# Expressions

Eric McCreath

# Expressions on integers

There is the standard set of interger operators in c.

We have:

```
y = 4 + 7;  // add
y = 7 - 3;  // subtract
y = 3 * x;  // multiply
y = x / 3;  // integer divide
y = x % 3;  // integer remainder
y = -x;     // integer negate
```

*c* uses the standard ordering when determining the order these operators are applied.  e.g. x * 7 + 5 is (x*7) + 5 and not x * (7 + 5) .  The basic rule of thumb for precedence: "if you didn't learn it in high school then add brackets".  Or if you are uncertain then add brackets.

Interestingly, the way the '%' operator works on negative numbers is not exactly specified in the *c* language.

# The assignment expression

The assignment operator is also an expression that returns the value that it assigned.

```
int x,y;
x = y = 7;  // this is okay because y is assigned to 7 and
            // this assignment operator returns 7 which is assigned to x.
```

Take care with this as it can make your code harder to understand.

# Expressions on floats

Like the expressions on integers there is a standard set of operations for floats.

```
y = 4.0 + 7.0;  // add
y = 7.0 - 3.0;  // subtract
y = 3.0 * x;    // multiply
y = x / 3.0;    // divide
y = -x;         // negate
```

If you mix integers with floats, c will convert the integer to a float and complete the operation using a floating operation.

If you wish convert a float to an integer then you can cast it with (int). e.g.

```
int value;
value = (int) 3.4;
```

# Comparison

You can compare both integers and floats using:

```
<       less than
<=      less than or equal to
>       greater than
>=      greater than or equal to
==      equal to
!=      not equal to
```

If you are using "==" on floating point numbers then this would generally indicate a problem with your code.

# Working with Booleans

Booleans are represented using integers (remeber 0 is false, non-zero is true). Operators on booleans include:

```
&&    logical add
||    logical or
!     logical not
```

The == operator may not work on booleans as two true booleans in c may have different values. You can use "(a && b) || (!a && !b)" to see if booleans a and b are equal.

Generally avoid side effects within boolean expressions.

# operations on bits

There is a number of operations that you can do on bits within an integer. These include:

```
x >> 4      right shift x by 4 bits
x << 2      left shift x by 2 bits
x & y       bitwise "and" of x with y
x | y       bitwise "or" of x with y
~x          negate each bit within x
x ^ y       bitwise "xor" of x with y
```

Arithmetic right shift fills the left most bits with the sign bit. Logical right shift fills the left most bits with 0.

ANSI c does not specify if arithmetic or logical shift is used.

# conditional operator

One handy, but sometimes forgotten, operator is the ternary conditional operator.

```
(boolean_expression ? expression1 : expression1)
   // in the above expression if the boolean_expression evaluates to true
   // then expression1 is evaluated and returned otherwise,
   // expression2 is evaluated and returned.
```

The use of this operator will often save on using a extra local variable along with a if-else conditional.  e.g. Using a if-else:

```
int maxval;
if (a > b) {
   maxval = a;
} else {
   maxval = b;
}
printf("The max is : %d\n",maxval);
```

Using ternary condidtional:

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

# math.h

Operators like square root, power, sin, cos, ... are not part of the c language. However, there are standard libraries that enable you to compute these functions.

The math.h library contains a useful range of mathematical operators. These include:

```
    cos(x)       calculate the cosine of x
    sin(x)       calculate the sine of x
    acos(theta) calculate the arc sine of theta
    pow(x,y)     take x to the power of y
    M_PI         the pi constant
    sqrt(x)      calculate the square root of x
```

See:

```
http://en.wikibooks.org/wiki/C_Programming/C_Reference/math.h
```

for a more extensive list.

On some compilers you may need to link the math library for these to work (in gcc use the "-lm" option).

# Exersizes

- Write a program that calculates the area of a triangle given the lengths of the sides. The program should take input on a single line with three space separated floats which are the side lengths of the triangle. It should output on a single line the area of the triangle. You may assume the triangles are possible. Hint using Heron's formula (where $a$ , $b$ , $c$ are the side lengths):

$$\text{Area} = \sqrt{s(s-a)(s-b)(s-c)} \text{ where } s = \frac{a+b+c}{2}$$