Type Systems

- A type is a collection of values and operations on them
  - boolean: values = {true, false}
    operations = {AND, OR, NOT, XOR}
  - int: values = {..., -1, 0, 1,...}
    operations = {ADD, SUBTRACT, MULT, ...}

- A type system defines the relationship between a type, its values, and the possible operations on those values
- A type error occurs when an operation is attempted on a value for which it is not defined

Type Checking

- Type checking is meant to ensure that operations are not applied incorrectly to the wrong types of things
  - In general, such a system can't catch everything
  - We hope it catches enough to be useful
- Types can be checked in two basic ways:
  - Statically typed languages
    - Types of all variables/objects are fixed at compile time
  - Dynamically typed languages
    - Types of a variables/objects are not determined until run time
- The more we can statically type, the more potential there is for efficiency in executable code
  - If type errors can be caught by the compiler, then the compiled code can run without any overhead to check for type problems during execution
**Static/Dynamic Typing**
- In practice, very few languages are strictly statically typed:
  - Ada and Rust are two well-known ones
  - Similarly, few do everything dynamically
  - Scheme and Ruby are two that do this
  - Many languages combine the two
    - Java checks many things at compile time, but not all (the actual operation of cast expressions, for example, isn’t checked until the code actually executes)
    - This is generally true of languages with down-casting, and some argue against adding it to languages at all for this reason

**Strength of a Type System**
- While the basic static/dynamic categories are easy to define, how “strong” a system is not entirely precise
  - Stronger languages detect more type errors
  - Weaker languages detect less of them
  - Languages exist on a spectrum, and the strength of a language is often better assessed in comparison to another one
  - For example, Java is more strongly typed than C: there are type errors that Java compilers and the JVM can detect that are not detected in C contexts (especially concerning use of pointers, which we will discuss later)

**Safety of a Type System**
- The strongest type of language is one that is entirely type-safe: no type errors are possible in executed code without obvious failure (either to compile or run) and nothing meaningless can be executed
  - Very few real-world languages are entirely type-safe (or at least can be proven with certainty to be that way)
    - In Ada, if certain restricted constructs are avoided, programs will be entirely type-safe
    - In Standard ML, similar things hold
    - For many complex languages, they are as safe as possible, but we can’t always be certain that things may not go wrong

**Type Assignment ≠ Type Checking!**
- While dynamic type assignment gives the programmer flexibility, this does not mean that a dynamically typed language is inherently more likely to have uncaught errors
  - Whether it tends to have more caught errors, due to difficulty programming with dynamic types, is a subject of many flame wars!
- The two classifications can be combined:
<table>
<thead>
<tr>
<th>Strong</th>
<th>Static</th>
<th>Dynamic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strong</td>
<td>Java, OCaml, Scala</td>
<td>Python, Ruby</td>
</tr>
<tr>
<td>Weak</td>
<td>C, C++</td>
<td>Javascript</td>
</tr>
</tbody>
</table>
Basic Types

- Most languages come with some basic types
  - Numbers of various sorts/ranges
  - Boolean values
  - Characters of text
- Even here, there are some differences, however
  - C was implemented without a specific Boolean type, and simply used an integer value (0 as false, anything else true)

```
int a = 0;
if (a = 1) {
    printf("true");
}
```

Prints “true”
In Java, e.g., will not compile

This Week

- **Topic:** Types, Chapter 7, 8.1–8.4 of text
- **Schedule changes:**
  - Next two assignments pushed back one week
  - No class on Friday, 05 April (next week): MICS conference
- **Office Hours:** Wing 210
  - Monday, 9:00 AM – 10:30 AM
  - Tuesday: 3:00 PM – 4:00 PM
  - Wednesday: 9:00 AM – 10:30 AM
  - Thursday: 2:00 PM – 3:00 PM
  - Friday: 9:00 AM – 10:30 AM