Ch-02

数据类型、运算符与表达式

主要内容：

- C++的”单词”及分类
- 数据
- 运算符与表达式

2.1 C++的“单词”（词法记号）及分类

- 标识符（可理解为“对象的名字”）
  - 关键字（又称为：保留字）
- 运算符
  - 单目、双目、三目
- 分隔符
  - “()” “{}” “,” “:” “;” “空白”
- 数据
  - 数字、字符、字符串、布尔量
- 其它符号
  - “/*”和“*/”是程序注释的定界符
  - “#”“< >”等

1、标识符

- 定义：用来标识变量、常量、函数等的字符序列，也就是一个“对象的名字”。
- 组成：
  - 只能由字母、数字、下划线组成，且第一个字符不能是数字；
  - 大小写敏感；
  - 不能使用关键字。
- 长度任意，但受编译器限制。

2、关键字

又称保留字，是一类被系统占用且具有特定含义的特殊标识符。

*** 不能用作一般标识符，即不允许用作变量名或函数名等。**

表1　C++的关键字

auto	bool	break	case	catch	char	class
const	const_cast	continue	default	delete	do	double
dynamic_cast	else	enum	explicit	extern	false	float
for	friend	goto	if	inline	int	long
mutable	new	operator	private	protected	public	register
reinterpret_cast		return	short	signed	sizeof	static
static_cast	struct	switch	template	this	throw	true
try	typedef	typeid	typename	union	unsigned	virtual
void	volatile	while				

3、空白

- 种类：空格、制表符、换行符、注释；
- 功能：指示“单词”的开始和结束位置。

说明：注释有两种形式：“/*...*/” 和“//”

2.2 C++中的数据类型及类型说明符

数据类型有什么用？

1、决定数据占用的存储空间大小；

——由操作系统和编译器的实现所决定

——可用sizeof运算验证

2、决定数据的取值范围；

3、决定数据上可行的运算有哪些。

表2-1　C++的基本数据类型

类型标识符	字宽（字节）	取值范围
bool		false, true
char	1	-128 ~ 127
signed char	1	-128 ~ 127
unsigned char	1	0 ~ 255
short [int]	2	-32768 ~ 32767
signed short [int]	2	-32768 ~ 32767
unsigned short [int]	2	0 ~ 65535
int	4	-2147483648 ~ 2147483647
signed [int]	4	-2147483648 ~ 2147483647
unsigned [int]	4	0 ~ 4294967295
long [int]	4	-2147483648 ~ 2147483647
signed long [int]	4	-2147483648 ~ 2147483647
unsigned long [int]	4	0 ~ 4294967295
float	4	$3.4\times10^{-38} \sim 3.4\times10^{38}$
double	8	$1.7\times10^{-308} \sim 1.7\times10^{308}$
long double	10	$3.4\times10^{-4932} \sim 3.4\times10^{4932}$

2.2.1 整型

□ 类型说明符

表1 VC++中的整型表示

类型		类型说明符	Bytes	可表示的范围
有符号	基本型	[signed] int	4	$-2^{31} \sim +2^{31}-1$
	短整型	[signed] short [int]	2	$-2^{15}-1 \sim +2^{15}-1$
	长整型	[signed] long [int]	4	$-2^{31} \sim +2^{31}-1$
无符号	基本型	unsigned [int]	4	$0 \sim 2^{32}-1$
	短整型	unsigned short	2	$0 \sim 2^{16}-1$
	长整型	unsigned long	4	$0 \sim 2^{32}-1$

注：不同的编译系统为整型分配的字节数是不相同的。

2.2.2 字符型

□ 类型说明符

char

□ 存放形式和取值范围

- **一个字符一般用8位（1个字节）来存放；**
- **存放的是该字符的ASCII码值（即整数），因此可把字符型看作一种特殊的整型。**

‘a’

0 1 1 0 0 0 0 1

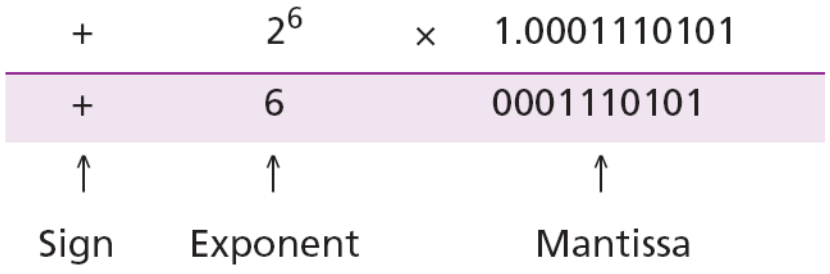
97

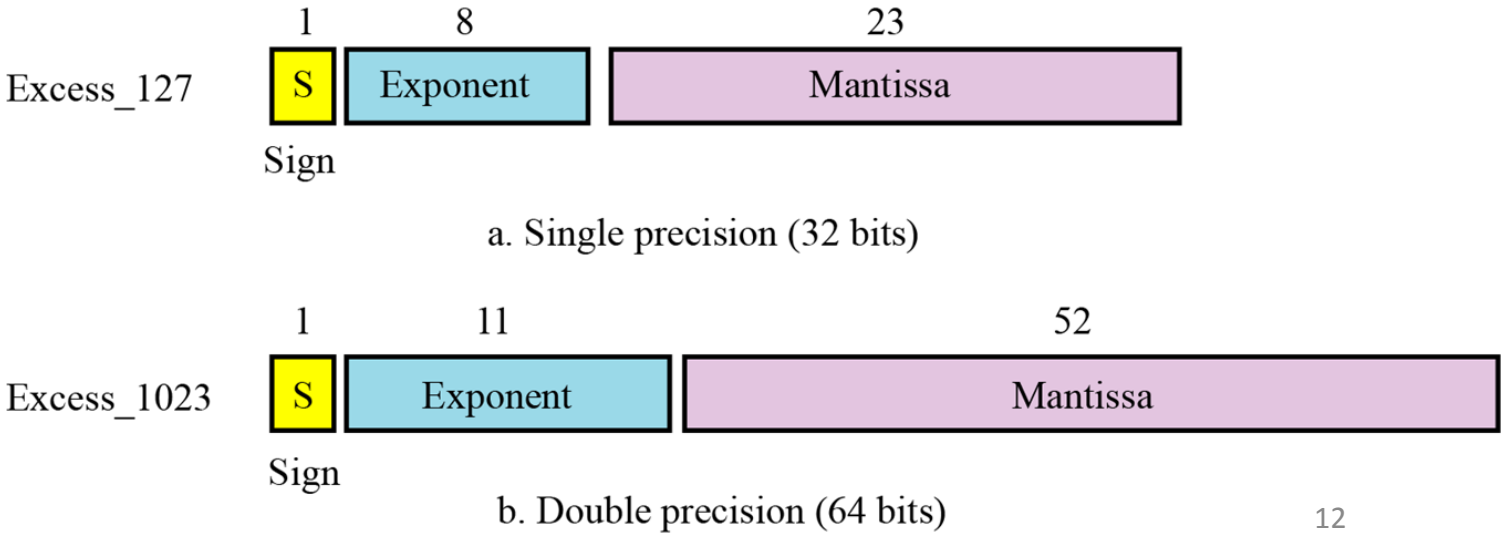
2.2.3 浮点型

□ 类型说明符

类型	类型说明符	Bytes (VC++)	绝对值的范围	有效数字
单精度浮点型	float	4	0和1.2×10^{-38} ~ 3.4×10^{38}	6位
双精度浮点型	double	8	0和2.3×10^{-308} ~ 1.7×10^{308}	15位

□ 浮点型数据在内存中的存放形式（IEEE浮点数存储标准）





2.2.4 空类型 void

- void类型只用于以下三种情况：
  - 函数返回值的限定
  - 函数参数的限定
  - 定义无类型指针变量
- 举例：

```
void printStar(void);  //限定printStar函数没有参数和返回值
void *p;  /* 指针变量p的基类型是任意的，
             即该指针可以指向任意类型的目标变量 */
```

2.3 数据

两种基本表现形式：常量 和 变量。

口常量

- **在程序运行过程中，其值不能被改变的量**

口变量

- **在程序运行过程中，其值可以被改变的量**

2.3.1 常量

通过常量的字面形式和值识别其类型

2.3.1.1 直接常量（字面值）

□ 整型常量

- 默认是有符号int整型，可通过数值大小识别其具体类型
- 表示方法
  - 十进制：有效字符0~9。如100、-8、0、+123
  - 八进制：以0开头，有效字符0~7。如010、024、0100、073
  - 十六进制：以0x（或0X）开头，有效字符0~9、A~F（或a~f）。

    如0x38、0x10或0X10、0XFF、0x0a

说明：

1. **长整型用L（或l）做后缀表示。例如：32765L，793l；**
2. **无符号型用U（或u）做后缀表示。例如：4352U，3100u；**
3. **unsigned long型用后缀U（或u）和L（或l）一起表示，L与U的先后顺序无关。例如：**

   49321ul，37825LU，41152Lu；
4. **无后缀时，整型常量类型按如下顺序确定：**

   int → long → unsigned long

口 浮点型常量

- **默认double类型**
  - ➢ 数值后面加F或f表示float类型，加L或l表示long double类型。
- 表示方式
  - ➢ 只用十进制表示
  - ➢ 小数形式： 0.02 .02 3.0 3. -7.4
  - ➢ 指数形式/浮点形式：12.3e3 .034e-5（*e前必须有数字，e后指数必须为整数*）
    - **规范化的指数形式**

      在字母e（或E）之前的小数部分中，小数点左边有且只有一位非零的数字。

      例如: 123.456，则1.23456e2称为“规范化的指数形式”。

逻辑常量（布尔常量）

- 只有两个：false 和 true
- 编译器处理逻辑常量时：false表示为0，true表示为1

字符常量

- 用“单引号” 括起来的“单个字符” 或“转义字符序列” ，例如：‘a’ 、 '+'、 '3'
- 在计算机中用对应的ASCII码值表示
- 表示方法：
  - 普通字符表示法：可以表示可打印字符 ，如：'a'、'A' 、'+' 、'3'、' ' 等
  - 转义字符序列表示法：可以表示所有字符，尤其是不可打印字符

    以字符'\'开头的字符序列，具体见下表：

字符	含义	ASCII码
\n	换行，将当前位置移到下一行开头	10
\t	水平制表，将当前位置移跳到下一个Tab位置	9
\b	退格，将当前位置移到前一列	8
\r	回车，将当前位置移到本行开头	13
\f	换页，将当前位置移到下页开头	12
\a	发出铃声	7
\0	空字符	0
\\	表示一个反斜杠 \	92
\’	表示一个单引号 ‘	39
\”	表示一个双引号 “	34
\ddd	1~3位八进制数（ASCII码）表示的字符	
\xhh	1~2位十六进制数（ASCII码）表示的字符	

□ 字符串常量

用“双引号” 括起的“字符序列”

例如："ABC"、"123"、"a"、"\n\t"、"\nGood morning“

- **存储时，编译器在字符串末尾自动添加字符’\0’作为字符串结束标志，但它不属于字符串的一部分。**

  例如执行 cout << “abc” <<endl; 将输出字符串”abc”，但字符串”abc”在内存中存储时会在字符’c’的后面再存储一个空字符’\0’作为字符串”abc”的结束标志。

- **注意区分 “a” 和 ’a’：类型不同，存储空间大小不同。**

2.3.1.2 符号常量（标识符常量）

- 以标识符来代表的常量
- 定义方式
  - ➢ 宏定义方式：#define 标识符 字面值
    - 一般用大写字母，例如 #define PRICE 30
    - 在编译之前（预编译阶段）由编译器将程序中的符号常量替换为它所代表的字面值，再对程序进行编译。
  - ➢ 常变量方式：const 类型说明符 标识符 = 字面值
    - 例如 const int price = 30
    - 使用方法详见“常变量”

说明：从字面形式上即可识别类型的常量称为“直接常量”或“字面值”。

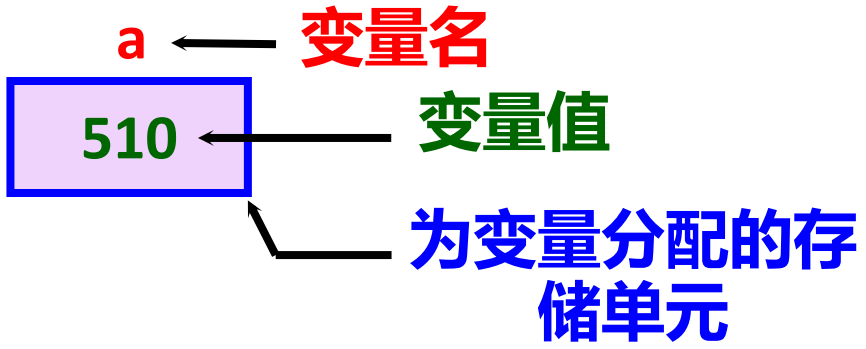
例2.1 符号常量的使用

```
1  #include <iostream>
2  using namespace std;
3
4  #define PRICE 30
5
6  int main( )
7  {
8      int num = 10, total;
9
10     total = num * PRICE;
11
12     cout << "total = " << total << endl;
13
14     return 0;
15 }
```

定义符号常量

使用符号常量

2.3.2 变量



口 相关概念

- 在程序运行过程中，变量的值是可以改变的。
- 变量用标识符表示，称为**变量名**；系统**为变量分配存储单元**，存储变量的值，即**变量值**。
- 程序执行时通过变量名来存取变量值。
  - 变量名实际上是以一个名字代表一个内存地址（即对应的存储单元）。
  - 在程序编译或运行时，编译系统会给每一个变量名分配对应的内存存储单元用于存储变量值。
  - 从变量中取值，实际上是通过变量名找到其对应的内存单元，从该单元中读取变量值。

□ 变量定义

- C/C++是强类型语言，变量必须先定义后使用
- 变量定义的一般格式：

数据类型 变量1, 变量2, …

  - 数据类型决定：1、占内存的字节数；2、数的取值范围；3、数据的有效操作
  - 变量名必须是合法标识符
  - 变量要先定义，后使用

```
int main() {
    int  a,b=2;
    float  data;
    a=1;
    data=(a+b)*1.2;
    …
    return 0;
}
```

□ 变量的赋值

方法1： 先定义变量，再赋值

例2.2

```
int i, j;
double s;
I = 4;
J = 4*i-11;
S = 0.02*1000.0;
```

方法2：初始化，即定义变量的同时赋值

例2.3

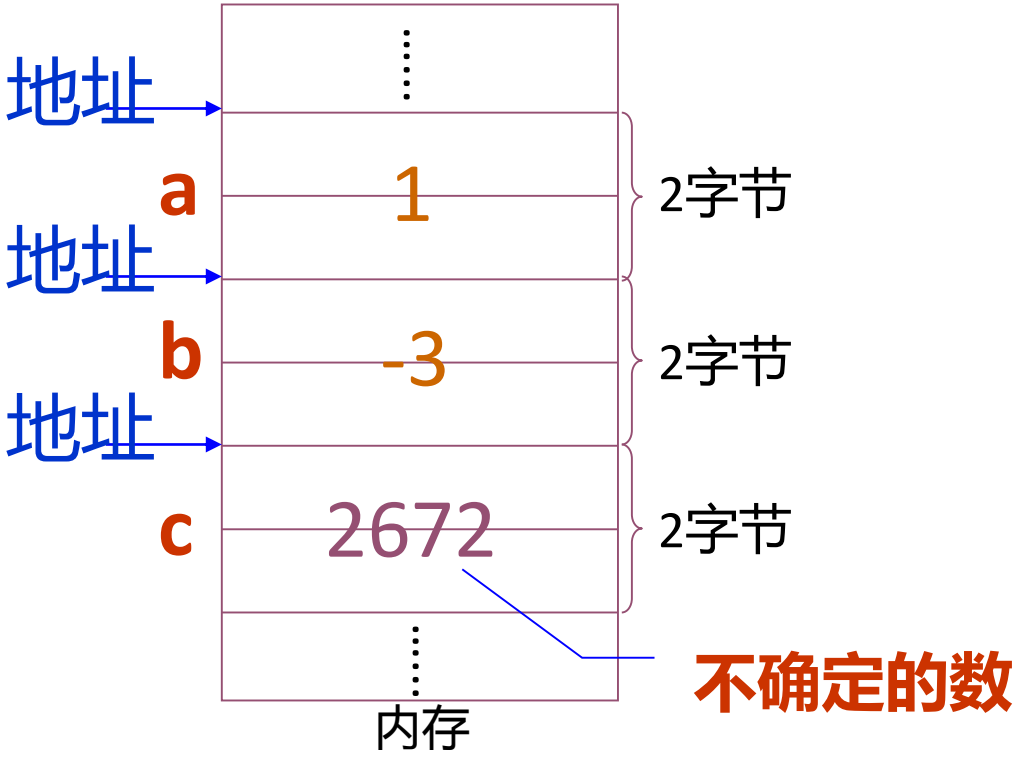
```
int i = 0;
i = i + 2;

/*若对几个变量赋同一个值，
不能写成 int a=b=c=5;*/
int a=5, b=5, c=5;
```

说明：*变量若未初始化，则值是默认值或随机值（由变量类型决定）*

□ 变量定义及初始化的作用

int a = 1, b = -3, c; /*编译系统根据变量的类型其分配相应
大小（字节数）的内存单元/



例：分析i的值分别是多少？

int i=0;	int i;
i=i+2;	i=i+2;

◆ 常变量

程序运行期间，值不能改变的变量

- 定义形式：

const 类型说明符 变量名

- 常变量必须在变量定义的同时进行初始化。

例如：const float pi = 3.1415926

2.4 运算符和表达式

- 运算符和表达式
  - 运算符的含义、使用方法、优先级别和结合性
  - 表达式的概念
  - 部分运算符及其表达式介绍
- 表达式的求值
  - 不同类型的数值数据间的混合运算
  - 复杂表达式的求值

2.4.1 运算符和表达式

2.4.1.1 运算符

C++ Operator Precedence

The following table lists the precedence and associativity of C++ operators. Operators are listed top to bottom, in descending precedence.

Precedence	Operator	Description	Associativity
1	::	Scope resolution	Left-to-right
2	a++ a--	Suffix/postfix increment and decrement	
	type() *type*{}	Functional cast	
	a()	Function call	
	a[]	Subscript	
	. ->	Member access	
3	++a --a	Prefix increment and decrement	Right-to-left
	+a -a	Unary plus and minus	
	! ~	Logical NOT and bitwise NOT	
	(*type*)	C-style cast	
	*a	Indirection (dereference)	
	&a	Address-of	
	sizeof	Size-of[note 1]	
	new new[]	Dynamic memory allocation	
	delete delete[]	Dynamic memory deallocation	
4	.* ->*	Pointer-to-member	Left-to-right
5	a*b a/b a%b	Multiplication, division, and remainder	
6	a+b a-b	Addition and subtraction	
7	<< >>	Bitwise left shift and right shift	
8	< <=	For relational operators < and ≤ respectively	
	> >=	For relational operators > and ≥ respectively	
9	== !=	For relational operators = and ≠ respectively	
10	a&b	Bitwise AND	
11	^	Bitwise XOR (exclusive or)	
12	|	Bitwise OR (inclusive or)	
13	&&	Logical AND	
14	||	Logical OR	
15	a?b:c	Ternary conditional[note 2]	Right-to-left
	throw	throw operator	
	=	Direct assignment (provided by default for C++ classes)	
	+= -=	Compound assignment by sum and difference	
	*= /= %=	Compound assignment by product, quotient, and remainder	
	<<= >>=	Compound assignment by bitwise left shift and right shift	
	&= ^= |=	Compound assignment by bitwise AND, XOR, and OR	
16	,	Comma	Left-to-right

1. ↑ The operand of sizeof can't be a C-style type cast: the expression `sizeof (int) * p` is unambiguously interpreted as `(sizeof(int)) * p`, but not `sizeof((int)*p)`.
2. ↑ The expression in the middle of the conditional operator (between ? and :) is parsed as if parenthesized: its precedence relative to ?: is ignored.

来源：
http://en.cppreference.com/w/cpp/language/operator_precedence

2.4.1.2 表达式

依据C++的语法规则，用运算符和括号将操作数连接起来得到的式子，称为C++的表达式。

2.4.1.3 部分运算符及其表达式介绍

- 算术运算符及其表达式
- 自增/自减运算符及其表达式
- 赋值运算符及其表达式
- 逗号运算符及其表达式
- 关系运算符及其表达式
- 逻辑运算符及其表达式
- 条件运算符及其表达式

□ 算术运算符

算术运算符	*	/	%	+	–
优先级	5			6	
结合性	从左到右				

说明：

- **“–”表示负号（单目运算符）时，优先级为3，结合性为从右到左；**
- **两整数相除（ / ），结果为整数（多数编译器采用向零取整）；**
- **求余运算符（%）要求两个操作数均为整型数据。**

算术表达式

表达式	5/2	-5/2.0	5%2	-5%2	1%10	5%1	5.5%2
结果	2	-2.5	1	-1	1	0	Error

自增/自减运算符

自增运算符	自减运算符
++	--
优先级：后置2，前置3	
结合性：后置从左到右，前置从右向左	

自增/自减表达式

- 运算符前置：++i 或 --i （先计算i=i+1或i=i-1，再使用i值）
- 运算符后置：i++ 或 i-- （先使用i值，再计算i=i+1或i=i-1）

表达式	j=3		a=3, b=5	
	k=++j	k=j++	c=(--a)*b	c=(a--)*b
结果	k=4, j=4	k=3, j=4	c=10, a=2	c=15, a=2

赋值运算符

赋值运算符	复合赋值运算符
=	+= -= *= /= %= <<= >>= &= ^= |=
优先级：15	
结合性：从右向左	

赋值表达式

- 左侧必须是变量，不能是常量或表达式；
- 一般情况下，右侧操作数的类型应与左侧变量的类型相匹配。

表达式	已知int i, j=3; float k, t=3.14;		展开复合赋值表达式	
	k=j	i=t	a+=3	x*=y+8
结果	k=3.000000	i=3	a=a+3	x=x*(y+8)

逗号运算符

逗号运算符	,
优先级	16（最低级别）
结合性	从左向右

逗号表达式

- 使用形式

表达式1, 表达式2, …, 表达式n

- 整个逗号表达式的值等于最右侧表达式（即表达式n）的值

表达式	已知int i=1, j=3, k=6, t=2;
	i+2, j*4, k%4, t-3
结果	-1

口 关系运算符

关系运算符	<	<=	>	>=	==	!=
优先级	8				9	
结合性	从左到右					

关系表达式

- 关系表达式的值是逻辑值“真”或“假”，分别用“1”和“0”表示。

表达式	5>2>7>8	‘a’>0	‘A’>100
结果	0	1	0

逻辑运算符

逻辑运算符	逻辑与 &&	逻辑或 ||	逻辑非 !
优先级	13	14	3
结合性	从左到右		从右到左

逻辑表达式

- 逻辑运算的真值表

操作数		逻辑运算			
a	b	!a	!b	a && b	a || b
非0	非0	0	0	1	1
非0	0	0	1	0	1
0	非0	1	0	0	1
0	0	1	1	0	0

表达式	已知 int a = 3; char b = ‘B’;							‘c’&&‘d’	4||0&&2
	!a	!b	a&&b	!a&&b	a&&!b	!a||b	a||!b		
结果	0	0	1	0	0	1	1	1	1

- **逻辑运算的短路特性**

逻辑表达式求值时，并非所有的逻辑运算符都被执行，只是在必须执行下一个逻辑运算符才能求出表达式的值时，才执行该运算符。

a&&b&&c //只在a为真时，才判别b的值；只在a、b都为真时，才判别c的值

a||b||c //只在a为假时，才判别b的值；只在a、b都为假时，才判别c的值

求表达式的值，变量m和n的值
已知 a=1;b=2;c=3;d=4;m=1;n=1;
则 (m=a>b)&&(n=c>d)

□ 条件运算符

条件运算符	? :
优先级	15
结合性	从右向左

条件表达式

- 使用形式

  表达式1 ? 表达式2 : 表达式3

- 若表达式1的值为真，则条件表达式的值等于表达式2的值；否则等于表达式3的值。

表达式	(a==b)?'Y':'N'	
结果	若a=1, b=1 则结果为'Y'	若a=1, b=2 则结果为'N'

□ sizeof运算符

sizeof运算符	sizeof()
优先级	3
结合性	从右向左

- 使用形式

  sizeof(类型说明符)　　或　　sizeof(表达式)

- 运算结果 = 该（表达式）类型在当前系统存储时所需内存的大小（以字节为单位）

2.4.2 表达式的求值

2.4.2.1 不同类型的数值数据间的混合运算

整型、浮点型、字符型数据间可以进行混合运算

规则：

要将数据从不同类型转换为相同类型才能进行运算。

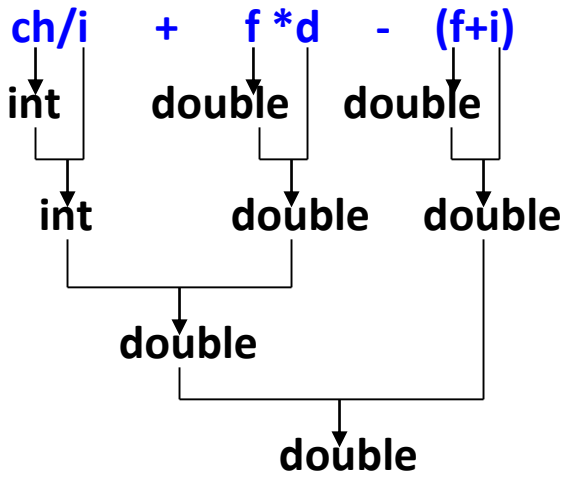
- 隐式类型转换：编译系统自动进行转换
- 强制类型转换：根据需要，手动进行转换

□ 隐式类型转换

算术运算时的隐式类型转换

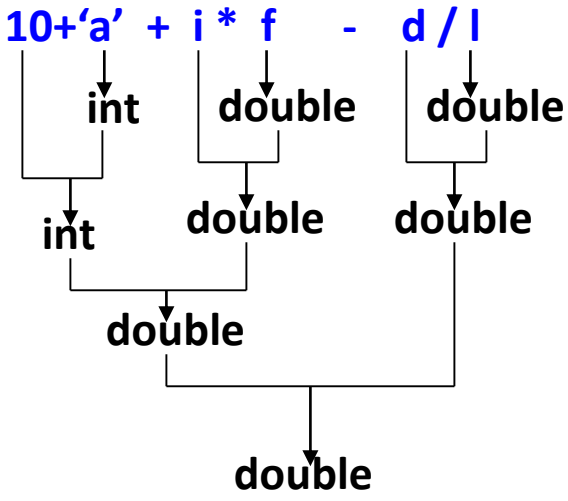
例2.4

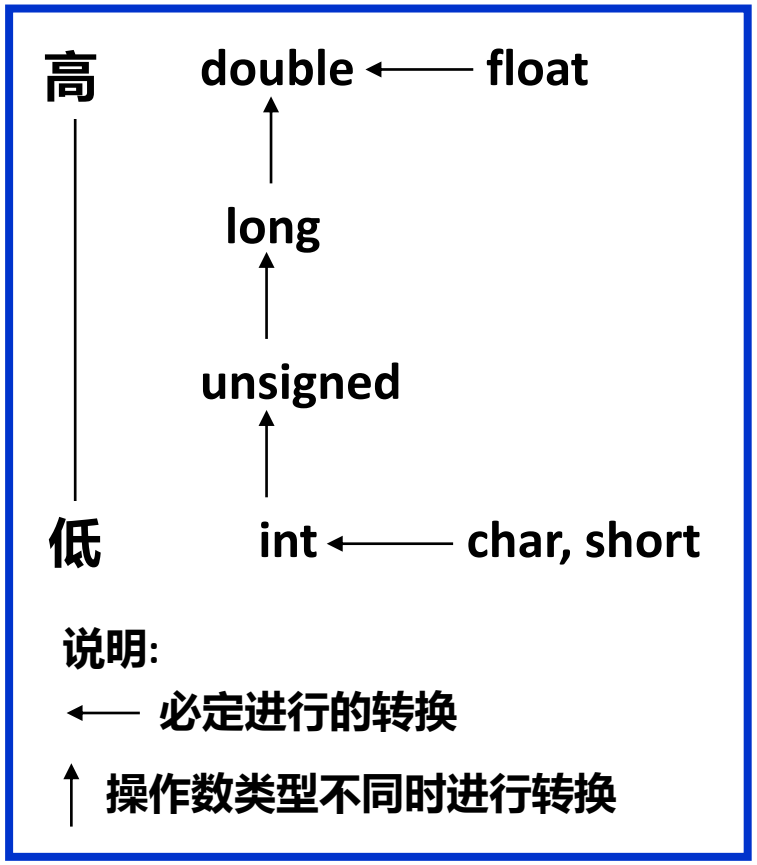
```
char ch;
int i;
float f;
double d;
```



例2.5

```
int  i;
float f;
double d;
long l;
```





- **赋值运算时的隐式类型转换**

若赋值运算符的两个操作数类型不一致，编译系统自动将右操作数的类型转换为左操作数的类型，然后再赋值。

- **整型变量 = 浮点型数据：舍弃小数部分。**
- **浮点型变量 = 整型数据：数值不变，但按浮点数形式存放。**
- **整型变量 = 字符型数据：**

  (1) 字符→无符号整型变量，则存入低8位，高8位补零；

  (2) 字符→有符号整型变量，则字符高位扩展。

- **long int型变量 = int型数据：把int型数据保存到long int的低16位，高位进行符号扩展。**

  int型变量 = long int型数据：long int型数据的低16位保存到int型变量。

C++中，以下4种情况时会发生隐式数据类型转换：

- **运算转换** —— 不同类型的数值数据间进行混合运算时，按照转换规则进行自动转换；
- **赋值转换** ------ 给变量赋值时，按照目标变量的类型进行自动转换；
- **输出转换** ------ 按照指定的输出类型格式进行自动转换；
- **函数调用转换** ------ 参数传递时，实参按照形参的类型进行自动转换。

□ 强制类型转换

(类型名)(表达式)　　//C风格，例如： (int)(x)

类型名(表达式)　　//C++风格，例如：int(x)、int(3.6)

说明：强制转换得到所需类型的中间变量，原变量类型不变

2.4.2.2 复杂表达式的求值

关键点：运算符的优先级和结合性

若int a=3, b=5, c; c=(a++)*(++b) ; 求表达式的值，a和b的值	a=4, b=6, c=18
a=3*5, a*4, a+5; 求表达式类型和值	逗号表达式，表达式值为20
x=(a=3, 6*3); 求表达式类型和值	赋值表达式，x的值为18
x=a=3, 6*a; 求表达式类型和值	逗号表达式，表达式值为18
若int a=3, b=2, c=1, d, f; 求下列表达式的类型和值 (1) b+c<a; (2) d=a>b; (3) f=a>b>c	(1) 关系表达式，表达式值为0 (2) 赋值表达式，d的值为1 (3) 赋值表达式，f的值为0
若int i=1, j=7, a; 求表达式a=i+(j%4!=0)的类型和值	赋值表达式，a的值为2
5>3&&2||8<4-!0; 求表达式类型和值	逻辑表达式，表达式值为1
max=a>b? a:b; 求表达式类型和值	赋值表达式，max的值为max(a, b)

CH02 End