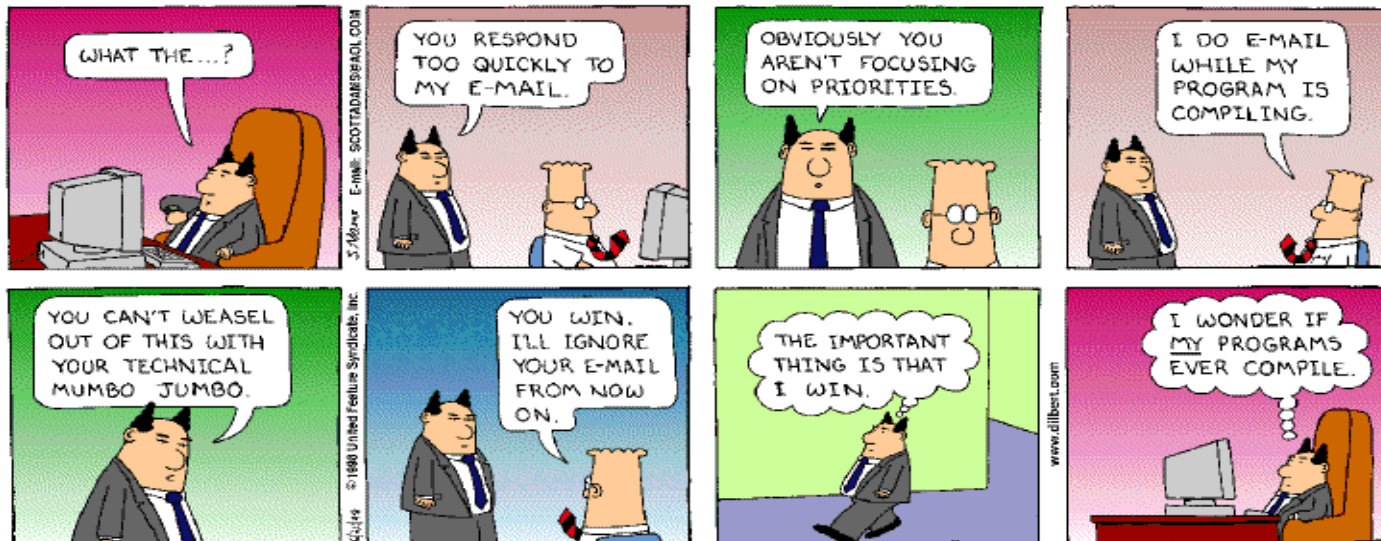


CIV 201
COMPUTER METHODS FOR PROBLEM SOLVING

Lecture 3
Decisions, Decisions



The *For* Loop

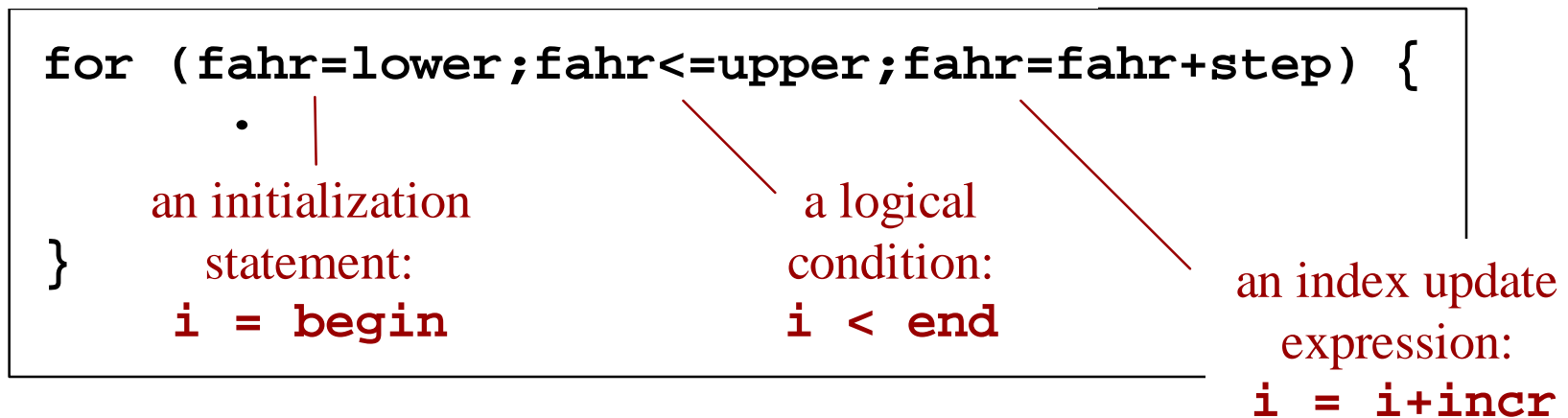
Recall the **while** loop from FahrCels3:

```
fahr=lower;
while(fahr<=upper) {
    .
    .
    .
    fahr=fahr+step;
}
```

This pattern is very common.
In such cases, it is better to use
a **for** loop

General pattern:
for (i=begin;i<end;i=i+incr){}

There are **three** parts inside the parens:



Note: there are only *two* semicolons.

More on For

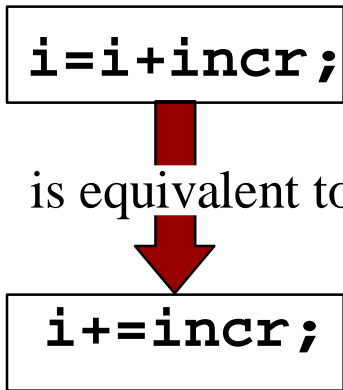
Recall the basic pattern:

```
for (i=begin;i<end;i=i+incr) {  
    .  
    .  
}
```

Variable **i** is called the *looping index*.

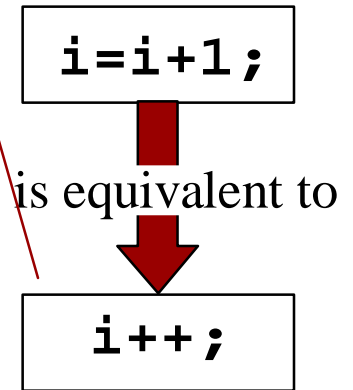
The looping index should always be an integer.

The symbol **++** is called the *increment operator*. It means add **1** to the variable to which it is attached. Right here. Right now.



```
for (i=begin;i<end;i++){  
    .  
    .  
}
```

This loop is by far the most common loop construct.



Computing Square Roots

The algorithm:

Given: a number x Guess a value for \sqrt{x} , say $x/2$ Call it r . Divide it into x .	Call that number s . If $r > \sqrt{x}$, then $s < \sqrt{x}$, and vice versa. Okay, average the two and call that r : $r \leftarrow \frac{r+s}{2}$ Repeat these steps until $r^2 \approx x$
--	---

In Java:

Using a **do-while** loop

```
r = x/2;  
do {  
    s = x/r;  
    r = (r+s)/2;  
    err = Math.abs(x-r*r);  
} while (err > EPSILON);
```

```
r = x/2;  
err = Math.abs(x - r*r);  
while (err > EPSILON) {  
    s = x/r;  
    r = (r+s)/2;  
    err = Math.abs(x - r*r);  
}
```

In this case, a **do-while** loop is better!

Computing Square Roots: From 1 to 20

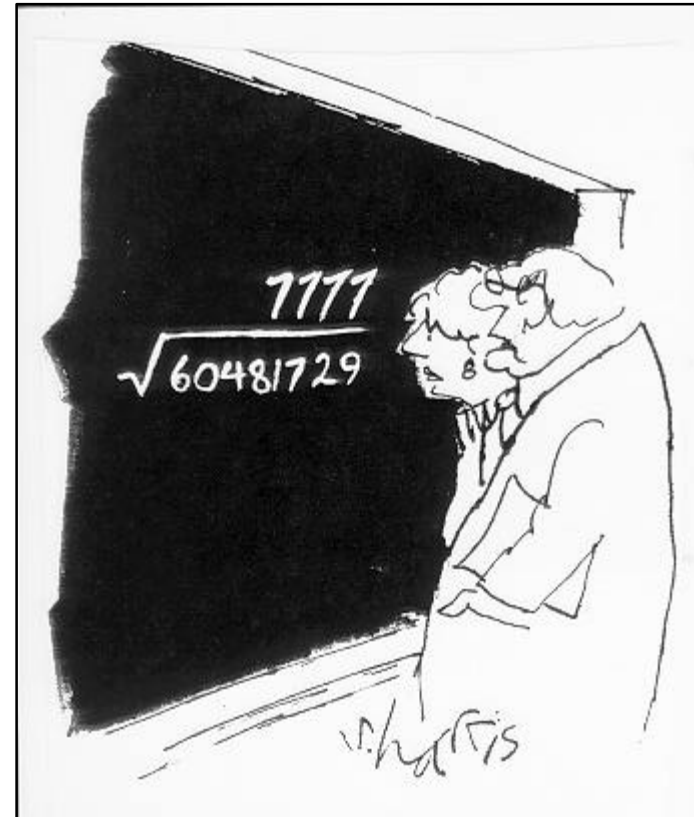
Here's the whole program:

```
import myutil.*;
public class Sqrt
{
    static final double EPSILON = 1.0e-10;
    public static void main(String[] args)
    {
        int i;
        double x,r,s,err;
        for (i=1; i<20; i++) {
            x = i;
            r = x/2;
            do {
                s = x/r;
                r = (r+s)/2;
                err = Math.abs(x - r*r);
            } while (err > EPSILON);
            System.out.print(Format.floating(5,1,x));
            System.out.print("\t");
            System.out.print(Format.floating(10,5,r));
        }
    }
}
```

static final...
means that the variable
can never be changed

Here's the Output

1.0	1.0
2.0	1.41421
3.0	1.73205
4.0	2.0
5.0	2.23607
6.0	2.44949
7.0	2.64575
8.0	2.82843
9.0	3.0
10.0	3.16228
11.0	3.31662
12.0	3.4641
13.0	3.60555
14.0	3.74116
15.0	3.87298
16.0	4.0
17.0	4.12311
18.0	4.24264
19.0	4.3589



“A wonderful square root. Let’s hope it can be used for the good of mankind.”