3. Conditionals and loops

Context: basic building blocks for programming

Conditionals and Loops

Control flow

- The sequence of statements that are actually executed in a program.
- **Conditionals and loops** enable us to choreograph control flow.
The if statement

Execute certain statements depending on the values of certain variables.
• Evaluate a boolean expression.
• If true, execute a statement.
• The else option: if false, execute a different statement.

Example: if (x < 0) x = -x;

Computes the absolute value of x

Example: if (x > y) max = x; else max = y;

Computes the maximum of x and y

Example of if statement use: simulate a coin flip

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

Pop quiz on if statements

Q. What does this program do?

A. Reads two integers from the command line, then prints them out in numerical order.
Example of if statement use: error checks

```java
public class IntOps {
    public static void main(String[] args) {
        int a = Integer.parseInt(args[0]);
        int b = Integer.parseInt(args[1]);
        int sum = a + b;
        int prod = a * b;
        System.out.println(a + " + " + b + " = " + sum);
        System.out.println(a + " * " + b + " = " + prod);
        if (b == 0) System.out.println("Division by zero");
        else System.out.println(a + " / " + b + " = " + a / b);
    }
}
```

Good programming practice. Use conditionals to check for and avoid runtime errors.

The while loop

Execute certain statements repeatedly until certain conditions are met.
- Evaluate a boolean expression.
- If true, execute a sequence of statements.
- Repeat.

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

Prints the powers of two from \(2^0\) to \(2^n\).
[stay tuned for a trace]

Example of while loop use: print powers of two

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

Prints the powers of two from \(2^0\) to \(2^n\).
Pop quiz on while loops

Q. Anything wrong with the following code?

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

Example of while loop use: implement Math.sqrt()

Newton-Raphson method to compute √c
• Initialize t₀ = c.
• Repeat until tᵢ = c/tᵢ (up to desired precision):
  Set tᵢ₊₁ to be the average of tᵢ and c / tᵢ.

<table>
<thead>
<tr>
<th>i</th>
<th>tᵢ</th>
<th>2/tᵢ</th>
<th>average</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2.0</td>
<td>1.0</td>
<td>1.5</td>
</tr>
<tr>
<td>1</td>
<td>1.5</td>
<td>1.3333333</td>
<td>1.4166667</td>
</tr>
<tr>
<td>2</td>
<td>1.4166667</td>
<td>1.4117647</td>
<td>1.4142157</td>
</tr>
<tr>
<td>3</td>
<td>1.4142157</td>
<td>1.4142114</td>
<td>1.4142136</td>
</tr>
<tr>
<td>4</td>
<td>1.4142136</td>
<td>1.4142136</td>
<td>computing the square root of 2 to seven places</td>
</tr>
</tbody>
</table>

Example of while loop use: implement Math.sqrt()

```java
public class Sqrt
{
    public static void main(String[] args)
    {
        double EPS = 1E-15; // error tolerance (15 places)
        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);
    }
}
```

Newton-Raphson method

Explanation (some math omitted)
• Goal: find root of function f(x).
• Start with estimate t₀.
• Draw line tangent to curve at x = tᵢ.
• Set tᵢ₊₁ to be x-coordinate where line hits x-axis.
• Repeat until desired precision.

Many students actually look forward to Mr. Atwood's math tests.
3. Conditionals & Loops

- Conditionals: the if statement
- Loops: the while statement
- An alternative: the for loop
- Nesting
- Debugging

The for loop

An alternative repetition structure.

- Evaluate an initialization statement.
- Evaluate a boolean expression.
- If true, execute a sequence of statements, then execute an increment statement.
- Repeat.

Example:

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

Prints the powers of two from 2^0 to 2^n

Example of for loop use: subdivisions of a ruler

Create subdivisions of a ruler to 1/N inches.
- Initialize ruler to one space.
- For each value i from 1 to n:
  sandwich i between two copies of ruler.

```java
public class Ruler {
    public static void main(String[] args) {
        int N = Integer.parseInt(args[0]);
        String ruler = "\n        for (int i = 1; i <= N; i++)
            ruler += i + + i + ruler;
        System.out.println(ruler);
    }
}
```

Note: Small program can produce huge amount of output.
1. What does the following program print?

```java
public class For {
    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;
        }
    }
}
```

2. Nesting conditionals and loops

Nesting
- Any ‘statement’ within a conditional or loop may itself be a conditional or a loop statement.
- Enables complex control flows.
- Adds to challenge of debugging.

Example:
```java
for (int i = 0; i < trials; i++)
{
    int t = stake;
    while (t > 0 && t < goal)
    {
        if (Math.random() < 0.5) t++; // if-else statements
        else t--;
        if (t == goal) wins++; // nested if-else statements
    }
} [Stay tuned for an explanation of this code.]
```
Example of nesting conditionals: Tax rate calculation

**Goal.** Given income, calculate proper tax rate.

<table>
<thead>
<tr>
<th>Income</th>
<th>Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 - 47450</td>
<td>22%</td>
</tr>
<tr>
<td>$47,450 - $114,649</td>
<td>25%</td>
</tr>
<tr>
<td>$114,650 - $174,699</td>
<td>28%</td>
</tr>
<tr>
<td>$174,700 - $311,949</td>
<td>33%</td>
</tr>
<tr>
<td>$311,950 +</td>
<td>35%</td>
</tr>
</tbody>
</table>

```java
public class PQifIf {
    public static void main(String[] args) {
        double income = Double.parseDouble(args[0]);
        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;
        System.out.println(rate);
    }
}
```

Pop quiz on nested if statements

Q. Anything wrong with the following code?

```java
public class PQifIf {
    public static void main(String[] args) {
        double income = Double.parseDouble(args[0]);
        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;
        System.out.println(rate);
    }
}
```

Gambler’s ruin problem

A gambler starts with $stake and places $1 fair bets.
- Outcome 1 (loss): Gambler goes broke with $0.
- Outcome 2 (win): Gambler reaches $goal.

Q. What are the chances of winning?
Q. How many bets until win or loss?

**One approach: Monte Carlo simulation.**
- Use a simulated coin flip.
- Repeat and compute statistics.

Example of nesting conditionals and loops: Simulate gambler’s ruin

```java
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;
        for (int t = 0; t < trials; t++) {
            int t = stake;
            while (t > 0 && t < goal) {
                if (Math.random() < 0.5) t++;
                else t--;
            }
            if (t == goal) wins++;
        }
        System.out.println(wins + " of " + trials);
    }
}
```
Digression: simulation and analysis

Facts (known via mathematical analysis for centuries)
• Probability of winning = stake / goal.
• Expected number of bets = stake × desired gain.

Example
• 20% chance of turning $500 into $2500.
  500/2500 = 20%
• Expect to make 1 million $1 bets.
  500/2500 × 500 = 1,000,000

Remarks
• Computer simulation can help validate mathematical analysis.
• For this problem, mathematical analysis is simpler (if you know the math).
• For more complicated variants, computer simulation may be the best plan of attack.

3. Conditionals & Loops
• Conditionals: the if statement
• Loops: the while statement
• An alternative: the for loop
• Nesting
• Debugging

Debugging
is 99% of program development in any programming language, even for experts.

Bug: A mistake in a program.
Debugging: The process of eliminating bugs.

"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

Impossible ideal: “Please compile, execute, and debug my program.”
Bottom line: Programming is primarily a process of finding and fixing mistakes.

Debugging
is challenging because conditionals and loops dramatically increase the number of possible outcomes.

<table>
<thead>
<tr>
<th>program structure</th>
<th>no loops</th>
<th>N conditionals</th>
<th>1 loop</th>
</tr>
</thead>
<tbody>
<tr>
<td>number of possible execution sequences</td>
<td>1</td>
<td>2^N</td>
<td>no limit</td>
</tr>
</tbody>
</table>

Most programs contain numerous conditionals and loops, with nesting.

Good news. Conditionals and loops provide structure that helps us understand our programs.

Old and low-level languages have a goto statement that provides arbitrary structure. Eliminating gos was controversial until Edgar Dijkstra published the famous note “Goto considered harmful” in 1968.

“The quality of programmers is a decreasing function of the number of goto statements in the programs they produce.”

— Edgar Dijkstra
Debugging a program: a running example

Problem: Factor a large integer \( N \).
Application: Cryptography.

Suprising fact: Security of internet commerce depends on difficulty of factoring large integers.

Method

• Consider each integer \( i \) less than \( N \).
• While \( i \) divides \( N \) evenly
  Print \( i \) (it is a factor of \( N \)).
  Replace \( N \) with \( N/i \).

Rationale:
1. Any factor of \( N/i \) is a factor of \( N \).
2. \( i \) may be a factor of \( N/i \).

Debugging a program: runtime and semantic errors

Does your legal Java program do what you want it to do?
• You need to run it to find out.
• Use java runtime to find the first error.
• Fix and repeat.

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

        if (N == 1)
            System.out.println("1 is a factor of N");
    }
}

Debugging a program: syntax errors

Is your program a legal Java program?
• Java compiler can help you find out.
• Use javac to find the first error.
• Repeat.
• Result: An executable Factors.class file

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

        if (N == 1)
            System.out.println("1 is a factor of N");
    }
}

Debugging a program: testing

Does your legal Java program always do what you want it to do?
• You need to test on many types of inputs to find out.
• Add trace code to find the first error.
• Fix the error.
• Repeat.

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

        if (N == 1)
            System.out.println("1 is a factor of N");
    }
}
Debugging a program: testing

Does your legal Java program always do what you want it to do?
- You need to test on many types of inputs it to find out.
- Add trace code to find the first error.
- Fix the error.
- Repeat.

```java
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 /= i;
            }
        }
        if (N > 1) System.out.println(N);
    }
}
```

Note: This working program still has a bug (try tuned).

Debugging a program: performance analysis

Q. How large an integer can I factor?

<table>
<thead>
<tr>
<th>digits in largest factor</th>
<th>i &lt; N</th>
<th>i &lt;= N/i</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>instant</td>
<td>instant</td>
</tr>
<tr>
<td>6</td>
<td>instant</td>
<td>instant</td>
</tr>
<tr>
<td>9</td>
<td>instant</td>
<td>instant</td>
</tr>
<tr>
<td>12</td>
<td>21 hours</td>
<td>instant</td>
</tr>
<tr>
<td>15</td>
<td>2.4 years</td>
<td>2.7 seconds</td>
</tr>
<tr>
<td>18</td>
<td>2.4 millennia</td>
<td>92 seconds</td>
</tr>
</tbody>
</table>

Lesson. Performance matters!

Note. Internet commerce is still secure: it depends on the difficulty of factoring 200-digit integers.

Debugging your program: summary

Program development is a four-step process, with feedback.

EDIT your program.

COMPILE your program to create an executable file.

RUN your program to test that it works as you imagined.

TEST your program on realistic and real input data.

SUBMIT your program for independent testing and approval.
3. Conditionals & Loops