Math functions

We have floating numbers that we can use for doing many of our engineering calculations. However, basic C does not provide for any math beyond the the standard add, subtract, multiply, divide — not even raising to a power.

Math capabilities are enhanced through a library of math functions. To access them, we must use another includes statement:

```
#include <math.h>
```

Like stdio.h, this is a header file that defines all of the available math function. When the compiler hits this directive in your code, it grabs a file that has all of the code in it and compiles it along with your code specific to your program.

Some useful functions in math.h

`z = pow(x, y)`	raises **x** to y-th power, assigns it to **z**
`y = sqrt( x )`	takes square-root of **x**, assigns it to **y**
`y = log( x )`	takes natural log of **x** (base e), assigns it to **y**
`y = log10( x )`	takes log of **x** (base 10), assigns it to **y**
`y = log2( x )`	takes log of **x** (base 2), assigns it to **y**
`y = sin( x ) [cos, tan]`	takes sine of **x** (x in radians), assigns it to **y**
`y = asin( x ) [acos, atan]`	takes arcsine of **x**, assigns it to **y (y in radians)**
`y = sinh( x ) [cosh, tanh]`	takes hyperbolic sine of **x**, assigns it to **y**
`y = asinh( x ) [acosh, atanh]`	takes hyperbolic arcsine of **x**, assigns it to **y**
`y = abs( x )`	takes absolute value of **x**, assigns it to **y** (integers)
`y = fabs( x )`	takes absolute value of **x**, assigns it to **y** (float)
`y = erf( x ) [erfc]`	takes error function of **x**, assigns it to **y**
`z = fmax(x, y) [fmin]`	takes larger of **x** & **y**, assigns it to **z**

There are more. See https://en.wikipedia.org/wiki/C_mathematical_functions

Examples

```
double pi = 3.14159;
double radius = 16.43257;
double area;
area = pi*pow(radius, 2);

double x = 27.345, y;

y = log( x );      //answer is 3.3085
y = log10( x );    //answer 1.43511
y = log2( y );     //answer 0.52116

double x = -4.37e8, y;

y = fabs( x );       //4.37e8
x = sqrt( y );       //answer 2.0904e4
x = pow( y, 0.5 );   //answer 2.0904e4
```

Nesting expressions

It is not necessary to have separate variables for each step in the calculation. The result of one function can nested as the input to the next function. Don't get carried away, though.

```
double x = -27.345, y, z, q, r = 4.0;
y = fabs( x );                              // 27.345
z = log( y );                               // 3.30853
q = pow( z, r);                             // 119.8236
printf("q = %5.4e.\n\n, y);                 //output is 1.1982e+02
```

```
double x = -27.345, y, r = 4.0;
y = pow( log( fabs( x )), r);
printf("y = %5.4e.\n\n, y);       //output is 1.1982e+02
```

```
double x = -27.345, r = 4.0;
printf("result is %5.4e.\n\n, pow( log( fabs( x )), r));
```