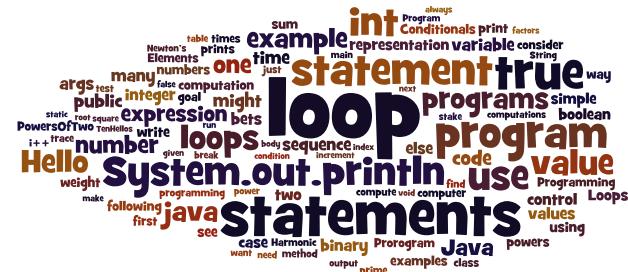


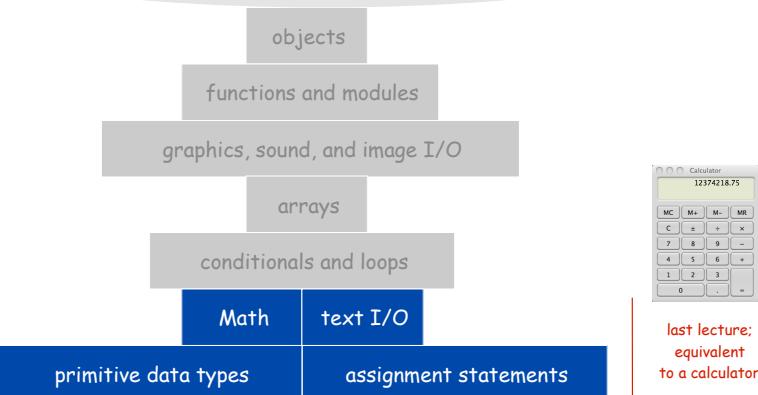
1.3 Conditionals and Loops



Introduction to Programming in Java: An Interdisciplinary Approach · Robert Sedgewick and Kevin Wayne · Copyright © 2008 · February 04, 2008 10:00 AM

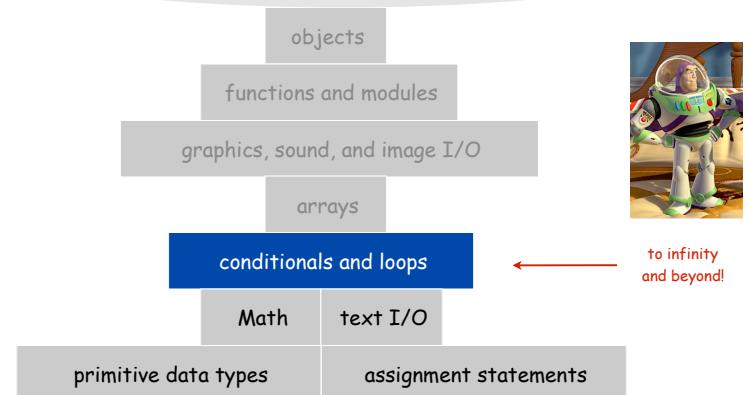
A Foundation for Programming

any program you might want to write



A Foundation for Programming

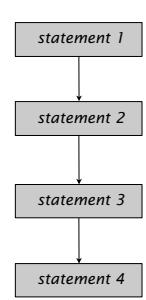
any program you might want to write



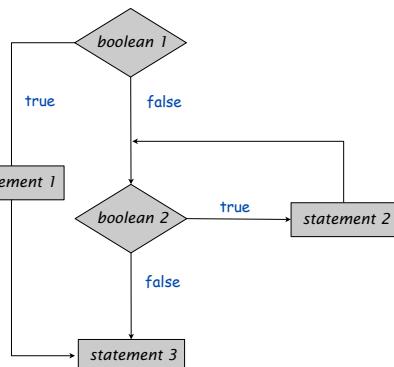
Conditionals and Loops

Control flow.

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



straight-line control flow



control flow with conditionals and loops

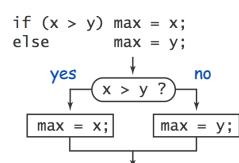
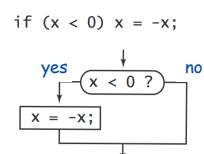
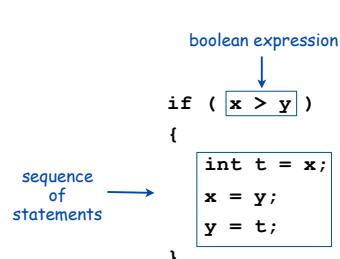
Conditionals



If Statement

The **if statement**. A common branching structure.

- Evaluate a boolean expression.
- If true, execute some statements.
- else option:** If false, execute other statements.



If Statement

Ex. Take different action depending on value of variable.

```

public class Flip
{
    public static void main(String[] args)
    {
        if (Math.random() < 0.5)
            System.out.println("Heads");
        else System.out.println("Tails");
    }
}
  
```

% java Flip
Heads
% java Flip
Heads
% java Flip
Tails
% java Flip
Heads

5

7

8

If Statement Examples

```
if (x > 0) x = -x;
```

absolute value

```
if (x > y) max = x;
else max = y;
```

maximum

x > y before		
x	y	t
1234	99	undefined
1234	99	1234
99	99	1234
99	1234	1234

x < y after		
x	y	t
1234	99	1234
99	99	1234
99	1234	1234

```
if (den == 0) System.out.println("Division by zero");
else System.out.println("Quotient = " + num/den);
```

error check for division operation

```
double discriminant = b*b - 4.0*c;
if (discriminant < 0.0)
{
    System.out.println("No real roots");
}
else
{
    System.out.println((-b + Math.sqrt(discriminant))/2.0);
    System.out.println((-b - Math.sqrt(discriminant))/2.0);
}
```

error check for quadratic formula

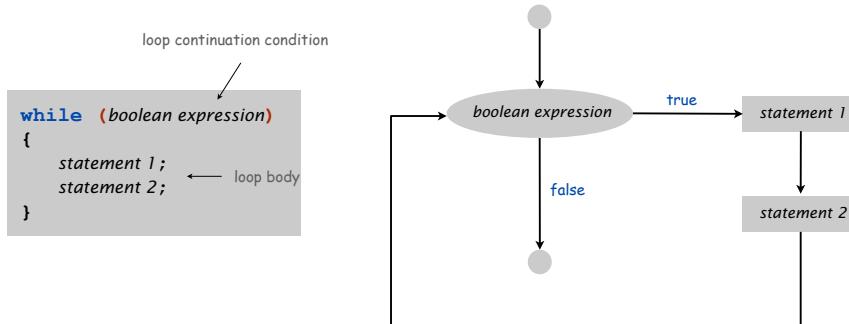
Loops



While Loop

The **while** loop. A common repetition structure.

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



While Loop Example: Powers of Two

Ex. Print powers of 2 that are $\leq 2^n$.

- Increment **i** from 0 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

9

10

11

12

Powers of Two (full program)

```
public class PowersOfTwo {
    public static void main(String[] args) {
        // last power of two to print
        int n = Integer.parseInt(args[0]);

        int i = 0; // loop control counter
        int v = 1; // current power of two
        while (i <= n) {
            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
```

13

TEQ on While Loops

Anything wrong with the following code?

```
public class PowersOfTwo {
    public static void main(String[] args) {
        int N = Integer.parseInt(args[0]);
        int i = 0; // loop control counter
        int v = 1; // current power of two
        while (i <= N)
            System.out.println(v);
            i = i + 1;
            v = 2 * v;
    }
}
```

14

While Loop Example: Square Root

Goal. Implement `Math.sqrt()`.

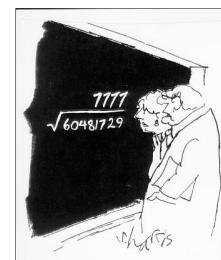
```
% java Sqrt 60481729
7777.0
```

Newton-Raphson method to compute the square root of c :

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

i	t_i	$2/t_i$	average
0	2.0	1.0	1.5
1	1.5	1.3333333	1.4166667
2	1.4166667	1.4117647	1.4142157
3	1.4142157	1.4142114	1.4142136
4	1.4142136	1.4142136	

computing the square root of 2 to seven places



"A wonderful square root. Let's hope it can be used for the good of mankind."

Copyright 2004, Sidney Harris
<http://www.sciencecartoonsplus.com>

15

While Loop Example: Square Root

Goal. Implement `Math.sqrt()`.

Newton-Raphson method to compute the square root of c :

- Initialize $t_0 = c$.
- Repeat until $t_i = c / t_{i-1}$, 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; } error tolerance
        System.out.println(t);
    }
}
```

```
% java Sqrt 2.0
1.414213562373095
```

15 decimal digits of accuracy in 5 iterations

16

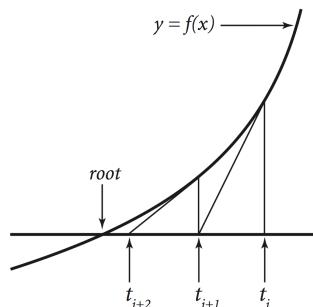
Newton-Raphson Method

Square root method explained (some math omitted).

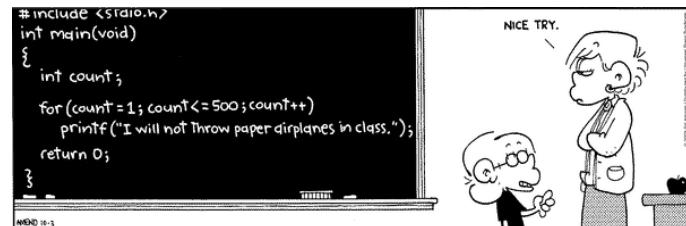
- Goal: find root of function $f(x)$.
- Start with estimate t_0 .

$f(x) = x^2 - c$ to compute \sqrt{c}

- 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.



The For Loop



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

17

18

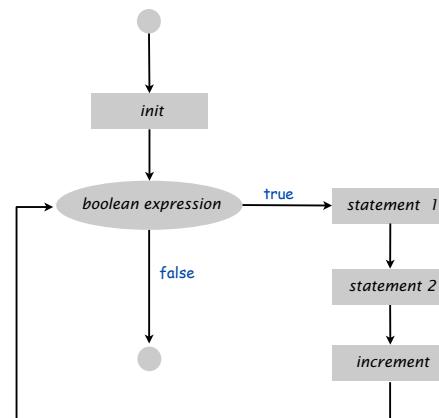
The For Loop

The **for** loop. Another common repetition structure.

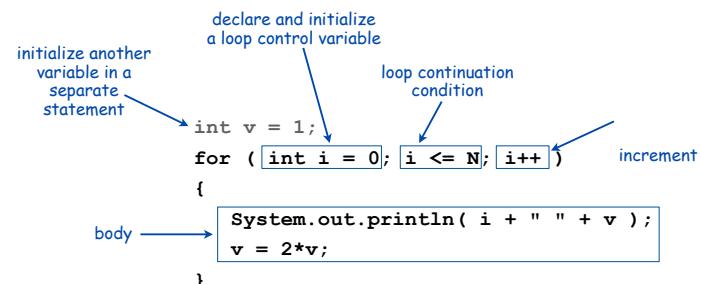
- Execute initialization statement.
- Check boolean expression.
- Execute sequence of statements.
- Execute increment statement.
- Repeat.

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

loop continuation condition



Anatomy of a for Loop



19

20

Anatomy of a for Loop

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

Every for loop has an equivalent while loop

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

v	i	output
1	0	
1	0	0 1
2	0	
2	1	1 2
4	1	
4	2	2 4
8	2	
8	3	3 8

Why for loops? Can provide more compact and understandable code.

21

For Loops: Subdivisions of a Ruler

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`.

```
public class Ruler
{
    public static void main(String[] args)
    {
        int N = Integer.parseInt(args[0]);
        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 "

end-of-loop trace

For Loops: Subdivisions of a Ruler

```
% 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 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
```

2¹⁰⁰ - 1 ≈ 1,267,650,600,228,229,401,496,703,205,375 integers in output

Observation. Loops can produce a huge amount of output!

23

Loop Examples

```
int sum = 0;
for (int i = 1; i <= N; i++)
    sum += i;
System.out.println(sum);
```

compute sum (1 + 2 + 3 + ... + N)

```
int sum = 0;
for (int i = 1; i <= N; i++)
    product *= i;
System.out.println(product);
```

compute N! (1 * 2 * 3 * ... * N)

```
for (int i = 0; i <= N; i++)
    System.out.println(i + " " + 2*Math.PI*i/N);
```

print a table of function values

```
int v = 1;
while (v <= N/2)
    v = 2*v;
System.out.println(v);
```

print largest power of 2 less than or equal to N

24

TEQ on For Loops

[easy if you read Exercise 1.3.13]

What does the following program print?

```
public class Mystery
{
    public static void main(String[] args)
    {
        int f = 0, g = 1;
        for (int i = 0; i <= 10; i++)
        {
            System.out.println(f);
            f = f + g;
            g = f - g;
        }
    }
}
```

Nesting



25

26

Nesting Conditionals and Loops

Nesting. Use a conditional or a loop within a conditional or a loop

- Enables complex control flows.
- Adds to challenge of debugging.

Any "statement" within a conditional or loop
may itself be a conditional or a loop statement

```
for (int i = 0; i < trials; i++)
{
    int t = stake;
    while (t > 0 && t < goal)
        if (Math.random() < 0.5) t++;
        else t--;
    if (t == goal) wins++;
}
```

if-else statement
within a while loop
within a for loop

Nested If Statements

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%

5 mutually exclusive
alternatives

27

28

Nested If Statements

Use **nested** if statements to handle multiple alternatives

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

Nested If-Else Statements

Need all those braces? Not always:

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

is shorthand for

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

but BE CAREFUL when nesting if-else statements (see Q&A p. 75).

29

30

TEQ on If-Else

Anything wrong with the following code?

```
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;
```

Nesting Example: Gambler's Ruin

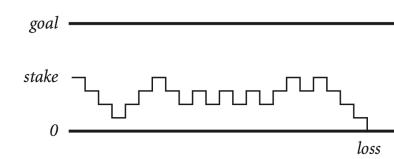
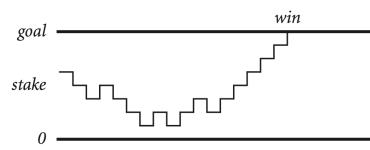
Gambler's ruin. Gambler starts with \$stake and places \$1 fair bets until going broke or reaching \$goal.

- 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.



31

32

Nesting Example: Gambler's Ruin Simulation

```
public class Gambler
{
    public static void main(String[] args)
    {
        // Get parameters from command line.
        int stake = Integer.parseInt(args[0]);
        int goal = Integer.parseInt(args[1]);
        int trials = Integer.parseInt(args[2]);

        // Count wins among args[2] trials.
        int wins = 0;
        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);
    }
}
```

if statement
within a while loop
within a for loop

```
% 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 substantial wait....

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.

$$500/2500 = 20\%$$

$$500 * (2500 - 500) = 1\,000\,000$$

Remark. Both facts can be proved mathematically

For more complex scenarios, computer simulation is often the best plan of attack.



34

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

Admiral Grace Murray Hopper

Photo # NH 98566-KN First Computer "Bug", 1945
 9/9
 1/500 Auton. started - auton ✓ { 1/500 1000 2000 3000 4000 5000 6000 7000 8000 9000 10000
 15000 16000 17000 18000 19000 20000 21000 22000 23000 24000 25000 26000 27000 28000 29000 30000 31000 32000 33000 34000 35000 36000 37000 38000 39000 40000 41000 42000 43000 44000 45000 46000 47000 48000 49000 50000 51000 52000 53000 54000 55000 56000 57000 58000 59000 60000 61000 62000 63000 64000 65000 66000 67000 68000 69000 70000 71000 72000 73000 74000 75000 76000 77000 78000 79000 80000 81000 82000 83000 84000 85000 86000 87000 88000 89000 90000 91000 92000 93000 94000 95000 96000 97000 98000 99000 100000
 100000 101000 102000 103000 104000 105000 106000 107000 108000 109000 110000 111000 112000 113000 114000 115000 116000 117000 118000 119000 120000 121000 122000 123000 124000 125000 126000 127000 128000 129000 130000 131000 132000 133000 134000 135000 136000 137000 138000 139000 140000 141000 142000 143000 144000 145000 146000 147000 148000 149000 150000 151000 152000 153000 154000 155000 156000 157000 158000 159000 160000 161000 162000 163000 164000 165000 166000 167000 168000 169000 170000 171000 172000 173000 174000 175000 176000 177000 178000 179000 180000 181000 182000 183000 184000 185000 186000 187000 188000 189000 190000 191000 192000 193000 194000 195000 196000 197000 198000 199000 200000
 200000 201000 202000 203000 204000 205000 206000 207000 208000 209000 210000 211000 212000 213000 214000 215000 216000 217000 218000 219000 220000 221000 222000 223000 224000 225000 226000 227000 228000 229000 230000 231000 232000 233000 234000 235000 236000 237000 238000 239000 240000 241000 242000 243000 244000 245000 246000 247000 248000 249000 250000 251000 252000 253000 254000 255000 256000 257000 258000 259000 260000 261000 262000 263000 264000 265000 266000 267000 268000 269000 270000 271000 272000 273000 274000 275000 276000 277000 278000 279000 280000 281000 282000 283000 284000 285000 286000 287000 288000 289000 290000 291000 292000 293000 294000 295000 296000 297000 298000 299000 300000
 300000 301000 302000 303000 304000 305000 306000 307000 308000 309000 310000 311000 312000 313000 314000 315000 316000 317000 318000 319000 320000 321000 322000 323000 324000 325000 326000 327000 328000 329000 330000 331000 332000 333000 334000 335000 336000 337000 338000 339000 340000 341000 342000 343000 344000 345000 346000 347000 348000 349000 350000 351000 352000 353000 354000 355000 356000 357000 358000 359000 360000 361000 362000 363000 364000 365000 366000 367000 368000 369000 370000 371000 372000 373000 374000 375000 376000 377000 378000 379000 380000 381000 382000 383000 384000 385000 386000 387000 388000 389000 390000 391000 392000 393000 394000 395000 396000 397000 398000 399000 400000
 400000 401000 402000 403000 404000 405000 406000 407000 408000 409000 410000 411000 412000 413000 414000 415000 416000 417000 418000 419000 420000 421000 422000 423000 424000 425000 426000 427000 428000 429000 430000 431000 432000 433000 434000 435000 436000 437000 438000 439000 440000 441000 442000 443000 444000 445000 446000 447000 448000 449000 450000 451000 452000 453000 454000 455000 456000 457000 458000 459000 460000 461000 462000 463000 464000 465000 466000 467000 468000 469000 470000 471000 472000 473000 474000 475000 476000 477000 478000 479000 480000 481000 482000 483000 484000 485000 486000 487000 488000 489000 490000 491000 492000 493000 494000 495000 496000 497000 498000 499000 500000
 500000 501000 502000 503000 504000 505000 506000 507000 508000 509000 510000 511000 512000 513000 514000 515000 516000 517000 518000 519000 520000 521000 522000 523000 524000 525000 526000 527000 528000 529000 530000 531000 532000 533000 534000 535000 536000 537000 538000 539000 540000 541000 542000 543000 544000 545000 546000 547000 548000 549000 550000 551000 552000 553000 554000 555000 556000 557000 558000 559000 560000 561000 562000 563000 564000 565000 566000 567000 568000 569000 570000 571000 572000 573000 574000 575000 576000 577000 578000 579000 580000 581000 582000 583000 584000 585000 586000 587000 588000 589000 589000 590000 591000 592000 593000 594000 595000 596000 597000 598000 599000 600000
 600000 601000 602000 603000 604000 605000 606000 607000 608000 609000 610000 611000 612000 613000 614000 615000 616000 617000 618000 619000 620000 621000 622000 623000 624000 625000 626000 627000 628000 629000 630000 631000 632000 633000 634000 635000 636000 637000 638000 639000 640000 641000 642000 643000 644000 645000 646000 647000 648000 649000 650000 651000 652000 653000 654000 655000 656000 657000 658000 659000 660000 661000 662000 663000 664000 665000 666000 667000 668000 669000 670000 671000 672000 673000 674000 675000 676000 677000 678000 679000 680000 681000 682000 683000 684000 685000 686000 687000 688000 689000 689000 690000 691000 692000 693000 694000 695000 696000 697000 698000 699000 700000
 700000 701000 702000 703000 704000 705000 706000 707000 708000 709000 710000 711000 712000 713000 714000 715000 716000 717000 718000 719000 720000 721000 722000 723000 724000 725000 726000 727000 728000 729000 730000 731000 732000 733000 734000 735000 736000 737000 738000 739000 740000 741000 742000 743000 744000 745000 746000 747000 748000 749000 750000 751000 752000 753000 754000 755000 756000 757000 758000 759000 760000 761000 762000 763000 764000 765000 766000 767000 768000 769000 769000 770000 771000 772000 773000 774000 775000 776000 777000 778000 779000 779000 780000 781000 782000 783000 784000 785000 786000 787000 788000 789000 789000 790000 791000 792000 793000 794000 795000 796000 797000 798000 799000 800000
 800000 801000 802000 803000 804000 805000 806000 807000 808000 809000 810000 811000 812000 813000 814000 815000 816000 817000 818000 819000 820000 821000 822000 823000 824000 825000 826000 827000 828000 829000 830000 831000 832000 833000 834000 835000 836000 837000 838000 839000 840000 841000 842000 843000 844000 845000 846000 847000 848000 849000 850000 851000 852000 853000 854000 855000 856000 857000 858000 859000 859000 860000 861000 862000 863000 864000 865000 866000 867000 868000 869000 869000 870000 871000 872000 873000 874000 875000 876000 877000 878000 879000 879000 880000 881000 882000 883000 884000 885000 886000 887000 888000 889000 889000 890000 891000 892000 893000 894000 895000 896000 897000 898000 899000 900000
 900000 901000 902000 903000 904000 905000 906000 907000 908000 909000 910000 911000 912000 913000 914000 915000 916000 917000 918000 919000 920000 921000 922000 923000 924000 925000 926000 927000 928000 929000 930000 931000 932000 933000 934000 935000 936000 937000 938000 939000 940000 941000 942000 943000 944000 945000 946000 947000 948000 949000 950000 951000 952000 953000 954000 955000 956000 957000 958000 959000 959000 960000 961000 962000 963000 964000 965000 966000 967000 968000 969000 969000 970000 971000 972000 973000 974000 975000 976000 977000 978000 979000 979000 980000 981000 982000 983000 984000 985000 986000 987000 988000 989000 989000 990000 991000 992000 993000 994000 995000 996000 997000 998000 999000 1000000

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

99% of program development

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.

Good news: Can use computer to test program.

Bad news: Conditionals/loops open up huge number of possibilities.

Really bad news: Cannot use computer to automatically find all bugs. ↗ stay tuned

Debugging Example

Factor. Given an integer $N > 1$, compute its prime factorization.

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

$$98 = 2 \times 7^2$$

$$17 = 17$$

Note: 1 is not prime.
(else it would have to
be in every
factorization)

$$11,111,111,111,111 = 2,071,723 \times 5,363,222,357$$

Debugging: 99% of Program Development

Programming. A process of finding and fixing mistakes.

- Compiler error messages help locate syntax errors.
- Run program to find semantic and performance errors.

```
public class Factors
{
    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;
        }
    }
}
```

Check whether
i is a factor.

if i is a factor
print it and
divide it out

This program has bugs!



39

Debugging: Syntax Errors

Syntax error. Illegal Java program.

- Compiler error messages help locate problem.
- Goal: no errors and a file named `Factors.class`.

```
public class Factors
{
    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;
        }
    }
}
```



40

Debugging: Syntax Errors

Syntax error. Illegal Java program.

- Compiler error messages help locate problem.
- Goal: no errors and a file named `Factors.class`.

```
public class Factors
{
    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
        }
    }
}
```

% javac Factors.java
Factors.java:6: ';' expected
 for (i = 2; i < N; i++)
 ^
1 error ← the FIRST error



41

Debugging: Syntax Errors

Syntax error. Illegal Java program.

- Compiler error messages help locate problem.
- Goal: no errors and a file named `Factors.class`.

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

Syntax (compile-time) errors



42

Debugging: Semantic Errors

Semantic error. Legal but wrong Java program.

- Run program to identify problem.
- Add print statements if needed to produce **trace**.

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

% javac Factors.java
% java Factors ← oops, need argument
Exception in thread "main"
java.lang.ArrayIndexOutOfBoundsException: 0
at Factors.main(Factors.java:5)



43

Debugging: Semantic Errors

Semantic error. Legal but wrong Java program.

- Run program to identify problem.
- Add print statements if needed.

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

% javac Factors.java
% % java Factors 98
Exception in thread "main"
java.lang.ArithmaticException: / by zero
at Factors.main(Factors.java:8)



44

Debugging: Semantic Errors

Semantic error. Legal but wrong Java program.

- Run program to identify problem.
 - Add print statements if needed.

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

need to start at 2 since
0 and 1 cannot be factors

need to start at 2 since
0 and 1 cannot be factors



45

Debugging: Semantic Errors

Semantic error. Legal but wrong Java program.

- Run program to identify problem
 - Add print statements if needed.



46

Debugging: Semantic Errors

Semantic error. Legal but wrong Java program.

- Run program to identify problem.
 - Add print statements if needed.

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

Semantic (run-time) error:
indents do not imply braces



47

Debugging: The Beat Goes On

Success? Program factors $98 = 2 \cdot 2 \cdot 7$

- Time to try it for other inputs.
 - Add **trace** to find and fix (minor) problems.

```
public class Factors
{
    public static void main(String[] args
    {
        long N = Long.parseLong(args[0])
        for (int i = 2; i < N; i++)
        {
            // Check whether i is a factor.
            while (N % i == 0)
            {
                // If so, print and divide.
                System.out.print(i + " ");
                N = N / i;
            }
        }
    }
}
```

```
% java Factors 98  
2 7 7 % ← need newline  
% java Factors 5  
% ← ??? no output  
% java Factors 6  
2 % ← ??? where's the 3?
```



48

Debugging: The Beat Goes On

Success? Program factors $98 = 2 \cdot 2 \cdot 7$.

- Time to try it for other inputs.
- Add `trace` to find and fix (minor) problems.

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

% javac Factors.java
% java Factors 5
TRACE 2 5
TRACE 3 5
TRACE 4 5
% java Factors 6
2
TRACE 2 3

AHA!
Print out N
after for loop
(if it is not 1)



49

Debugging: Success?

Success? Program seems to work.

- Remove trace to try larger inputs.
- [stay tuned].

```
public class Factors
{
    public static void main(String[] args)
    {
        long N = Long.parseLong(args[0])
        for (int i = 2; i < N; i++)
        {
            // Check whether i is a factor.
            while (N % i == 0)
            {
                // If so, print and divide.
                System.out.print(i + " ");
                N = N / i;
            }
            if (N > 1) System.out.println(N);
            else System.out.println();
        }
    }
}
```

Corner case:
print largest
factor
(and new line)

```
???  
%$%@#$!!  
forgot to recompile  
% java Factors 5  
TRACE 2 5  
TRACE 3 5  
TRACE 4 5  
% javac Factors.java  
% java Factors 5  
5  
% java Factors 6  
2 3  
% java Factors 98  
2 7 7  
% java Factors 3757208  
2 2 2 7 13 13 397
```

Time to add comments
(if not earlier).

50

Debugging Your Program

[Debugging Your Program. \[summary\]](#)

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!

[but stay tuned for more debugging]