FCPA 2022  
  
How to Learn C

Student Workbook 01

January 10, 2022

Author

Paul Kimball  
Interface Associates

# Welcome to C Programming

## C has been around for almost 50 years!

### In dog years = 300, maybe?

### In computer years = 20+ generations of Moore's Law\*

## How has it survived so long?

### Established

### Small and Fast

### Relevant - Respects details; very useful for writing system software

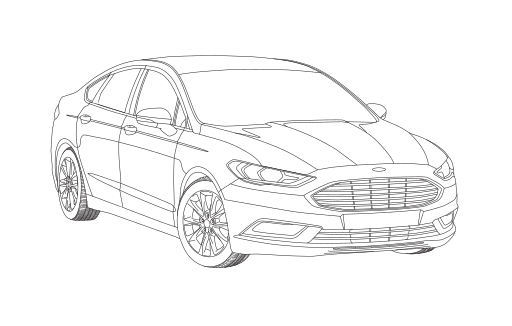
### Maintained - Specification is updated every few years

### Stable

\*Refers to an observation by Gordon Moore in 1965 that the number of transistors in a dense integrated circuit (IC) doubles about every two years.

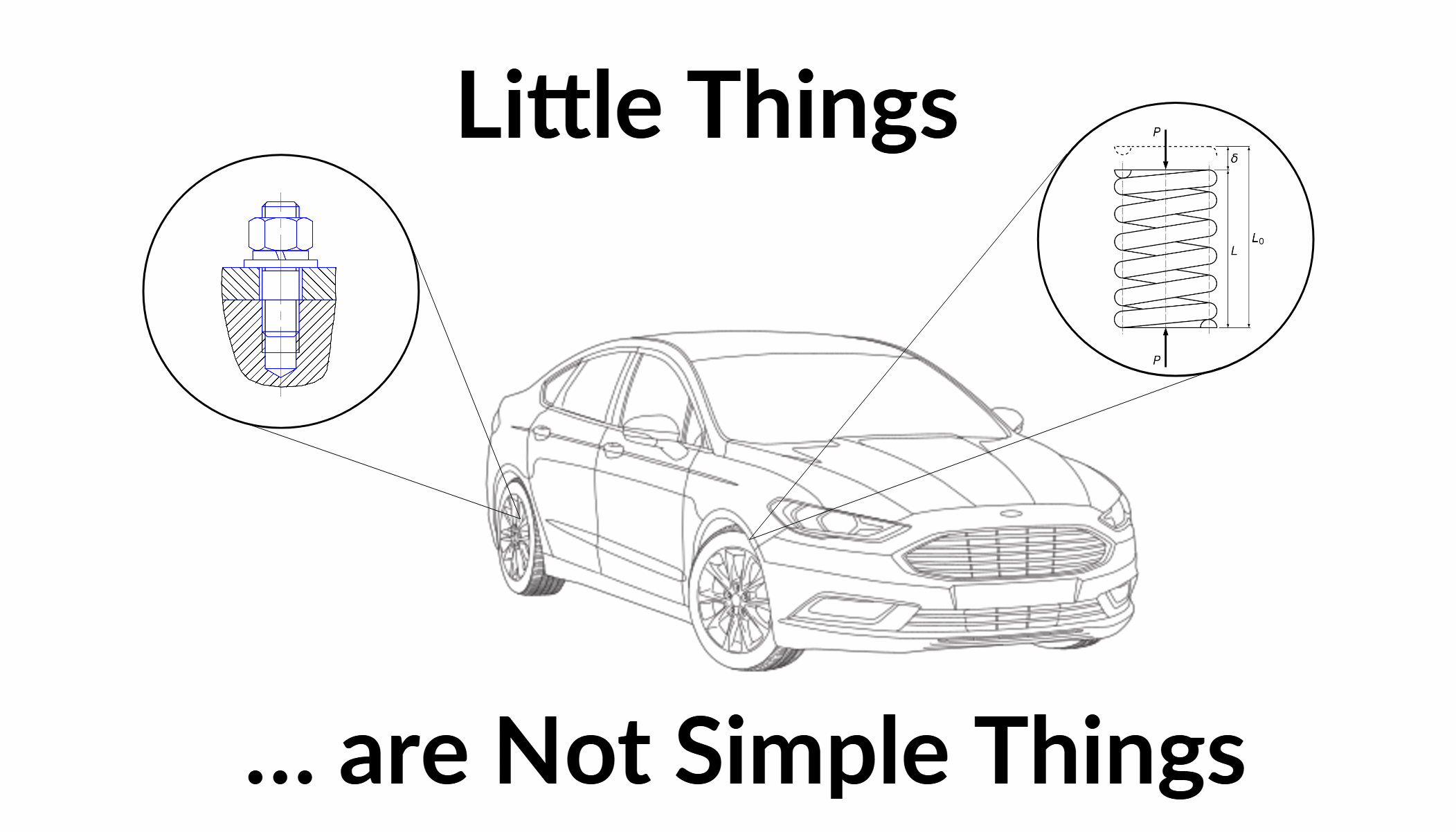
"Big" Things are Simple

* Fun, Functionality, Freedom, and Safety too!



* ANYONE can drive a car!

"Little" Things are Complicated



* Very few people know how to design or build a wheel stud
  + Or a spring...
  + Or...
* All the little things have to fit together

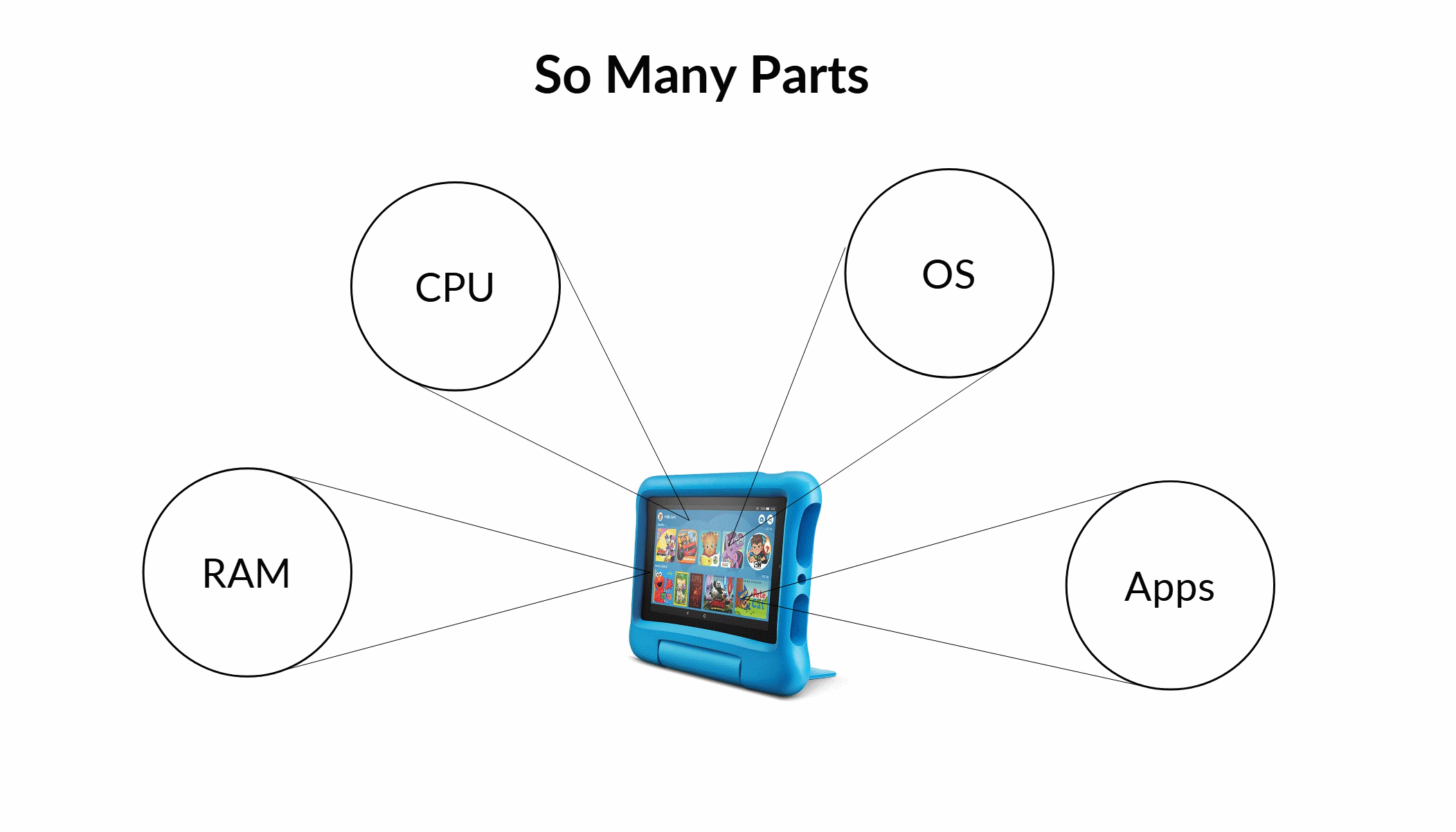
"Big" Software Is Easy



* Fun, connected, easy to use, safe and secure!
* ANYONE can use a computer

"Little" Software is Complicated

* Very few people know how to design or write a program



* All the little programs have to work together

# You Won't Just Learn C

* You'll also learn about these "little" things:
* Visual Studio and IDEs
  + Make 'em your own with fun extensions and customizations
* The C Toolchain (preprocessor, compiler, linker)
  + A million options for a million hardware needs!
* The debugger
  + See for yourself what really happens
* Command shell(s)
  + A chance to meet new commands and options
* The file system and I/O streams
  + Where's my stuff?
* Git and GitHub source-code management
  + Killer or Quirky? You decide! It's here to stay.
* Testing tools and MISRA
  + Whose rules really rule?
* MCUs and Memory
  + Your future is embedded
* Plus, the complicated "littlest" things:
  + Math and numbers
  + Characters and Strings
  + Addresses and memory
  + Instructions

# Programming Languages

* Are human languages
  + Have roots in older languages
  + Change over time
  + Adopt useful features from competing languages
  + Have recognizable periods of use

|  |  |  |
| --- | --- | --- |
| **Era** | **English** | **C** |
| Classical | Middle English | K&R C |
| Early Modern | Elizabethan | ANSI C89 |
| Modern | Victorian | ISO C99 |
| Modern | American | ISO C11 |
| Modern | Hipster | ISO C17 |

* Many local dialects and conventions
  + Usage is constrainted by context (e.g. MISRA rules, "Polite" english)
  + We love to bicker about details
* Some ancient forms survive
  + Verily, Thou knowest this be Naught but Truth
  + x = \*(c\_ptr++);
* Humans are EXTREMELY GOOD at learning languages
  + You are hard-wired for this!
  + Also, your ancestors ate mammoths, so, you're ready.

# How We're Going to Do This

## Over 70% Coding

## Before the class

* + Pre-work that makes you think about Embedded C

#### Setting up your workstation and software

#### Assigned readings

#### On-line video content

* + C Skills IQ evaluation
  + Survey

## Every day:

* + Standing meeting; be prepared to talk about:

#### An accomplishment from the day before

#### Something that is still challenging you

#### Some topic of life out on the C shore (I'll pick one)

* + Theme of the day - where are we going
  + Minimalist lecture with examples
  + Instructor-led Programming (with Popsicle Sticks), leading to

#### Individual programming

#### Team programming

* + Book of the day
  + Recap

## Every Week

* + A lot of programming projects
  + A little quiz (probably Friday morning)

## After the class

* + C Skills IQ re-evaluation
  + Much parties
  + Keep in touch!

How We'll Code

* I'll tell you about it
* I'll show you how to do it
* We'll do it together
* You'll do it alone
* You'll show me

# Learning C Syntax

* There's muscle memory involved
  + Learned by repetition, practice
* Slow down
  + Read/write word by word
* Use it, don't avoid it
  + Write weird things over and over until they feel right
  + Spell out words completely many times before relying on intellisense
  + Read the references even when you don't completely understand them
* Consolidate your Memories
  + Read and repeat concepts out loud
  + Say it in other words
  + Write it down
  + Tell someone else about it
  + Take a break
  + Sleep on it
* Consult expert viewpoints to try to understand WHY
  + It'll make you laugh, after it makes you cry

# Compartmentalize

* You will switch contexts a lot
* Cheat sheets and checklists are essential
  + I'll give you some; expand them and make your own
* Build your environment to help you maintain your context
  + Reserve a place to code and study
  + Include visual and sensory cues that mark your "study" space
  + Use visual cues (fonts, colors) to distinguish your "coding" apps
* Organize your virtual desktop
  + Put course materials in a folder that is easy to get to
  + Organize labs and handouts by day, subject, workbook
  + Name folders and files to help your future self find them
  + Tidy up your screen backgrounds and toolbars
* Organize your physical space
  + Room for books, manuals, notebooks
  + Room to stretch
  + Room to write

# Keep Me Honest

* Verify what we teach by trying it!
  + Write it, run it, test it, try to break it
* You'll be Googling a LOT!
  + And I will point you to lots of resources
* Check more than one source
  + There's usually several accepted ways to accomplish the same thing

Classic C way

Modern more structured way

Embedded systems way

* + Plus, several idiosyncratic ways that are one person's dream, another's nightmare
* Ask for others' opinions; don't be surprised by strong feelings
  + Better have a sense of humor
  + Be adaptable
  + Be resilient

## Don't rely on "Tips and Tricks"; internalize the language and tools

# Work With a Net

* Lean on your tools - find your favorites!

### Code editor will help you with syntax, spelling

### Git repositories save your code so you can always try again

### Compiler/linker messages are trying to help you! Read them.

### Debugger can tell you what really happened. Use it frequently while learning.

* Lean on your references

### Specifications

### Datasheets

### Books

### Example code

### Google

* Lean on your friends

### Fellow programmers in your breakout room or IRL

## Lean on me!

### I'm always in the (virtual) lobby to help you and answer questions!

# Stay Comfortable but Alert

* We'll be doing a LOT of typing
  + But you'll be doing 10x more sitting, reading, and talking
* Take care of yourself

### Eat, hydrate, take bathroom breaks

### Stand up and stretch!

### Have snacks and bevs handy

### Get enough sleep

## If you feel like you're fading, do something about it!

### Ask a question

### Talk to someone

### Take a break, but come back!!

# Ergonomics

* Indirect lighting



* A comfortable, adjustable chair
* Keyboard and monitor at the correct heights
* A good keyboard and mouse, easy to hold, tactile feedback
* Large screen(s) and big, clear fonts
* A pleasant, context-establishing windows theme
* Physical aids as appropriate
  + Reading glasses
  + Footrest, armrest
  + Headset, speakerphone

# Don't Worry

* There's plenty for everyone
* You won't break the machines
  + You can't break my backup repository
* You won't fail

### I already made all the mistakes I could think of

* Ask for help; you'll get it
* Discuss your problems
* Savor your successes