

C程序设计语言

第2章 类型、运算符与表达式

孙志岗
sun@hit.edu.cn
http://sunner.cn

标识符 (Identifiers)

- 用户自定义的符号叫标识符
 - 如变量名、函数名、宏和类型名
- 标识符由字母、数字和下划线组成，大小写敏感
- 不可以是数字开头
- 标识符要直观，能表达它的功能
- 下划线和大小写通常用来增强可读性
 - ~~variable_name~~
 - variable_name, VARIABLE_NAME
 - VariableName, variableName ✓
- 关键字 (keyword) 不可作为标识符
 - int, float, for, while, if等 (教材164页)
- 某些功能的变量采用习惯命名
 - 如: for语句所采用的循环变量习惯用i, j, k

2004-12-19

Types, Operators and Expressions

2



b, B, KB, MB, GB, TB

- Megabyte(MB), 中文叫法: 兆
- Kilobyte(KB), 中文叫法: K
- Byte, 中文叫法: 字节
- bit, 中文叫法: 位
- Gigabyte(GB), 中文叫法: G
- Terabyte(TB), 中文叫法: T

1 TB == 1,024 GB

1 GB == 1,024 MB

1 MB == 1,024 KB

1 KB == 1,024 B

1 B == 8 b

2004-12-19

Types, Operators and Expressions

3



b, B, KB, MB, GB, TB

- 世界上有10种人，1种人懂二进制，1种人不懂二进制
- 一个位有多大?
 - 只能是“0”或者“1”，这叫二进制
- 二进制诠释了计算机的哲学
 - 种类众多的复杂事物都是由若干种简单事物构成

祝您中秋节快乐!



2004-12-19

Types, Operators and Expressions

4



b, B, KB, MB, GB, TB

一个字节有多大?

- 可以表示数字0~255
- 保存一个字符（英文字母、数字、符号），ASCII编码
- 两个字节保存一个汉字
 - GB2312, 6763字
 - GB13000.1, 20902字
 - GB18030, 27533字
 - BIG5, 13000字
- 两个字节保存一个宽字符，UNICODE编码

基本数据类型 (Data Type)

- **int**
 - 整数，在目前绝大多数机器上占4个字节
 - TC2中是2个字节
 - 所占字节数取决于机器字长
- **float**
 - 单精度浮点数，一般是4个字节长
- **double**
 - 双精度浮点数，一般是8个字节长
- **char**
 - 字符，一般是1个字节长
 - 用来表示256个ASCII字符，或者0~255的整数

数据类型修饰符

- **short**
 - **short int**，短整数，一般2个字节长。通常简写为**short**
- **long**
 - **long int**，长整数，一般是4个字节长。通常简写为**long**
 - **long double**，高精度浮点数，一般是10个字节长。
- **signed**
 - 用来修饰**char**、**int**、**short**和**long**，说明他们是有符号的整数（正整数、0和负整数）。一般缺省都是有符号的，所以这个修饰符通常省略
- **unsigned**
 - 用来修饰**char**、**int**、**short**和**long**，说明他们是无符号的整数（正整数和0）

整型类型的取值范围

CHAR_MAX	UCHAR_MAX或SCHAR_MAX	char类型的最大值
CHAR_MIN	0或SCHAR_MIN	char类型的最小值
INT_MAX	+32767	int类型的最大值
INT_MIN	-32767	int类型的最小值
LONG_MAX	+2147483647	long类型的最大值
LONG_MIN	-2147483647	long类型的最小值
SCHAR_MAX	+127	signed char类型的最大值
SCHAR_MIN	-127	signed char类型的最小值
SHRT_MAX	+32767	short类型的最大值
SHRT_MIN	-32767	short类型的最小值
UCHAR_MAX	255	unsigned char类型的最大值
UINT_MAX	65535	unsigned int类型的最大值
ULONG_MAX	4294967295	unsigned long类型的最大值
USHRT_MAX	65535	unsigned short类型的最大值

浮点类型的取值范围

FLT_RADIX	2	指数表示的基数, 如2、16
FLT_ROUNDS		加法的浮点舍入规则
FLT_DIG	6	精度的十进制数字数
FLT_EPSILON	1E-5	使 $1.0+x$ (1.0成立的)最小的 x
FLT_MANT_DIG		基数为FLT_RADIX的尾数中的数字数
FLT_MAX	1E+37	最大浮点数
FLT_MAX_EXP		使 FLT_RADIX^{n-1} 可表示的最大的 n
FLT_MIN	1E-37	最小规范化的浮点数
FLT_MIN_EXP		使 10^n 为规范化数的最小的 n
DBL_DIG	10	精度的十进制数字数
DBL_EPSILON	1E-9	使 $1.0+x$ 1.0成立的)最小的 x
DBL_MANT_DIG		基数为FLT_RADIX的尾数中的数字数
DBL_MAX	1E+37	最大双精度浮点数
DBL_MAX_EXP		使 FLT_RADIX^{n-1} 可以正常表示的最大的 n
DBL_MIN	1E-37	最小双精度浮点数
DBL_MIN_EXP		使 10^n 为规范化数的最小的 n

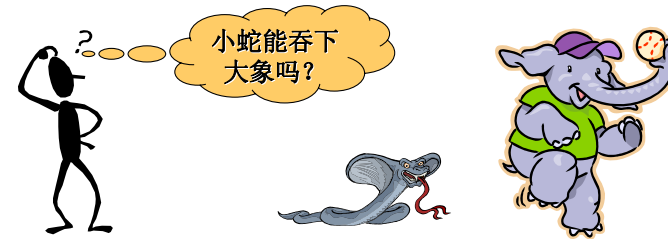
2004-12-19

Types, Operators and Expressions

9

超出取值范围会怎样?

- TC2中int的范围是-32767~32767
- 如果我们给它一个小于-32767或者大于32767的数会如何呢?
- 现场编程测验.....



2004-12-19

Types, Operators and Expressions

10

溢出 (Overflow) 造成的危害

- 一台安装了Windows 95/98的机器, 如果连续运行**49.7**天没有重新启动, 可能死机
- 原因:
 - Windows自启动时刻起, 有一个计数器, 记录系统已经运行了多少毫秒。这个计数器是个**unsigned long** 类型的变量
 - **unsigned long**的最大值是: **4294967295**
 - 一天有 $24*60*60*1000 = 86400000$ 毫秒
 - $4294967295 / 86400000 = 49.71026961805.....$
 - 当**49.7**天的时候, 此计数器会溢出, 引起死机

2004-12-19

Types, Operators and Expressions

11

浮点数的陷阱

```
#include <stdio.h>

main()
{
    float f;

    f = 123.456;

    if (f == 123.456)
        printf("f is equal to 123.456 indeed.");
    else
        printf("In fact, f is equal to %f\n", f);
}

■ 运行结果会是什么?
```

float.c

2004-12-19

Types, Operators and Expressions

12

浮点数的陷阱

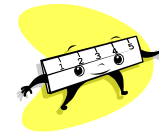
- `float`的精度低，较易发生精度带来的相等性判断问题
- `double`精度高，这个问题发生的概率小一些，但也存在
- 解决办法：

```
if (fabs(f - 123.456) < 1E-5)
    .....
```

根据精度要求设定

使用变量要注意

- 不要对变量所占的字节数想当然
 - 用`sizeof`获得变量或者数据类型的长度
 - 用ANSI C定义的宏确定数据的表示范围，解决溢出问题



常数 (Constant)

- 整型常数
 - 123、456
 - 123456
 - 123L、123L、123456L、123456L
- 浮点常数
 - 123.45、456.78
 - 1e-2、4.5e3
 - 123.45f、456.78F、1e-2f、4.5e3F
 - 123.45L、456.78L、1e-2L、4.5e3L

八进制与十六进制常数

- 以数字“0”开始的整型常数是八进制数
 - 010和10大小不一样
 - 因为八进制并不常用，所以此种表示法比较少见，因而常被用错
- 以“0x”或者“0X”开始的整型常数是十六进制
 - A~F和a~f用来表示十进制的10~15
 - 0x11, 0x05, 0xFA, 0xFF
 - 十六进制的形式比较常用，尤其在进位一级的控制的时候

枚举 (Enumeration) 常数

- 一个几乎被遗忘的角色
- 从程序来窥其一斑

```
enum weeks {
    MON, TUE, WED, THU, FRI, SAT, SUN
};
enum weeks today, tomorrow;

today      = MON;
tomorrow   = today + 1;
if (tomorrow == TUE)
    printf("Tomorrow is Tuesday.\n");
else
    printf("Tomorrow is NOT Tuesday.\n");
```

enum.c

2004-12-19

Types, Operators and Expressions

21

变量声明

- 变量必须“先定义，后使用”
 - 所有变量必须在第一条可执行语句前定义
- 声明的顺序无关紧要
- 一条声明语句可声明若干个同类型的变量，变量名之间用逗号分隔
- 变量定义后，即占用内存，可向其存入各种数据，并可通过变量名使用数据
- 声明变量，是初始化变量的最好时机
 - 不被初始化的变量，其值为危险的随机数

```
char  esc = '\\';
int   i   = 0;
int   limit = MAXLINE + 1;
float eps = 1.0e-5;
```

2004-12-19

Types, Operators and Expressions

22

常量

- 用 `const` 修饰定义的变量为常量
 - `const int i=0;`
- 常量只能在定义时赋值，然后不能再改变其值
- 常数、常量、宏和枚举，都可以用来表示一个永远不会改变的数
 - 前者不建议直接使用，而后三者代替
 - 后三者的工作机理是完全不同的，达到的效果也不尽相同

2004-12-19

Types, Operators and Expressions

23

C语言中的运算

- 计算机只会计算
- 任何事物都要被表示成数字和公式的形式后，才能被计算机计算（被计算机处理）
 - 事物到数字和公式的转换过程叫数学建模
- 因为：事物在计算机内的处理都是一种计算
- 又因为：计算就要有操作数、运算法则和计算结果
- 所以：事物在计算机内的处理都有操作数、运算法则和计算结果
- 计算结果你可以留用，也可以忽略

2004-12-19

Types, Operators and Expressions

24

算术运算符

- +, -, *, /
 - 加、减、乘、除运算
 - 四则混合运算中, 先算乘除, 后算加减, 先算左, 后算右
- %
 - 求余运算



2004-12-19

Types, Operators and Expressions

25

关系运算符

- >, >=, <, <=, ==, !=
 - 大于, 大于等于, 小于, 小于等于, 等于, 不等于
 - 关系运算符运算出的结果为0和1
 - 0, 表示假, 即该关系不成立
 - 1, 表示真, 即该关系成立
- 在所有涉及到真假判断的地方, 0表示假, 非0表示真



2004-12-19

Types, Operators and Expressions

26

找别扭

- | | |
|---|--|
| ■ <pre>int a=1; if (a == 0) printf("OK");</pre> | ■ <pre>int a=1; if (a = 0) printf("OK");</pre> |
| ■ <pre>int a=0; if (a == 0) printf("OK");</pre> | ■ <pre>int a=0; if (a = 0) printf("OK");</pre> |

2004-12-19

Types, Operators and Expressions

27

== 和 =

- | | |
|------------------------------------|------------------------------------|
| ■ <pre>int a; a = 0; a == 1;</pre> | ■ <pre>int a; a == 0; a = 1;</pre> |
|------------------------------------|------------------------------------|
- 一定要分清==和=
- 下面用法能起点小作用:
- | | |
|---|--|
| ■ <pre>int a=0; if (0 == a) printf("OK");</pre> | ■ <pre>int a=0; if (0 = a) printf("OK");</pre> |
|---|--|
- 编译出错

2004-12-19

Types, Operators and Expressions

28

逻辑运算符

- 逻辑运算也被称为布尔（Boolean）运算，运算结果也是1和0
- &&
 - 与运算
 - (a > b && b > c); a大于b, 并且b大于c
- ||
 - 或运算
 - (a > b || b > c); a大于b, 或者b大于c
- !
 - 求反
 - (!a); 如果a是0, 结果非0; 如果a是非0, 结果是0
 - 并不改变a的值

2004-12-19

Types, Operators and Expressions

29

类型转换

- 在进行赋值操作时，会发生类型转换
- 将取值范围小的类型转为取值范围大的类型是安全的
- 反之是不安全的
 - 如果大类型的值在小类型能容纳的范围之内，则平安无事
 - 但是，浮点数转为整数，会丢失小数部分（非四舍五入）
 - 反之，转换后的结果必然是错误的，具体结果与机器和实现方式有关。避免如此使用

2004-12-19

Types, Operators and Expressions

30

字符串与数值类型之间的转换

- `int i = "123"`
 - 这样用是不行地
- `atof()`, `atoi()`, `atol()`
 - 把字符串转为double, int和long
 - 定义在stdlib.h中
- `sprintf()`
 - 可以用来把各种类型的数值转为字符串
 - 定义在stdio.h中

2004-12-19

Types, Operators and Expressions

31

自动类型转换

- 两个同种数据类型的运算结果，还是该类型
- 两个不同种数据类型的运算结果，是两种类型中取值范围更大的那种
 - `long double > double > float > long > int > short > char`
 - 只要两者中有一个是unsigned, 就都转为unsigned再计算
- 把数据赋值给另外一种类型变量也会发生自动类型转换
 - 从小到大, 顺利转换
 - 从大到小, 发出警告（好的编译器会给出）

2004-12-19

Types, Operators and Expressions

32

类型强转

- 可以通过“(类型)表达式”的方式把表达式的值转为任意类型
 - 强转时，你必须知道你在做什么
 - 强转与指针，并称C语言两大神器，用好了可以呼风唤雨，用坏了就损兵折将



2004-12-19

Types, Operators and Expressions

33

加一和减一运算符

- $i++$, $i--$, $++i$, $--i$
 - $++$ 让参与运算的变量加1， $--$ 让参与运算的变量减1
 - 运算符为后缀，先取 i 的值，然后加/减1
 - 运算符为前缀，先加/减1，然后取 i 的值
- 在一行语句中，使用加1或者减1运算的变量只能出现
 - 不仅可读性差，而且因为编译器实现的方法不同，容易导致不同编译器运行效果不一样，贻害无穷

2004-12-19

Types, Operators and Expressions

34

位操作运算符

- | | |
|---|--|
| ■ $\&$ <ul style="list-style-type: none">- 按位与运算 | ■ \ll <ul style="list-style-type: none">- 按位左移运算 |
| ■ $ $ <ul style="list-style-type: none">- 按位或运算 | ■ \gg <ul style="list-style-type: none">- 按位右移运算 |
| ■ \wedge <ul style="list-style-type: none">- 按位异或运算 | ■ \sim <ul style="list-style-type: none">- 按位求反 |

2004-12-19

Types, Operators and Expressions

35

赋值运算符

- 赋值运算的结果是被赋值变量赋值后的值
 - $a = b = c = 0;$
- 下面两个语句是等价的
 - $i = i + 2;$
 - $i += 2;$
- $+$ 、 $-$ 、 $*$ 、 $/$ 、 $\%$ 、 \ll 、 \gg 、 $\&$ 、 \wedge 、 $|$ 运算符都可以按此种方式处理
- 这种形式看起来更直观，而且执行效率一般也能更高一些

2004-12-19

Types, Operators and Expressions

36

条件表达式

- 把a和b中的最大值放入z中
 - `if (a > b)`
 `z = a;`
 - `else`
 `z = b;`
- `z = (a > b) ? a : b;`
- 此种表达式切忌用得过于繁杂

2004-12-19

Types, Operators and Expressions

37

优先级

- () [] -> .
- ! ~ ++ -- + - * & (类型) sizeof
- * / %
- + -
- << >>
- < <= > >=
- == !=
- &
- ^
- |
- &&
- ||
- ? :
- = += -= *= /= %= &= ^= |= <<= >>=
- ,

2004-12-19

Types, Operators and Expressions

38

优先级

- 能背下优先级表的人凤毛麟角
 - 脑细胞太宝贵了，不能用来死记硬背
- 用括号来控制运算顺序更直观、方便，并减少出错的概率
 - 先算乘除，后算加减，有括号就先算括号里的
 - 括号太多，有时候不清晰
 - 注意用空格做好分隔
 - 实在不行就拆分表达式

2004-12-19

Types, Operators and Expressions

39

这一章我们学到了

- 标识符的命名规则
- 数据类型
 - `char, short, int, long, float, double, long double`
 - `signed, unsigned`
 - `enum`
- `sizeof`
- 常数、转义字符
- 算术运算、关系运算、逻辑运算、加一/减一运算、位运算、赋值运算
- 类型转换
- ? :
- 优先级

2004-12-19

Types, Operators and Expressions

40

