<MeritBadge>
    <Programming />
</MeritBadge>

Presentation by Zach Colbert
Adapted from Materials by Robert Baker

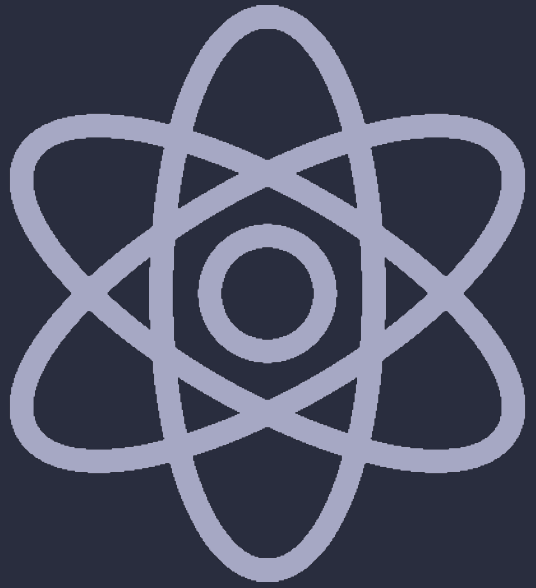
About Me

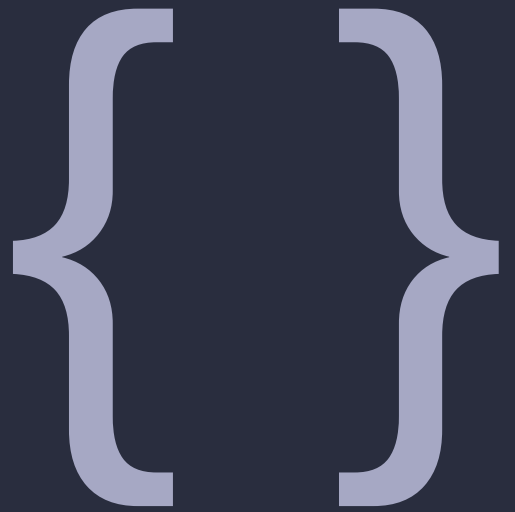

BE PREPARED
B S A


ORDER OF THE ARROW
BOY SCOUTS
OF AMERICA


ASSISTANT SCOUTMASTER

About Me





Requirements

What is Programming?

- Writing instructions for a machine.
- Computers do very simple math – we build on that to do complicated tasks.

What is Programming?

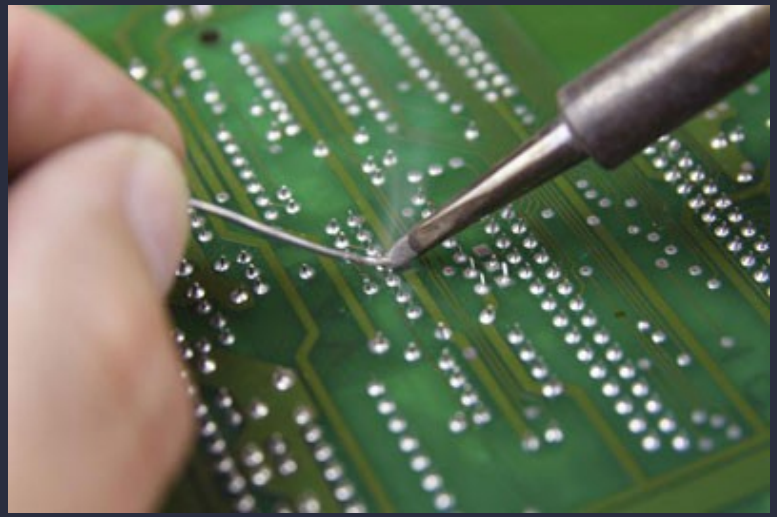
- Not just for software developers anymore.
- Programmed devices are everywhere –
  someone has to make them work!
- Different industries use different techniques, languages, systems…

- This merit badge just scrapes the surface.

{ Programming Safety }

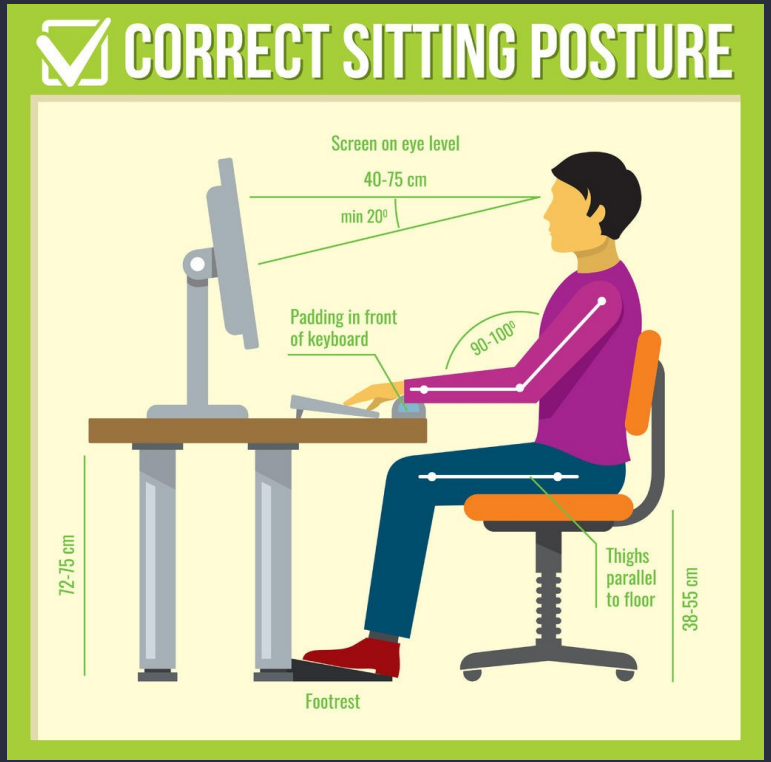
Equipment Safety

- Electric Discharges
  - Keep liquids away!
  - Power cords in good condition.
  - Unplug from power before removing components.
- Cuts and Scrapes
  - Watch out for sharp metal parts, solder joints, cardboard boxes...



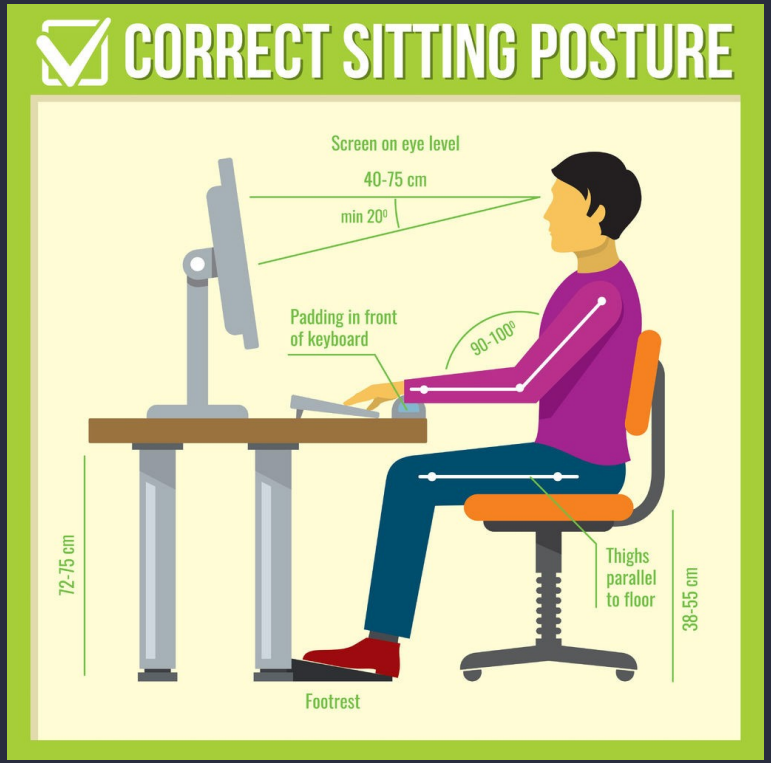
Repetitive Stress Injury

- Symptoms
  - Pain, swelling, tingling, numbness in the hands or wrists.
- Tips
  - Watch your posture.
    - Feet flat on the floor.
    - Back and wrists straight.
    - Eyes level.
  - Is your chair comfy? Can you adjust it?



Repetitive Stress Injury

- Tips
  - Stand up and stretch frequently!
- Resource: Cleveland Clinic



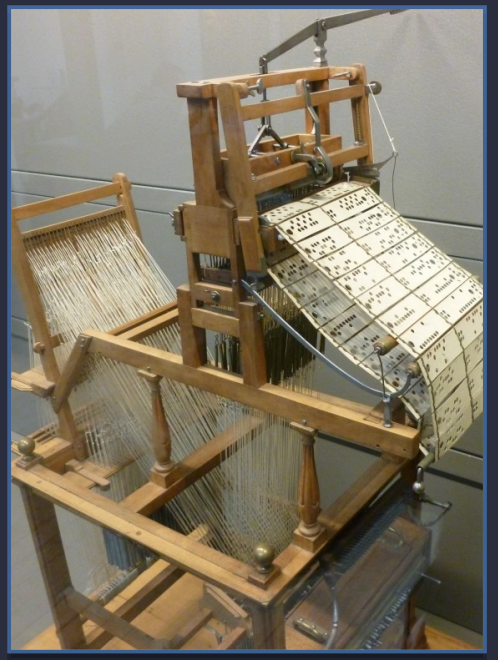
Eye Strain

- Symptoms
  - Itchy, dry eyes
  - Blurry vision
  - Headache
  - Sensitivity to light
- Tips
  - Limit screen time.
  - Take regular breaks with the 20-20-20 rule.
  - Adjust screen settings for brightness, text size, etc.
- Resource: Mayo Clinic

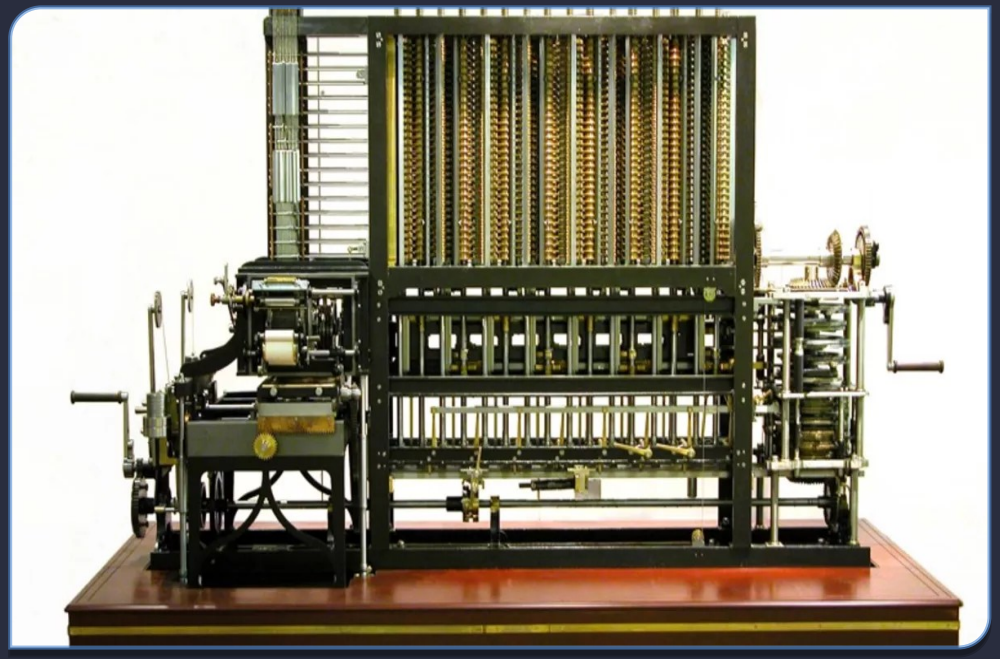
{ History of Computers }

Before Computers

- 1800s: Programmed Machines in Factories
- 1804: Joseph Jacquard
  - Mechanical loom.
  - Hole-punched cards.
  - Wove patterns into fabric.
- Left: The Jacquard Loom
- Right: A portrait of Jacquard woven in silk. The Jacquard Loom required 24,000 punched cards to weave the portrait in 1839.



Before Computers

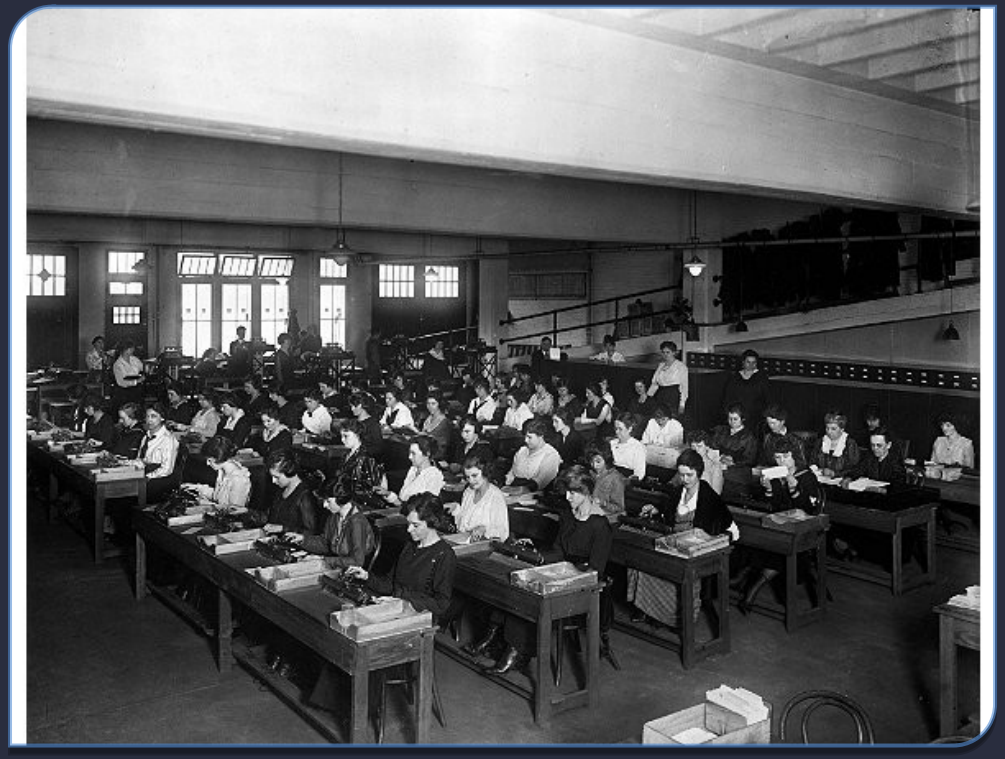


- Difference Engines
  - 1819, English mathematician Charles Babbage.
  - Mechanical calculators for tabulating polynomial functions.
  - Later designs handled numbers up to 31 digits!
  - First Babbage engine constructed in 1991.
    - 8 feet high, 5 tons.
    - Video

Before Computers

- Tabulating Machine
  - 1880s, American inventor Herman Hollerith.
  - Punched cards – but for data!
  - Electrical machines for counting.
  - Used in the 1890 US Census.
  - Continued to be used by businesses through the 20th century.

The First Computers



- “Computer” used to be a job description, not a machine.
  - Particularly important during the World Wars for maps, navigation, etc.
  - Primarily done by women.
  - Large teams of people worked together to perform and check calculations by hand.
  - “Kilogirl” – 1000 hours of computing labor.

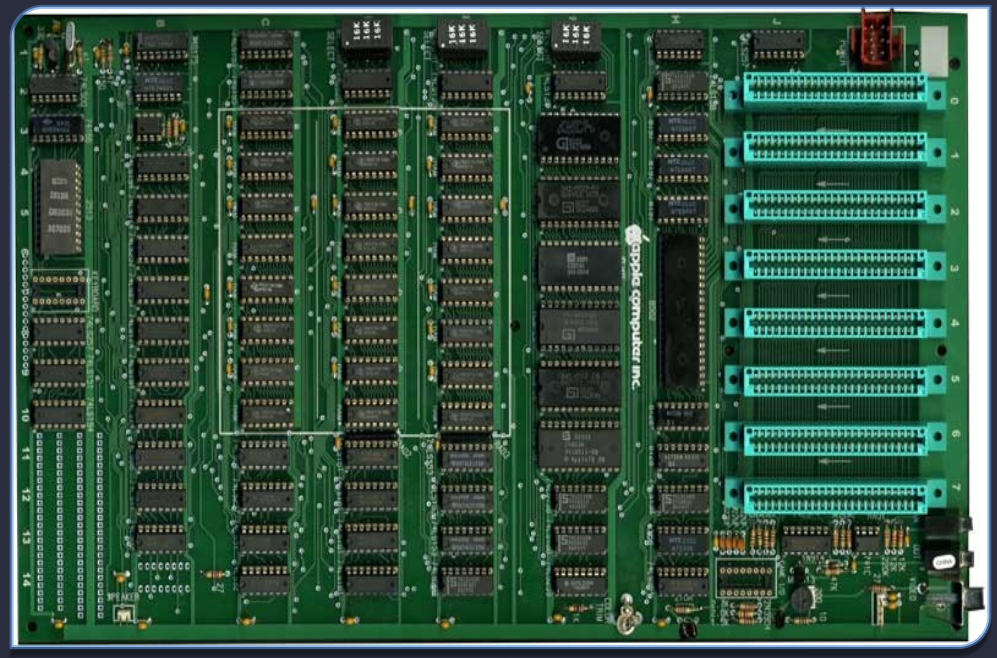
Early Computers

- 1946: ENIAC
  - “Electronic Numerical Integrator and Computer”
  - First general-purpose electronic computer.
  - Built with vacuum tubes and mechanical switches.
  - Base 10, not binary.
  - Developed by the US Army for nuclear research, artillery firing tables.

Early Computers

- 1951: UNIVAC
  - “Universal Automatic Computer”
  - First general-purpose electronic computer for business.
  - Became popular after correctly predicting outcome of the 1952 US Presidential election.

Pre-Modern Computers

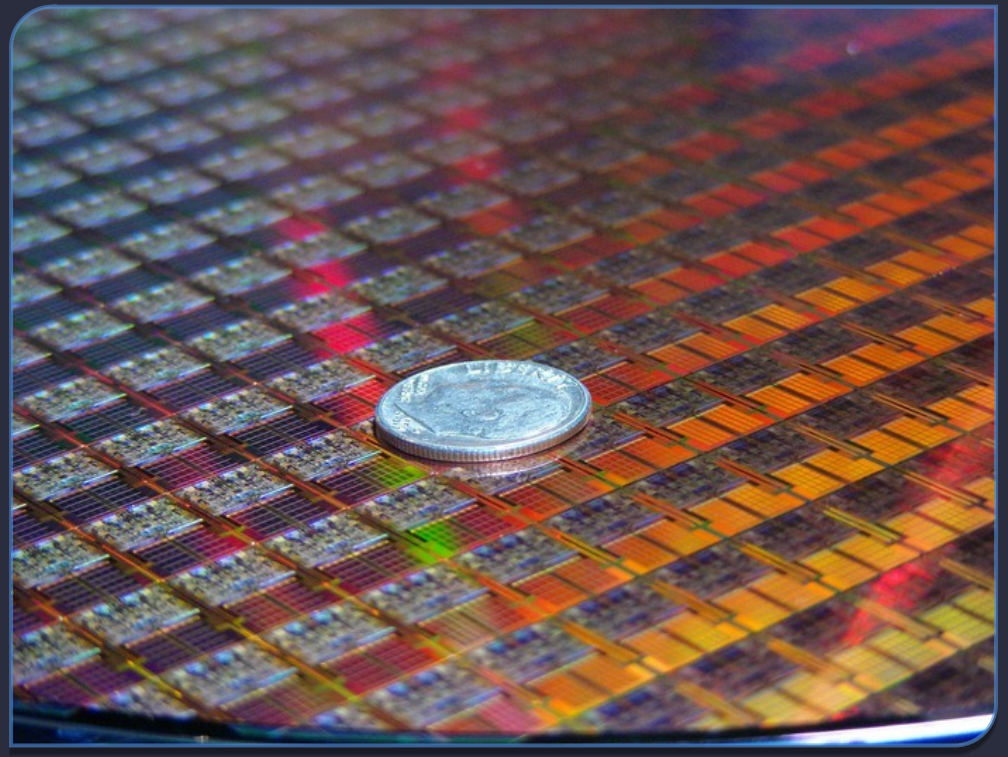


- Integrated Circuits (ICs)
  - Vacuum tubes and mechanical switches were large, bulky.
  - ICs made computers much smaller.
  - Smaller parts led to more powerful computers.

- Left: Motherboard for an Apple II home computer, 1977. All of the black chips make up the central processing unit (CPU). Each is about one inch wide.

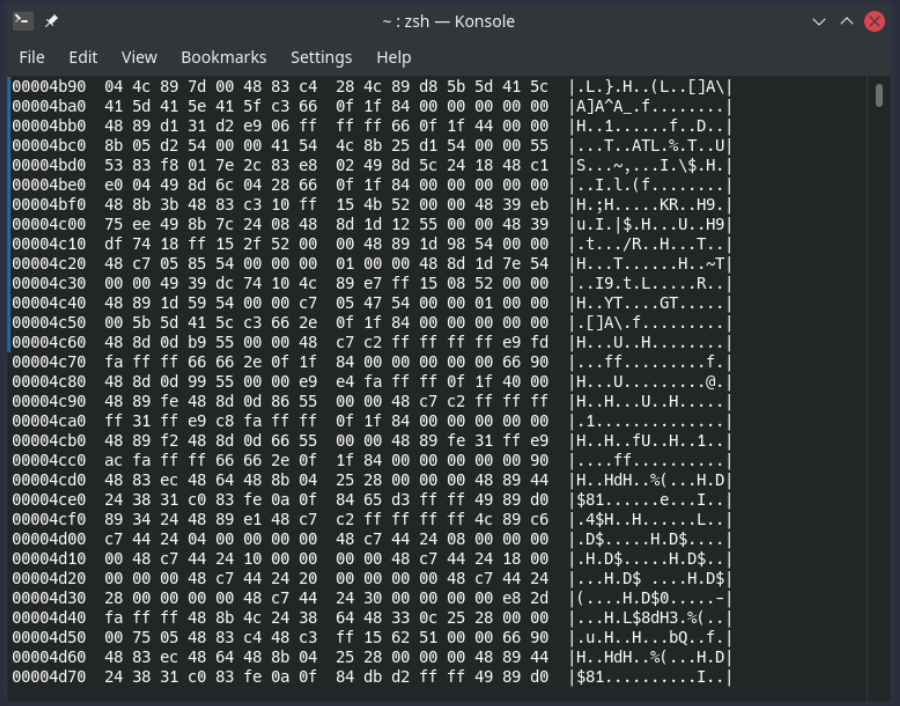
Modern Computers

- Microprocessors
  - You thought ICs were small?
  - Each small square in the picture is an entire CPU.
  - Modern computers are many times smaller, many times more powerful than any in history.
  - ~ 10 nanometer transistors.



[Evolution of Programming Languages]

Machine Language (ML)



- Binary Numbers → Instructions for the CPU
- Hard for humans to read.
  - Reverse engineers are smart!
- Compare to switches.
- As fast as your hardware.
- All languages end up translated into ML at some point.

Assembly Language (ASM)

- Slightly easier for humans to read and write.
- Still very fast.
  - Instructions map directly to ML.
- Not portable.
  - Each CPU design has its own ASM language.
  - Why is portability good?
- Few people today need to write ASM.

```
128  ; getInput: Prompts user for input, validates against min/max values, and stores
129  ;           the value in a global variable.
130  ;   Receives: Memory address at which to store value, passed by stack.
131  ;             Offset of string prompt message, passed by stack.
132  ;             Min/max values defined as global constants OP_MIN and OP_MAX.
133  ;   Returns: Unsigned integer stored in memory.
134  ;   Preconditions: DWORD memory space allocated and writeable. OP_MIN and OP_MAX
135  ;                  initialized to unsigned integer values.
136  ;   Registers changed: None
137  getInput PROC
138      push EBP
139      mov EBP, ESP
140      pushad
141
142      validateLoop:
143          mov EDX, [EBP + 8]      ; Move address of prompt string from stack
144          call WriteString        ; Print user prompt
145          call ReadDec            ; Get user input
146          cmp EAX, OP_MIN         ; Compare to minimum value
147          jl  lowBound            ; Try again if less than minimum
148          cmp EAX, OP_MAX         ; Compare to max value
149          jg  highBound           ; Try again if greater than maximum
150          jmp inBound             ; Value is within bounds!
151
152      lowBound:
153          call WriteDec
154          mov EDX, offset valTooLow
155          call WriteString        ; Print a warning message
```

High-Level Languages

```c
int _set_redirects(char* cmd) {
    int i, inf, outf, status = 0;
    char* buffer, *token;

    // Avoid strtok destruction of command line by making a copy
    buffer = calloc(CMD_MAX + 1, sizeof(char));
    memset(buffer, '\0', CMD_MAX + 1);
    strcpy(buffer, cmd);

    // Tokenize word following input redirect character
    token = strtok(buffer, "<");
    token = strtok(NULL, "<");

    if (token != NULL) {
      // Trim leading space
      for (i = 1; i < strlen(token); i++) {
        token[i-1] = token[i];
      }
      token[strlen(token) - 1] = '\0';

      // Isolate the word following the redirect char
      if (strstr(token, " ") != NULL) { token = strtok(token, " "); }

      // Trim trailing newline if one exists
      i = strlen(token) - 1;
      if (token[i] == '\n') { token[i] = '\0'; }
```

- Easy for humans to:
  - Read
  - Write
  - Debug
- Advanced Features
  - Abstract away the details, do the tedious stuff for me!
- Portable between systems.

Syntax

```cpp
/** Zach's Super Cool Example Program **/
#include <iostream>

int main() {
    char* hello = "Hello world!";

    std::cout << hello << std::endl;

    return 0;
}
```

```java
// Zach's Super Cool Example Program

class HelloWorld {
    public static void main(String[] args) {

        String hello = "Hello world!";

        System.out.println(hello);
    }
}
```

```javascript
// Zach's Super Cool Example Program

function main() {
    var hello = "Hello world!";

    console.log(hello);

    return 0;
}

main()
```

```python
# Zach's Super Cool Example Program

def main():
    hello = "Hello world!"

    print(hello)

main()
```

Modern Languages

C
C++
Java
JavaScript
HTML
CSS
Python
Ruby
PHP
OpenCL

SQL
MATLAB
Erlang
Ada
Objective-C
Swift
Mathematica
C#
Visual Basic
Rust

F#
R
Go
PowerShell
Bash
TypeScript
PostScript
CoffeeScript
Perl
x86 MASM

RegEx
PL/SQL
MIPS
ColdFusion
LaTeX
XML
JSON
Ladder Logic
YAML
Batch

- This is just a short list!

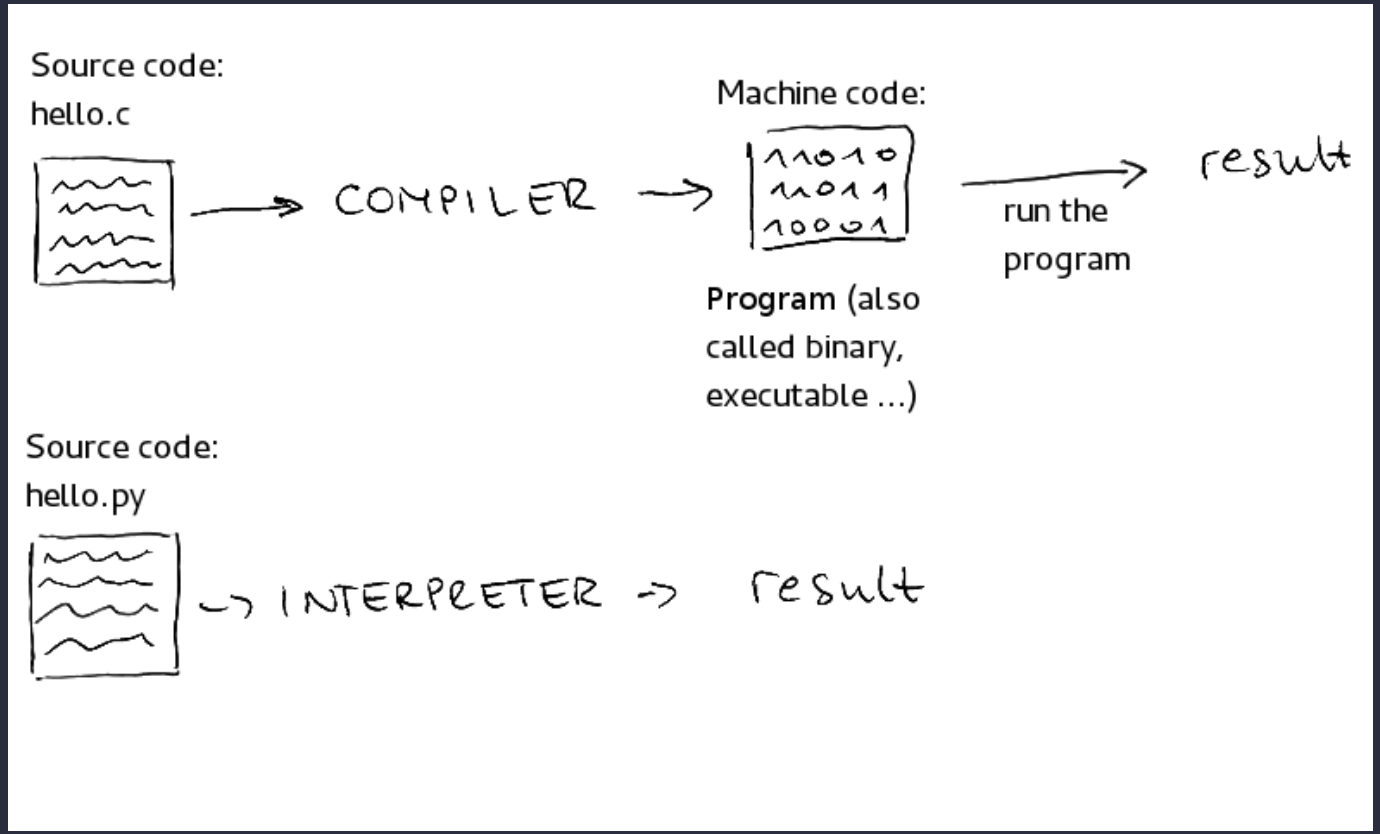
Language Types

- Many ways to classify languages.
- One language may fit more than one category.
- Examples:
  - General-Purpose Languages: C, C++, Python, Golang, Javascript
  - Scripting Languages: PowerShell, Bash, Python, Javascript
  - Markup Languages: HTML, JSON, LaTeX
  - Declarative Languages: CSS, SQL, RegEx

Use Cases

- Languages have features that make them better for some uses than for others.
- Examples:
  - C++ : General purpose, high performance. Desktop applications.
  - C : General purpose, high performance. Operating systems and drivers.
  - Java : General purpose, cross-platform. Desktop and mobile apps.
  - SQL : Database communication.
  - HTML : Markup langage for webpage design.
  - Javascript : Scripting for interactive webpages.

Compiled vs. Interpreted



Categorizing Languages

- Levels
  - Machine Level (binary)
  - Low Level (ASM)
  - High Level (C++, Python, …)
- Compiled vs. Interpreted

- Use Cases
  - General Purpose (C++, Python)
  - Scripting (PowerShell, Bash)
  - Markup (HTML, JSON)
  - many other categories exist...

{ Programmed Devices }

{ Intellectual Property }

Types of IP

- Copyright
- Patent
- Trademark
- Trade Secret

Types of IP

- Copyright
  - Protects expression of ideas, authorship (books, movies, art).
- Patent
  - Protects functional things (machines, items, processes).
- Trademark
  - Protects brands (words and phrases, symbols, logos).
- Trade Secret
  - Protects secret information by never disclosing it.

Owning vs. Licensing Software

- Owning
  - You have the software.
  - You probably have the code.
  - You do what you want.
- Licensing
  - You have the software.
  - You have permission to use the software, with some conditions.

- Do I own a copy of Google Chrome?
- Do I own a copy of Windows?
- Do I own a copy of an app I built?

Licensing Software

- Freeware
- Shareware
- Demo
- Public Domain

Licensing Software

- Freeware
  - 100% free to use.
  - Not necessarily free to modify or distribute.
- Shareware
  - Free to use.
  - You may have to pay later, or pay for extra features.
  - Not necessarily free to modify or distribute.
- Demo
  - “Free Trial.”
  - May have restrictions like missing features, limited time.
  - Not necessarily free to modify or distribute.
- Public Domain
  - Absolutely no ownership.
  - Free to use, modify, distribute, or even sell.

Licensing Software

- Licensed (Commercial)
- Subscription-based

Licensing Software

- Licensed (Commercial)
  - You pay once.
  - You have permission to use the software.
  - Not free to modify or distribute.
- Subscription-based
  - You pay periodically.
  - You have permission to use the software.
  - Not free to modify or distribute.

Licensing Software

- Open-Source
  - You have access to the code.
  - You can modify and distribute the code – with some restrictions.
  - Most open source software will allow you to contribute your modifications.
- Closed-Source
  - You don’t have access to the code.

{ Careers in Programming }

https://www.onetonline.org/find/descriptor/result/2.B.3.e?a=1

{ Next Steps }