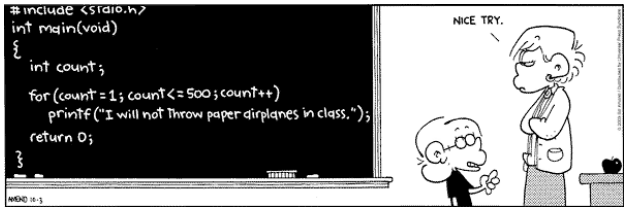
Lecture 3: Loops



Copyright 2004, FoxTrot by Bill Amend
www.ucomics.com/foxtrot/2003/10/03

The While Loop

While Loops

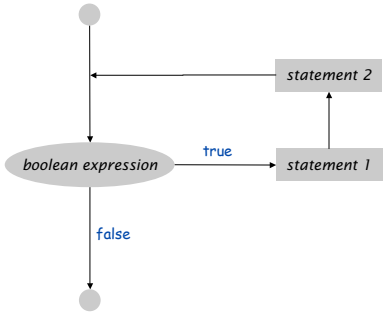
The `while` loop. A common repetition structure.

- Check a boolean expression.
- Execute a sequence of statements.
- Repeat.

loop continuation condition

```
while (boolean expression) {
   statement 1;
   statement 2;
}
```

loop body



While Loops: Powers of Two

Ex. Print first `n` powers of 2.

- Increment `i` from `1` to `n`.
- Double `v` each time.

```
int i = 0;
int v = 1;
while (i <= n) {
   System.out.println(v);
   i = i + 1;
   v = 2 * v;
}
```

i	v	i <= n
0	1	true
1	2	true
2	4	true
3	8	true
4	16	true
5	32	true
6	64	true
7	128	false

```
1
2
4
8
16
32
64
```

n = 6

Click for demo

Powers of Two

```
public class PowersOfTwo {
   public static void main(String[] args) {
      int n = Integer.parseInt(args[0]);
      int i = 0;
      int v = 1;
      while (i <= 6) {
         System.out.println(v);
         i = i + 1;
         v = 2 * v;
      }
   }
}
```

```
% java PowersOfTwo 4
1
2
4
8

% java PowersOfTwo 6
1
2
4
8
16
32
64
```


While Loops: Square Root

Q. How might we implement `Math.sqrt()` ?

A. To compute the square root of c:

- Initialize $t_0 = c$.
- Repeat until $t_i = c / t_i$, up to desired precision: set t_{i+1} to be the average of t_i and c / t_i.

t_0			=	2.0
t_1	=	$\frac{1}{2}(t_0 + \frac{2}{t_0})$	=	1.5
t_2	=	$\frac{1}{2}(t_1 + \frac{2}{t_1})$	=	1.416666666666665
t_3	=	$\frac{1}{2}(t_2 + \frac{2}{t_2})$	=	1.4142156862745097
t_4	=	$\frac{1}{2}(t_3 + \frac{2}{t_3})$	=	1.4142135623746899
t_5	=	$\frac{1}{2}(t_4 + \frac{2}{t_4})$	=	1.414213562373095

computing the square root of 2

While Loops: Square Root

Q. How might we implement `Math.sqrt()` ?

A. To compute the square root of c:

- Initialize $t_0 = c$.
- Repeat until $t_i = c / t_i$, up to desired precision: set t_{i+1} to be the average of t_i and c / t_i.

```
public class Sqrt {
   public static void main(String[] args) {
      double EPS = 1E-15;
      double c = Double.parseDouble(args[0]);
      double t = c;
      while (Math.abs(t - c/t) > t*EPS) {
         t = (c/t + t) / 2.0;
      }
      System.out.println(t);
   }
}
```

error tolerance

```
% java Sqrt 2.0
1.414213562373095
```

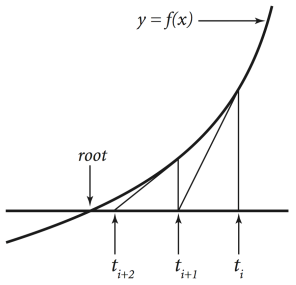
15 decimal digits of accuracy in 5 iterations

Newton-Raphson Method

Square root method explained.

- Goal: find root of function f(x).
- Start with estimate t_0.
- Draw line tangent to curve at $x = t_i$.
- Set t_{i+1} to be x-coordinate where line hits x-axis.
- Repeat until desired precision.

f(x) = x^2 - c to compute √c



The For Loop

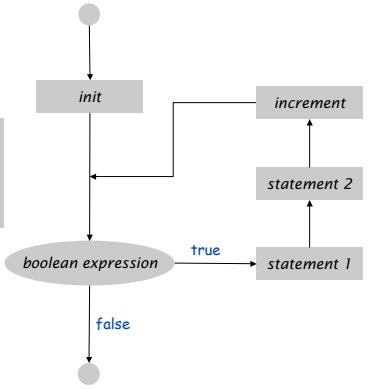
The `for` loop. Another common repetition structure.

- Execute initialization statement.
- Check boolean expression.
- Execute sequence of statements.
- Execute increment statement.
- Repeat.

loop continuation condition

```
for (init; boolean expression; increment) {
   statement 1;
   statement 2;
}
```

body



Create subdivision of a ruler.

- Initialize `ruler` to empty string.
- For each value `i` from `1` to `N`:
  sandwich two copies of `ruler` on either side of `i`.

```
int N = 3;
String ruler = " ";
for (int i = 1; i <= N; i++) {
   ruler = ruler + i + ruler;
}
System.out.println(ruler);
```

i	ruler
	" "
1	" 1 "
2	" 1 2 1 "
3	" 1 2 1 3 1 2 1 "


```
% java Ruler 1
 1

% java Ruler 2
 1 2 1

% java Ruler 3
 1 2 1 3 1 2 1

% java Ruler 4
 1 2 1 3 1 2 1 4 1 2 1 3 1 2 1

% java Ruler 5
 1 2 1 3 1 2 1 4 1 2 1 3 1 2 1 5 1 2 1 3 1 2 1 4 1 2 1 3 1 2 1

% java Ruler 100
Exception in thread "main"
java.lang.OutOfMemoryError
```

Observation. Loops can produce a huge amount of output!

Nesting

Nesting Conditionals and Loops

Conditionals enable you to do one of 2^n sequences of operations with n lines.

```
if (a0 > 0) System.out.print(0);
if (a1 > 0) System.out.print(1);
if (a2 > 0) System.out.print(2);
if (a3 > 0) System.out.print(3);
if (a4 > 0) System.out.print(4);
if (a5 > 0) System.out.print(5);
if (a6 > 0) System.out.print(6);
if (a7 > 0) System.out.print(7);
if (a8 > 0) System.out.print(8);
if (a9 > 0) System.out.print(9);
```

2^{10} = 1024 possible results, depending on input

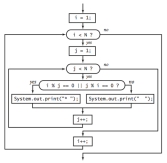
Loops enable you to do an operation n times using only 2 lines of code.

```
double sum = 0.0;
for (int i = 1; i <= 1024; i++)
   sum = sum + 1.0 / i;
```

computes 1/1 + 1/2 + ... + 1/1024

More sophisticated programs.

- Nest conditionals within conditionals.
- Nest loops within loops.
- Nest conditionals within loops within loops.



Nested If-Else

Ex. Pay a certain tax rate depending on income level.

Income	Rate
0 - 47,450	22%
47,450 - 114,650	25%
114,650 - 174,700	28%
174,700 - 311,950	33%
311,950 -	35%

```
double rate = 0.35;
if (income <  47450) rate = 0.22;
if (income < 114650) rate = 0.25;
if (income < 174700) rate = 0.28;
if (income < 311950) rate = 0.33;
```

wrong graduated income tax calculation

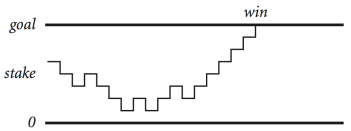
Gambler's Ruin

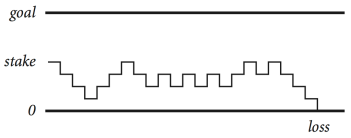
Gambler's ruin. Gambler starts with $stake and places $1 fair bets until going broke or reaching $goal.

- What are the chances of winning?
- How many bets will it take?

One approach. Monte Carlo simulation.

- Flip digital coins and see what happens.
- Repeat and compute statistics.





Gambler's Ruin

```
public class Gambler {
   public static void main(String[] args) {
      int stake  = Integer.parseInt(args[0]);
      int goal   = Integer.parseInt(args[1]);
      int trials = Integer.parseInt(args[2]);
      int wins   = 0;
      // repeat experiment N times
      for (int i = 0; i < trials; i++) {
         // do one gambler's ruin experiment
         int t = stake;
         while (t > 0 && t < goal) {
            // flip coin and update
            if (Math.random() < 0.5) t++;
            else                     t--;
         }
         if (t == goal) wins++;
      }
      System.out.println(wins + " wins of " + trials);
   }
}
```


Simulation and Analysis

stake goal trials

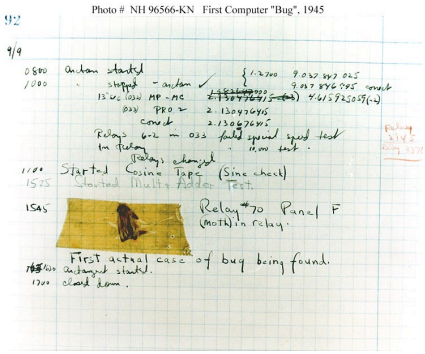
```
% java Gambler 5 25 1000
191 wins of 1000

% java Gambler 5 25 1000
203 wins of 1000

% java Gambler 500 2500 1000
197 wins of 1000
```

after a few hours of computing....

Fact. Probability of winning = stake ÷ goal.
Fact. Expected number of bets = stake × desired gain.
Ex. 20% chance of turning $500 into $2500, but expect to make one million $1 bets.

Remark. Both facts can be proved mathematically; for more complex scenarios, computer simulation is often the best plan of attack.

Debugging



http://www.history.navy.mil/photos/images/h96000/h96566kc.htm

Admiral Grace Murray Hopper

Debugging a Program

Factor. Given an integer N, compute its prime factorization.

3,757,208 = $2^3 \times 7 \times 13^2 \times 397$

i	N	output
2	3757208	2 2 2
3	469651	
4	469651	
5	469651	
6	469651	
7	469651	7
8	67093	
9	67093	
10	67093	
11	67093	
12	67093	
13	67093	13 13
14	397	
15	397	
16	397	
17	397	
18	397	
19	397	
20	397	
		397

3757208/8

Application. Break RSA cryptosystem.

Debugging a Program: Syntax Errors

Syntax error. Illegal Java program.
- Compiler error messages help locate problem.
- Eventually, a file named `Factors.class`.

Check if i is a factor.

```
public class Factors1 {
   public static void main(String[] args) {
      long N = Long.parseLong(args[0])
      for (i = 0; i < N; i++) {
         while (N % i == 0)
            System.out.print(i + " ")
            N = N / i
      }
   }
}
```

As long as i is a factor, divide it out.

Compile-time error

Debugging a Program: Semantic Errors

Semantic error. Legal but wrong Java program.
- Use "`System.out.println`" method to identify problem.

Check if i is a factor.

```
public class Factors2 {
   public static void main(String[] args) {
      long N = Long.parseLong(args[0]);
      for (long i = 2; i < N; i++) {
         while (N % i == 0)
            System.out.print(i + " ");
            N = N / i;
      }
   }
}
```

As long as i is a factor, divide it out.

no output (17) or infinite loop (49)

Run-time error

Debugging a Program: Performance Errors

Performance error. Correct program but too slow.
- Use profiling to discover bottleneck.
- Devise better algorithm.

Check if i is a factor.

```
public class Factors3 {
   public static void main(String[] args) {
      long N = Long.parseLong(args[0]);
      for (long i = 2; i <= N; i++) {
         while (N % i == 0) {
            System.out.print(i + " ");
            N = N / i;
         }
      }
   }
}
```

As long as i is a factor, divide it out.

too slow for large N (999,999,937)

Performance error

Debugging a Program: Success

Fact. If N has a factor, it has one less than or equal to its square root.
Impact. Many fewer iterations of `for` loop.

Check if i is a factor.

```
public class Factors {
   public static void main(String[] args) {
      long N = Long.parseLong(args[0]);
      for (long i = 2; i*i <= N; i++) {
         while (N % i == 0) {
            System.out.print(i + " ");
            N = N / i;
         }
      }
      if (N > 1) System.out.println(N);
      else       System.out.println();
   }
}
```

As long as i is a factor, divide it out.

Corner case: biggest factor occurs once.

Debugging a Program: Analysis

Q. How large an integer can I factor?

```
% java Factors 168
2 2 2 3 7

% java Factors 3757208
2 2 2 7 13 13 397

% java Factors 9201111169755555703
9201111169755555703
```

after a few minutes of computing....

largest factor

Digits	(i <= N)	(i*i <= N)
3	instant	instant
6	0.15 seconds	instant
9	77 seconds	instant
12	21 hours †	0.16 seconds
15	2.4 years †	2.7 seconds
18	2.4 millennia †	92 seconds

† estimated

Programming in Java

Programming in Java. [a slightly more realistic view]

1. Create the program.
2. Compile it..
   Compiler says: That's not a legal program.
   Back to step 1 to fix your errors of syntax.
3. Execute it.
   Result is bizarrely (or subtly) wrong.
   Back to step 1 to fix your errors of semantics.
4. Enjoy the satisfaction of a working program!

Debugging a Program

Debugging. Cyclic process of editing, compiling, and fixing errors.

- Always a logical explanation.
- What would the machine do?
- Explain it to the teddy bear.

You will make many mistakes as you write programs. It's normal.

> As soon as we started programming, we found out to our surprise that it wasn't as easy to get programs right as we had thought. I can remember the exact instant when I realized that a large part of my life from then on was going to be spent in finding mistakes in my own programs. - Maurice Wilkes

> If I had 8 hours to chop down a tree, I would spend 6 hours sharpening an axe. - Anonymous

Control Flow Summary

Control flow.

- Sequence of statements that are actually executed in a program.
- Conditionals and loops: enables us to choreograph the control flow.

Control Flow	Description	Examples
Straight-line programs	All statements are executed in the order given.	
Conditionals	Certain statements are executed depending on the values of certain variables.	`if` `if-else`

Loops	Certain statements are executed repeatedly until certain conditions are met.	`while`
		`for`
		`do-while`