What you need to know about the “stored program” model

CMSC 122

What is the "stored program model?"

- Informally, a mathematical model offered by John von Neumann as a description of a machine capable of storing and performing algorithms–or computer programs.
- Refresh your understanding: what is a an algorithm/program?
    - A finite sequence of "instructions;"
    - Well-defined criteria for success;
    - Expressible (explainable) to the computer—meaning that the problem can be represented in some formal language.
    - Solution must be obtainable in an "acceptable" amount of time and space.
- Computer programs are machine-executable interpretations of algorithms.

How do we express programs and related elements?

- Some language that must be translatable into a language that the “machine” understands.
- But, what does “work” or “the solution” look like?
- Informally, the “program” transforms the “data” into something that looks like a solution.
- But where does this transformation take place? Where does “data” live?

Data & Program share "memory"

Data and Program reside in memory

This sounds like a simple statement, but it changed everything . . . .

- For one thing, this meant that data and program were composed of the same elements, i.e., bits.
- So we needed a way to tell these apart: *location*! Interpret the data starting at this location to mean this . . . , and . . . .
- Thus, the "meaning" of any element (object) in storage is its "location," i.e., its address!
- But, because programs can "write" (transform) data and programs are data, programs can write programs. (More on this later!)

How does this look in JavaScript?

Below are some typical JavaScript variable (and constant) declarations.

```
/* reserve a space for aVariableName */
var aVariableName;
/* reserve a space for aVariableName and
 * store value in that space.
 */
var aVariableName = value; // assigns value to aVariableName
/* reserve space for a constant named aConstantName
 * and set it to a value (that remains unchangable).
 */
const aConstantName = value; // assigns value to aConstantName
```

Some fine points here . . .

The biggest conceptual problem novices have with many programming language is their use of the equals sign, =.

- Variables (and constants) name "locations," which are places in the memory where values can be stored and retrieved.
- But, the equals sign, =, commonly names a binary relation between two objects that has certain properties (name these).
- In JavaScript it is called an "assignment" and it means the result of "storing" a value into a location. In "pseudo-code," I write

  $$\texttt{aVariableName} \leftarrow \texttt{value}$$

  to mean

  ```
  aVariableName = value;
  ```

Pseudo-Code

- I will often use "pseudo-code" to describe an algorithm, but I will then follow-up with the JAVASCRIPT equivalent.
- This is especially helpful when discussing/demonstrating location operations, such as assignment.

Create an in-class example of storing values obtained from a user, performing some transformations, storing those results, and displaying this in a Browser window.