型
  - char ->int
  - int ->浮点数
  - float -> double
- 例：5%7+3*1.0 -> 8.0
- 例：

```
int i; float f;
i=3.14;
f=i;
printf("%d,%f", i,f);
->3,3.0
```

自动类型转换示例

```
#include <stdio.h>
int main()
{
    __________  x;
    x=  _______表达式__________;
    printf("%_____", x);
    return 0;
}
```

- 表达式处代码若为：
  - 3+4/5 -> 3
  - 5.5+6/4 -> 6.5
  - 12+5/4.0 ->13.25
  - 5%7+3*1.0 ->8.0
  - 20-5*9%11 -> 19
  - 2+3*(6+7/3%9) -> 26
  - 6.0/(3/2) -> 6.0

强制类型转换

- 可以利用强制类型转换运算符 ()将一个表达式转换成所需类型
- 一般形式:（类型名)(表达式）
- 例:
  - (double) a ; //将 a 转换成double类型
  - (int)(x+y); // 将x+y的值转换成整型
  - (float)(5%3); //将5%3的值转换成float型

强制类型转换

- 在转换时产生一个临时变量，存放转换好的数据
- 原来数据的类型未发生改变
- 例：

```
#include <stdio.h>
int main()
{
    float  x;
    int  i, j,k;
    x=3.6;
    i=3.6;
    j=(int)x;
    k=(int)(i+x);
    printf("%f, %d,%d, %d",x,i,j,k);
    //输出 3.600000,3,3,6
    return 0;
}
```