01 Mo Lab Notes 2023

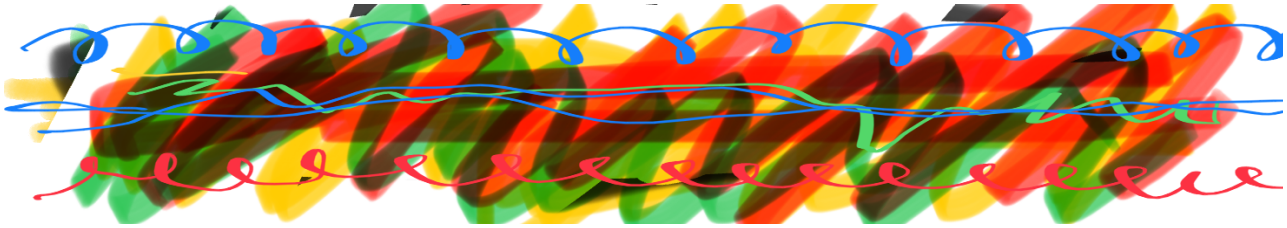
Mobile App Development Lab 2023
ITPG-GT 2372
1/24/2023 - 5/2/2023
3:20 PM - 5:50 PM Tu
370 Jay St, Room 409 Loc: Brooklyn Campus

Full-time access to an iOS device and a Mac laptop computer running the latest operating system and development tools are required.

Prerequisite: Some programming experience (such as ICM) and willingness to learn Apple's Swift programming language.

One of the most transformative consumer products in history, the iPhone remains the standard bearer for great design and user experience. With the latest versions of iOS and iPhone, Apple puts depth sensing and augmented reality in our pockets. How do we take advantage of this incredible platform to produce our own compelling experiences?

This course will be a hands-on workshop where we explore the world beyond generic apps and push the boundaries of what's possible on iOS hardware. Each week, you'll be asked to complete a programming exercise meant to foster your understanding of iOS application development. We'll leverage existing open source libraries to quickly build out your app with features such as real time communication and cloud storage.



We aim to create distributed instruments for computed expression.

Course overview

https://github.com/mobilelabclass-itp/content-2023

Grading:

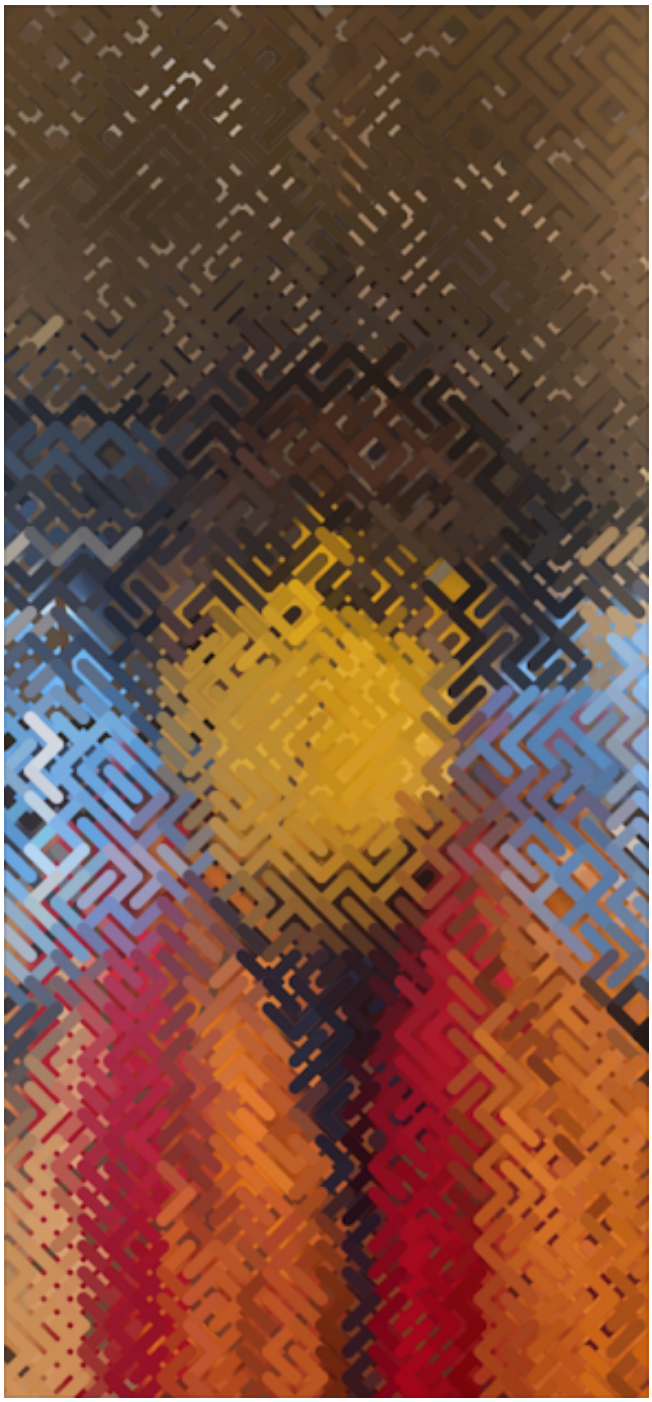
- Regular Assignments 30%
- Participation and Attendance 30%
- Final Project 40%

keep final project in mind as you learn Swift - a destination can guide your learning

Class Intros

[] create intro wiki page
[] github user name reminder

Skin Tone screen, part of Colored Portraits installation
https://jht1493.github.io/2021-NYU-ITP-Installation/



My DICE platform
http://www.johnhenrythompson.com/3-dice
2001 QuickDraw / QuickTime / C++ Carbon MacOS app
2005 Carbon / Javascript / PowerPlant
2017 Objective-C, AVFoundation
2021 NYU-ITP Installation - Javascript / p5js / ml5.js
I plan to update DICE to Swift / SwiftUI this year.

Example of evolving software ecosystem

My other iOS apps

Creative coding and creative learning

Chris Lattner - creator of the Swift language
https://nondot.org/sabre/

How we got here:

Doug Engelbart
- 1968 NLS (oN-Line System)
- mouse, video conference, multiple views of information

Alan Kay - Smalltalk

Guy L. Steele and Gerald Jay Sussman - Scheme (predecessor to JavaScript / Java)

Richard Stallman - Open Source movement - GNU/Linux

Linus Torvalds - Linux + git

https://www.youtube.com/watch?v=8QiPFmIMxFc
Bret Victor Inventing on Principle
Jan 17, 2018
33 mins
>> creators need an immediate connection to work

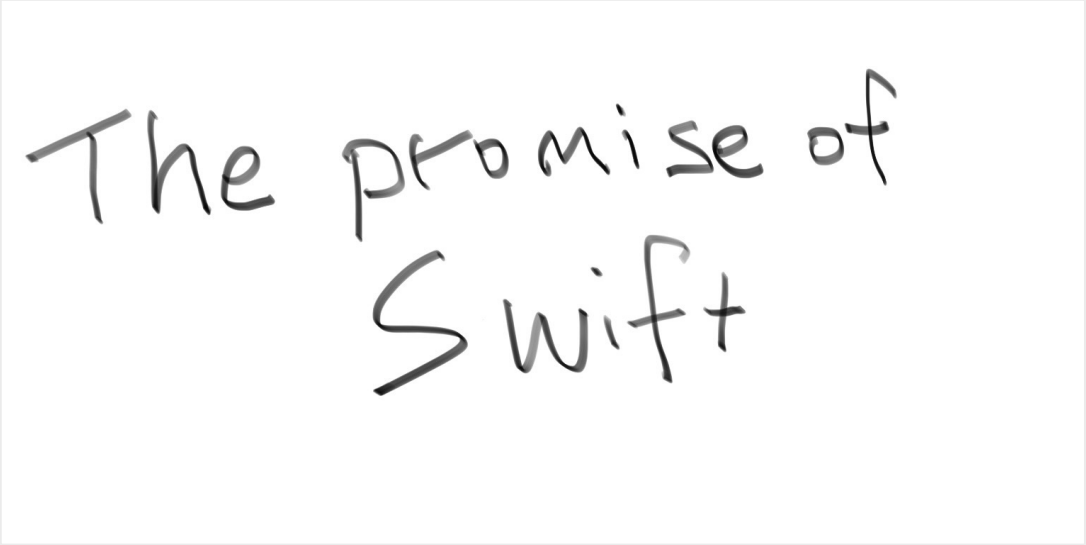
Class discussion - your experience with creative coding.

Why Apple Swift?
Apple has demonstrated history of innovation
- 1984 Macintosh
- 2001 iPod
- 2007 iPhone

Goal to make computer personal expressive instruments

App for creative expression building on the unique palette (libraries) of the platform

-Break



https://www.youtube.com/watch?v=MO7Ta0DvEWA
Apple WWDC 2014 - Swift Introduction
10 min

Separating marketing speak from reality
Why Objective-C vs. Python, rather than C / C++?
Complexity and expressiveness is the underlying issue
Appropriate level of abstraction

In rapidly evolving software ecosystem hard to finding good documentation and examples

I could not find working code for demo of swift intro

Focus on SwiftUI to simplify learning
Apple migrating away for Storyboard/UIKit

Strategy

- Explore features of interest topic in WWDC videos, building towards a final project
- Organize your notes - I use VS Code
- Apple documentation
- Apple sample code
- Check resources. Current? Works?

Different approaches to learning - multiple paths

- Reading

- Watching
- Exercises
- Experimenting

Explore resources

https://github.com/mobilelabclass-itp/content/blob/main/weeks/01_intro.md#resources---apple-platform

https://developer.apple.com/videos/all-videos
WWDC Videos - What Apple says is possible on their platform
search to find area of interest: RealityKit, AVFoundation, Filters

>> Example WWDC treasure hunt:

https://developer.apple.com/videos/topics/

https://developer.apple.com/videos/photos-camera

https://developer.apple.com/videos/play/wwdc2018/503/

Creating Photo and Video Effects Using Depth

https://developer.apple.com/documentation/avfoundation/cameras_and_media_capture/capturing_photos_with_depth

https://developer.apple.com/documentation/avfoundation/cameras_and_media_capture/streaming_depth_data_from_the_truedepth_camera

>> Big stretch
https://developer.apple.com/documentation/realitykit/swiftstrike_creating_a_game_with_realitykit
SwiftStrike: Creating a Game with RealityKit
Create a multiplayer game with ARKit, RealityKit, and Swift using the SwiftStrike app as a guide.

>> Photos app curated memory:
Philadelphia Jun 24, 2016
I don’t use social media
https://github.com/jht1493/jht-site#jht-site

How do we get there?

https://docs.swift.org/swift-book/GuidedTour/GuidedTour.html
- Apple Swift Docs - A Swift Tour

Hands on:
- Create your repo
- Use Xcode to create a playground in your repo
- Update your repo
- Add link to repo on homework page

Review homework
https://github.com/mobilelabclass-itp/content-2023/blob/main/weeks/01_intro.md#homework

Explore 01-Javascript-to-Swift and 01-Playground pages

Build a pages for experimenting with language features as you learn them
Hands on example

Recap repos
- content
- content wiki
- 01-Javascript-to-Swift
- 01-Playground

Big picture recap and context

The Evolution of Software ecosystems

>> Swift is rapidly evolving

Swift 5.5 2021-03-12 2021-09-20
Swift 5.0 2018-09-25 2019-03-25
Swift 4.0 2017-02-16 2017-09-19
Swift 3.0 2016-05-06 2016-09-13
Swift 2.2 2016-01-05 2016-03-21

>> SwiftUI is new - intro 2019

Other Software ecosystems for mobile apps
  Flutter/Dart
  React Native
  HTML5/Javascript/CSS/Electron (for the desktop)
  Android Java/Kotlin
Where do they differ?
Where do they converge?

Compiled vs. Interpreted

Complexity
  State management
  Async

History of computer language evolution
inflection points
new programming paradigms

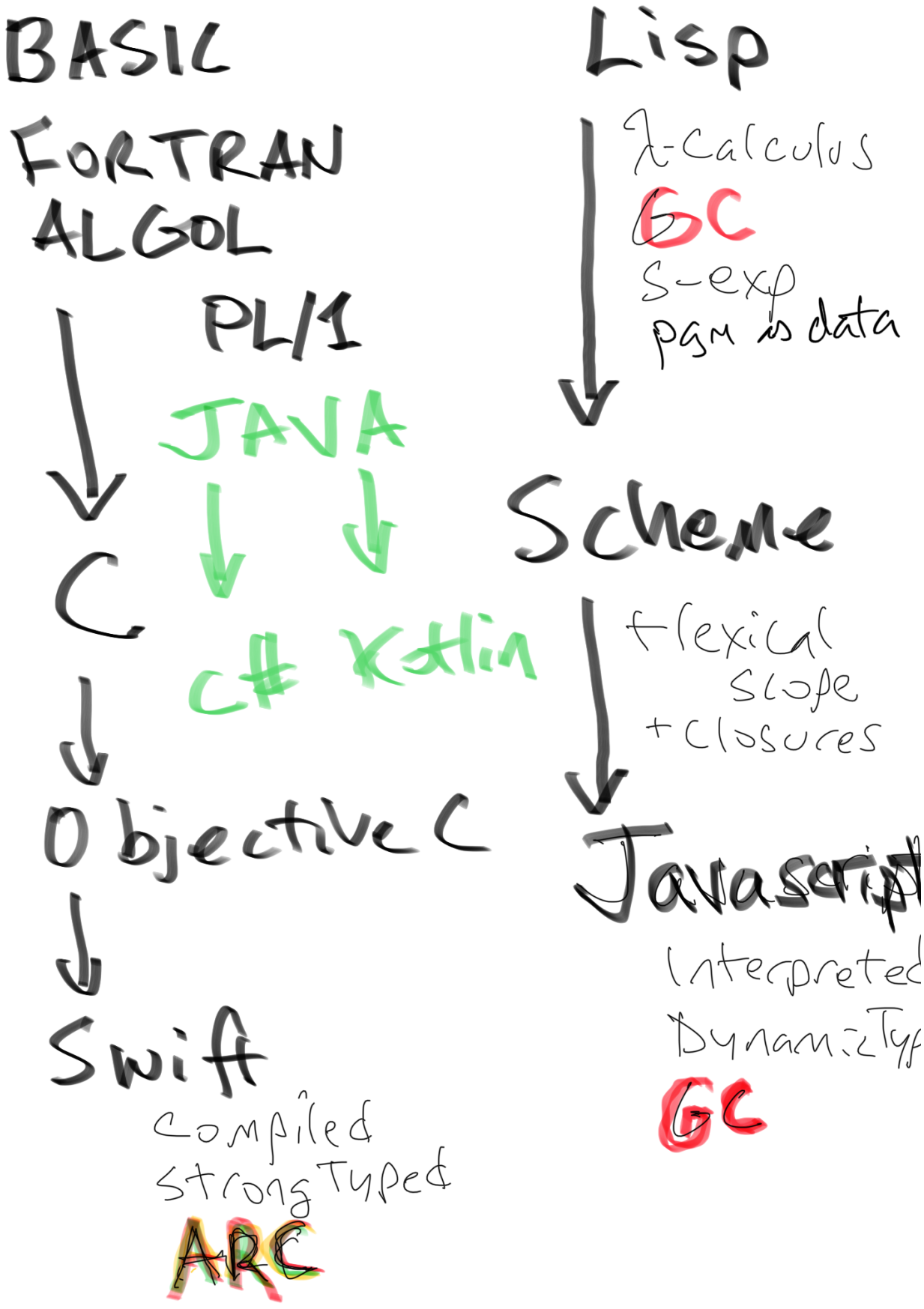
  - BASIC

  - Structured Programming
    Algol lexical scoping vs. Fortran flat
    Recursion

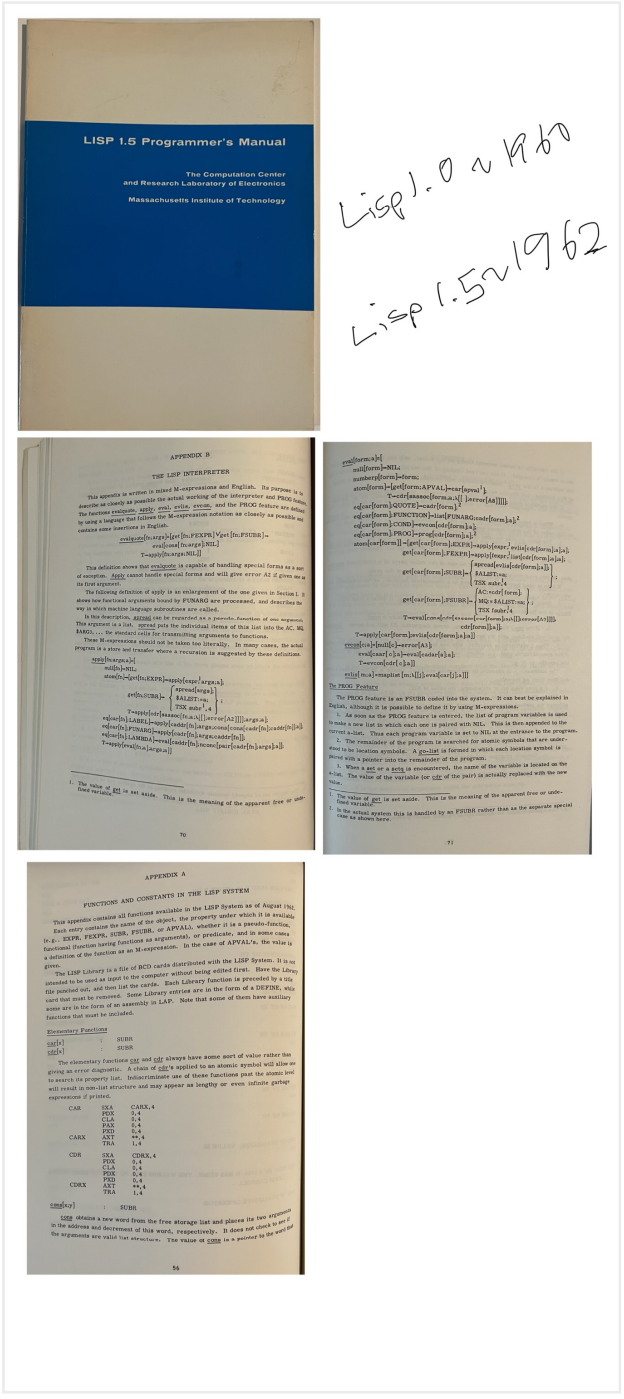
  - Data Abstraction
    --> Algol, C, Pascal
    <-- Fortran, Assembler

  - Dynamic
    Lisp -> Scheme -> JavaScript

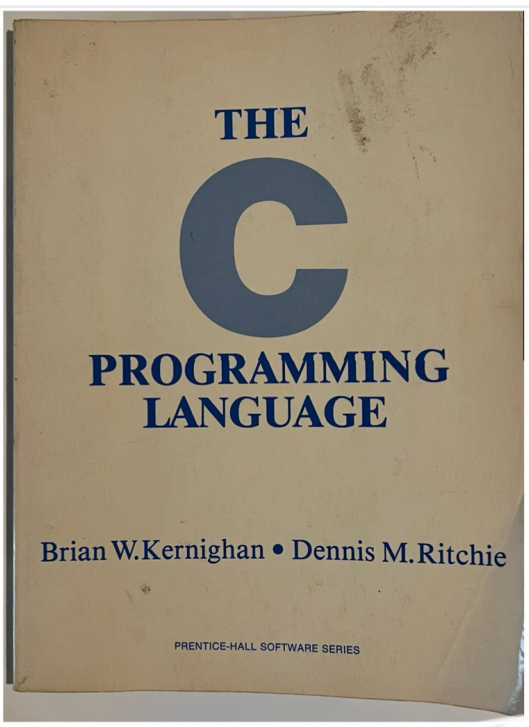
  - OOP
    Smalltalk
    Object Pascal
    Objective-C
    Java
    C# (Microsoft's response to Java)

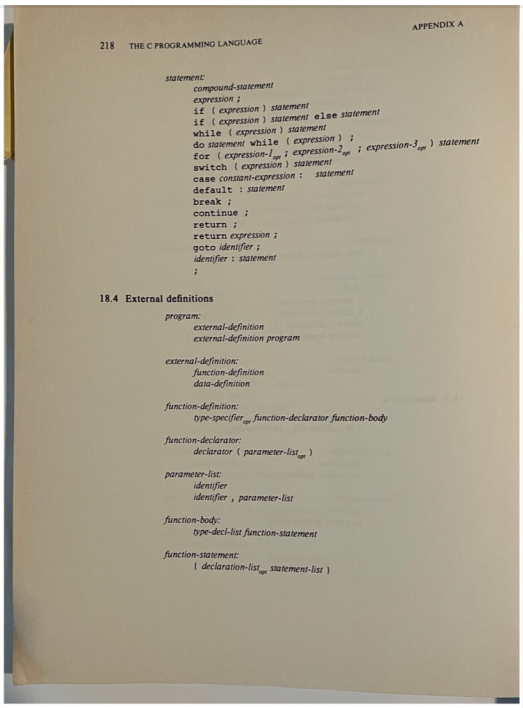

BASIC
FORTRAN
ALGOL
PL/1
C
Objective C
Swift
compiled
strong Typed
ARC
Lisp
λ-calculus
GC
S-exp
pgm is data
JAVA
C# Kotlin
Scheme
+lexical
scope
+closures
Javascript
Interpreted
Dynamic Typ
GC

Lisp 1.5 Programmer's Manual
approx. 106 pages
Beautifully concise and expressive language - the root of Scheme/Javascript, Java/C# and other recent languages
In spired me to write Lisp interpreter for PDP-8 (~1978), and Lisp compile for IBM 360 (~1983)
Foundation for my creation of the Lingo scripting language for Macromedia/Adobe Director (1990's)



The C Programming Language
approx. 227 pages
From complex to simple (Multics to Unix) and back to complex (C++)





APPENDIX A

218 THE C PROGRAMMING LANGUAGE

statement:
compound-statement
expression ;
`if` (*expression*) *statement*
`if` (*expression*) *statement* `else` *statement*
`while` (*expression*) *statement*
`do` *statement* `while` (*expression*) ;
`for` (*expression-1*$_{opt}$; *expression-2*$_{opt}$; *expression-3*$_{opt}$) *statement*
`switch` (*expression*) *statement*
`case` *constant-expression* : *statement*
`default` : *statement*
`break` ;
`continue` ;
`return` ;
`return` *expression* ;
`goto` *identifier* ;
identifier : *statement*
;

18.4 External definitions

program:
external-definition
external-definition program

external-definition:
function-definition
data-definition

function-definition:
type-specifier$_{opt}$ *function-declarator function-body*

function-declarator:
declarator (*parameter-list*$_{opt}$)

parameter-list:
identifier
identifier , *parameter-list*

function-body:
type-decl-list function-statement

function-statement:
{ *declaration-list*$_{opt}$ *statement-list* }

~1978

Multics (PL/1)

↓

Unix (C)

↓

Linux (git)

MainFrame ~ sizen =room=

Mini = fridge-

Micro = toaster=