ICS 111
Fundamentals of computation

- Sequence, Repetition, Conditionals, Parallel Execution, Program Structure.
- Arithmetic and Logical Operations.
- Data Types, Variables, and Memory.

Building Blocks of Computations

- Each of the things a computer can do is relatively simple
  - any educated human can add two numbers or verify whether a number is > 0 or turn a light on or off
  - that is the kind of things a computer can do
- The power of computers comes from doing many of these simple things:
  - quickly, and
  - very reliably
- How to put together these simple things to achieve useful goals?

Putting Together Simple Computations

- Sequence
  - do A, then do B
- Repetition
  - do A 100 times
- Conditionals
  - if A do B, otherwise (else) do C
- Parallel Execution
  - do A, B, and C, possibly at the same time
- Program Structure
  - organize the program so programmers can understand it better

Sequence of Operations

buy the ingredients for a cake

mix the ingredients together

let them rise

preheat the oven

put the ingredients in a cake pan

put the cake pan in the oven

carefully take the cake pan out of the oven

- no single step gives you a cake, but all of them together do

Repetition of Operations

```
while there are cherries left to eat
    eat the next cherry
    properly dispose of the pit
```

- this repeats a sequence of operations
- a repetition is generally known as a **loop**
- the number of loops is:
  - variable (different on different days)
  - fixed (always equal to the number of cherries)

Example of Looping

```
total = 0
foreach kind in { bills, coins }
    foreach unit in kind  // kind is bill or coin, unit is the bill or coin
        total = total + value of the unit
```

- loops can be **nested**
- an **assignment** (x = x + y) assigns a new value to a **variable** (x, total, unit, and kind are variables)t
  - the variable itself can appear in the **expression**
  - its value in the expression is the old value, before the assignment takes place

Conditional Execution

```
if my team wins
    collect money from my friend
else
    pay money to my friend
```

- the condition is **true** or **false**
  - something that is true or false is a **boolean**
- only one of the branches is executed
  - the **if branch** or the **else branch**
- the else part is optional

Example of Conditional Execution

```
wasNegative = false
if a > 0
    b = a
else if a < 0
    b = -a
    wasNegative = true
```

- the final value of b is the absolute value of a
  - but only if a is non-zero!
  - this may or may not be an error
- the wasNegative boolean variable keeps track of whether the value of a is less than 0
- if a is zero (the missing “else”), there are no assignments

Parallel Execution

- All of these primitives (sequence, repetition, conditional execution) were available to John Backus in the 1950s
- Since then, computers have evolved to have:
  - multiple processing cores
  - vector execution units
- Faster programs will take advantage of these hardware features
- Parallel execution is not part of 111, so this is just an introduction

Two Ways to do Parallel Execution

- Do different things, possibly at the same time
  - This is similar to a sequence, but the different things may be done at the same time
  - main mechanism: **threads**
- Do the same thing at the same time to different items of data
  - This is similar to a repetition, but again the different things may be done at the same time
  - main mechanisms: **vector processing**, **map/reduce**

Parallel Execution Example

- 10 delivery people can deliver 20 pizzas
  - much faster that 1 delivery person can deliver all 20 pizzas
  - each of the 10 delivery people delivers their pizzas sequentially

Program Structure

- All the preceding mechanisms are needed to write useful programs
- However, programming is very much a human activity
- Humans need help with the complexity of large programs
  - no single human can completely understand a program with millions of lines of code
- Program Structure helps humans write correct code
- Programming is a human activity!!!

Unstructured Programs

```
1. a = 0
2. b = 3
3. a = a + b
4. if a < 0 goto 3
5. if b < 0 goto 7
6. b = b - a
7. a = -a
8. goto 2
```

Structured Programs

- It is **unnecessarily** hard to find out what an unstructured program is doing
- structured programs make the code more accessible to programmers, without removing whatever complexity is actually necessary

Some Mechanisms for Structuring Programs

- Make it easy to create logical abstractions
  - for example, a math library or a function for spell-checking or drawing a picture
  - Related code can be in the same source file, less-related code can be in different files
- Hide unnecessary details
  - when calling `Math.sin(x)`, you don't need to know how many internal variables the `sin` function has
  - As much as possible, names should only have local significance
- None of this is easy, all have tradeoffs

Arithmetic and Logical Operations

- Java provides the four basic arithmetic operators, plus modulo (%):

  + − * / %

- and comparison operators:

  < <= == => >

  == can be used with non-numerical values

  remember that = is used for assignments, not comparisons

- and logical operators:

  && || !

Java Example

```
if (((3 + a / 2) == 7) &&
    (b < 0)) {
  b = a;
}
```

- operators have precedence, so

  3 + a / 2 is: **3 + (a/2)**, and not (3 + a) / 2
- in this class, it is safe to over-parenthesize to be sure what the grouping is
- using the wrong grouping gives runtime errors!

Data Types

- It would not make sense to add a boolean (true or false) to an integer, nor to use an integer as the condition of an **if** statement
- Every value in Java has a data type
- Data types that we've seen so far include boolean and integer

Primitive Java Data Types

- Java has 8 primitive data types:
  - `boolean`
  - four integer data types:
    - `byte`, -128 to 127
    - `short`, -32,768 to 32,767
    - `int`, - 2,147,484,648 (-2^{31}) to 2,147,483,647
    - `long`, -2^^{63} to 2^^{63}-1
  - two floating point data types, `float` and `double`
    - floating point can represent a fractional number such as 3.14
    - in your code, almost always use `double`
  - `char`

Non-Primitive Java Data Types

- Every value in Java that is not one of the primitive data types is an `Object`
- one common type of `Object` is a `String`, used to hold a sequence of printable characters such as “Hello, world!”
- Java classes give programmers the power to create new Object types
  - we introduce classes and objects later in this semester

Computer Memory

- A computer stores values in memory
- There are many different kinds of memory in a computer, including disk storage and main memory (RAM)
- Most program values are stored in RAM
  - generally, when computer people talk of memory, we mean RAM
- Values are stored in **named** locations called **variables**
- Each variable in Java has a type

Java Variables

```
int x = 3;
boolean b = true;
x = x + 1;
b = ! b;
```

- the variable **declaration** creates the **name** in the program, and reserves space in memory at runtime
- the declaration is usually combined with variable **initialization**
  - variables should only be used after initialization
- the variable can store any one value of the given type
- at the end of the above code, x is 4 and b is false

Variables, Values, Expressions

- a variable is a named location in memory
  - examples: `int x, double y, boolean b`
- an expression is a part of a computer program that computes a value
  - `2 + 2` is an expression with value `4`
  - `x > 0` is an expression whose value (`true` or `false`) depends on the value of `x`

Types in Java

- Java expressions, values, variables each have a type
- we have seen the primitive types and `Object`
- in general, types must match:
  - only `boolean` values can be used in the condition part of an `if` statement
  - only numeric types can be used in arithmetic expressions
- however, Java offers some flexibility
  - for example, we can mix integers and doubles in the same arithmetic expression (the result is a `double`)
  - we will see more such flexibility as we learn more Java
- types can help catch errors at compile time rather than at runtime!

Summary

- Basic operations include assignments, and arithmetic, logic, and boolean computations.
- Basic operations can be combined sequentially or in parallel, executed conditionally or repeated any number of times
- Variables are names for memory locations
- Program Structure reduces the cognitive complexity of programs