

$\Delta I \Phi$

Lesson 03:

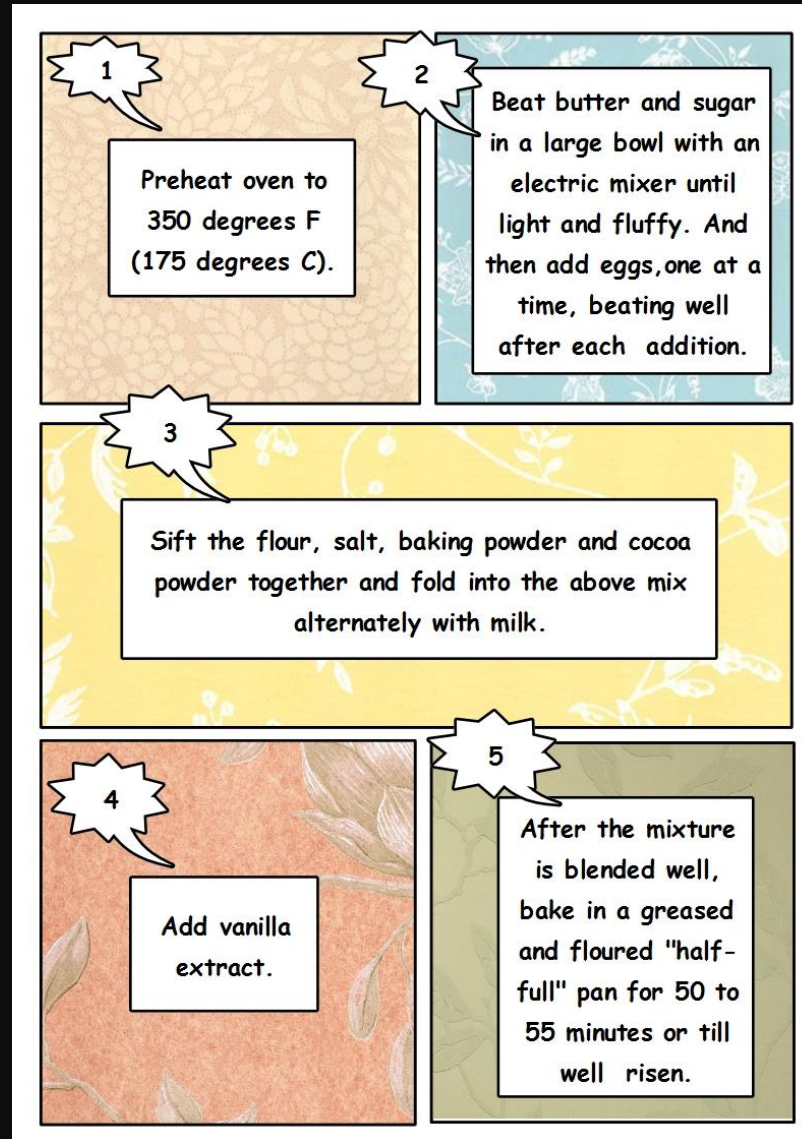
Computer

Programming

The Algorithm

A List of Well-Defined
Instructions for
Completing a Task.

Algorithm to Bake a Cake

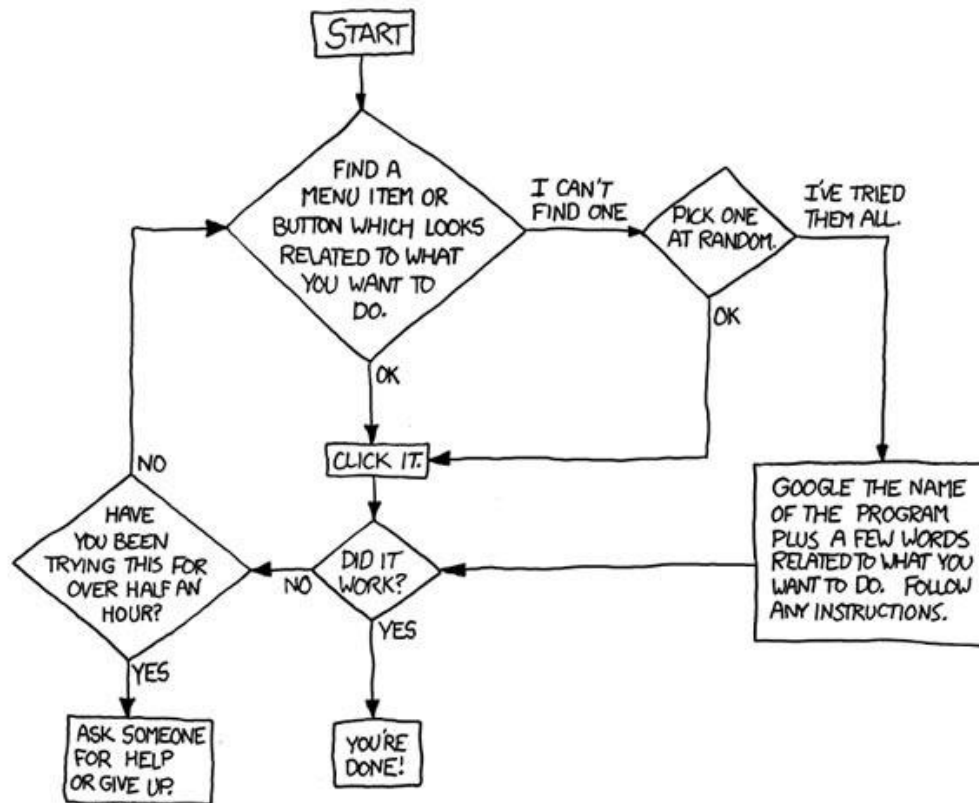


How Detailed
Does an
Algorithm
Need to Be?

Troubleshooting Algorithm

DEAR VARIOUS PARENTS, GRANDPARENTS, CO-WORKERS,
AND OTHER "NOT COMPUTER PEOPLE:"

WE DON'T MAGICALLY KNOW HOW TO DO EVERYTHING IN EVERY
PROGRAM. WHEN WE HELP YOU, WE'RE USUALLY JUST DOING THIS:



PLEASE PRINT THIS FLOWCHART OUT AND TAPE IT NEAR YOUR SCREEN.
CONGRATULATIONS; YOU'RE NOW THE LOCAL COMPUTER EXPERT!

Boolean Logic

UNIVERSE (Integers from 0 to 30)

1 7 11 13 17 19 23 29

SET A (Even numbers/Multiples of 2)

2 4 8 14 16 22 26 28

SET AB (Multiples of 6)

6 12 18 24

SET AC
(Multiples of 10)

10 20

SET ABC
(Multiples of 30)

0 30

SET B
(Multiples of 3)

5 25

SET BC
(Multiples of 15)

15

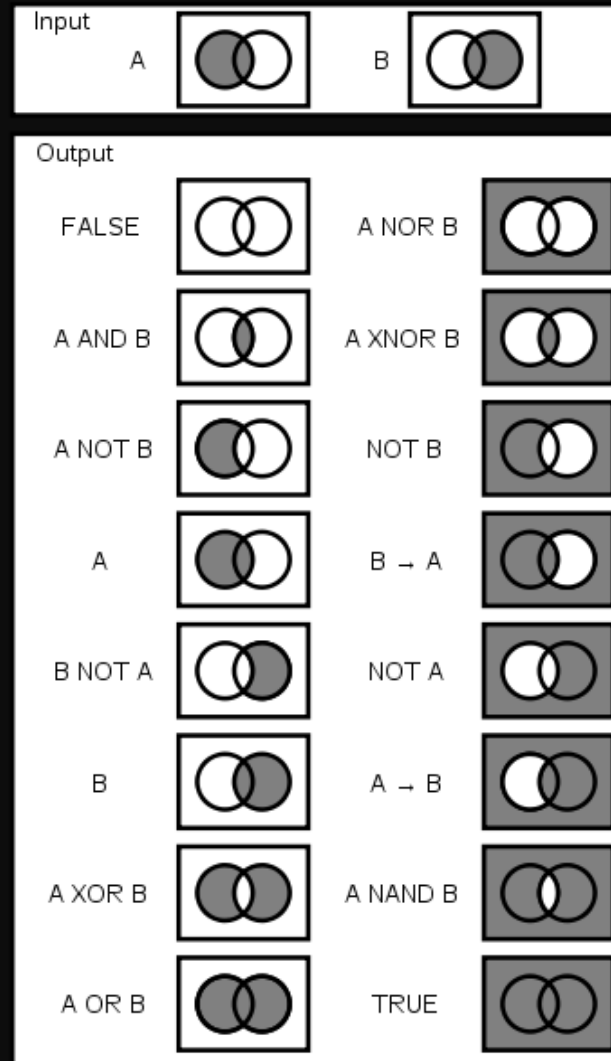
3 9 21 27

SET C (Multiples of 5)

Logic Gates

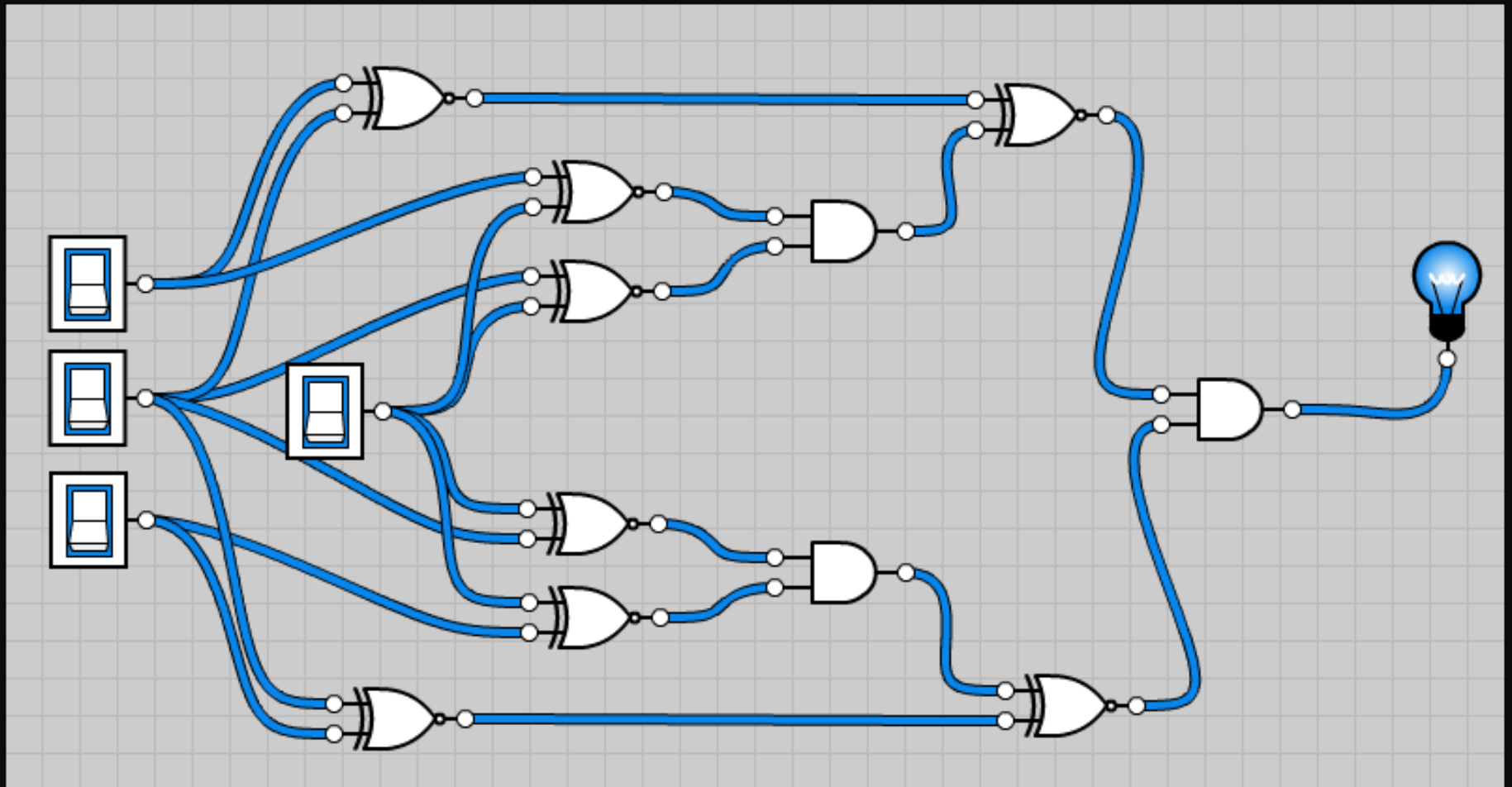
Name	Graphic Symbol	Algebraic Function	Truth Table															
AND		$F = A \cdot B$ or $F = AB$	<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>F</th> </tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </tbody> </table>	A	B	F	0	0	0	0	1	0	1	0	0	1	1	1
A	B	F																
0	0	0																
0	1	0																
1	0	0																
1	1	1																
OR		$F = A + B$	<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>F</th> </tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </tbody> </table>	A	B	F	0	0	0	0	1	1	1	0	1	1	1	1
A	B	F																
0	0	0																
0	1	1																
1	0	1																
1	1	1																
NOT		$F = \bar{A}$ or $F = A'$	<table border="1"> <thead> <tr> <th>A</th> <th>F</th> </tr> </thead> <tbody> <tr><td>0</td><td>1</td></tr> <tr><td>1</td><td>0</td></tr> </tbody> </table>	A	F	0	1	1	0									
A	F																	
0	1																	
1	0																	
NAND		$F = (\overline{AB})$	<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>F</th> </tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </tbody> </table>	A	B	F	0	0	1	0	1	1	1	0	1	1	1	0
A	B	F																
0	0	1																
0	1	1																
1	0	1																
1	1	0																
NOR		$F = \overline{(A + B)}$	<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>F</th> </tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </tbody> </table>	A	B	F	0	0	1	0	1	0	1	0	0	1	1	0
A	B	F																
0	0	1																
0	1	0																
1	0	0																
1	1	0																

Logic Gates

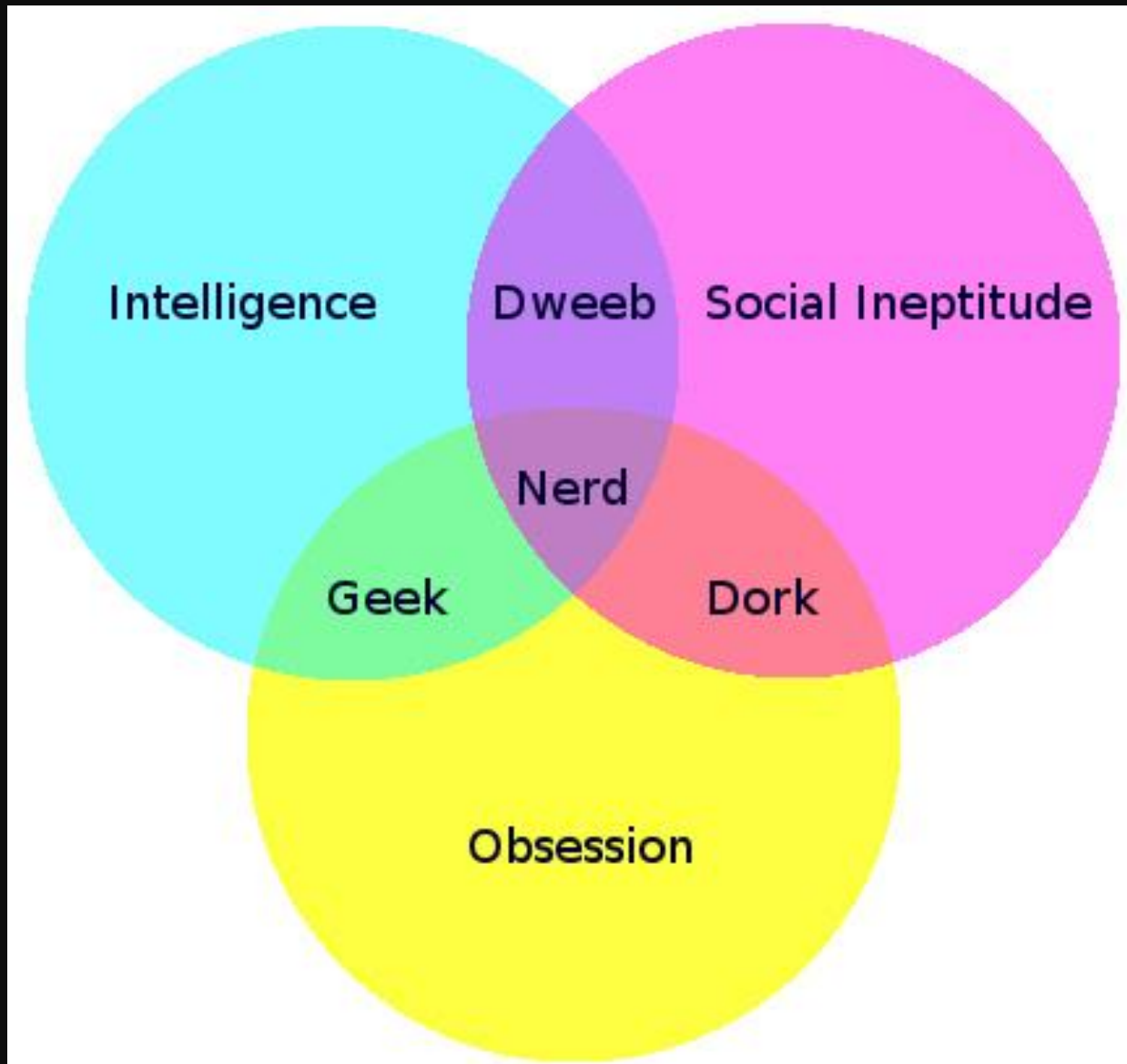


Venn Diagrams

Combining Logic Gates



Nerd Venn Diagram



History of Computer Languages

First Actual Computer Bug

Photo # NH 96566-KN First Computer "Bug", 1945

92

9/9

0800 Antam started
 1000 " stopped - antam ✓
 13⁰⁰ (033) MP-MC ~~1.982447000~~ { 1.2700 9.037 847 025
 (033) PRO 2 2.130476415 } 9.037 846 995 connect
 connect 2.130676415 4.615925059(-2)

Relays 6-2 in 033 failed special speed test
 in relay .. 10.00 test.

Relays changed

1100 Started Cosine Tape (Sine check)
 1525 Started Multi Adder Test.

1545



Relay #70 Panel F
 (moth) in relay.

First actual case of bug being found.
~~1630~~ 1630 antam started.
 1700 closed down.

Relay
 3345
 Relay 3370

Grace Murray Hopper, 1945

Assembly Languages (2GL)

```
CODE:00405CC2 008      lea    eax, [ebp+L_MD5_CTX_Struct]
CODE:00405CC8 008      call  Check_message_digest ; returns 1 if match
CODE:00405CCD 008      mov    ebx, eax
CODE:00405CCF 008      test   bl, bl
CODE:00405CD1 008      jz     short populate_eax
CODE:00405CD3 008      mov    eax, ds:g_prepped_session_key
CODE:00405CD8 008      push  eax
CODE:00405CD9 00C      push  47
CODE:00405CDB 010      lea   eax, [ebp-37h]
CODE:00405CDE 010      push  eax
CODE:00405CDF 014      call  Prep_session_key_and_store_in_global_memory
CODE:00405CE4
CODE:00405CE4      populate_eax:                ; CODE XREF: Login_php+E0j
CODE:00405CE4                ; Login_php+109j
CODE:00405CE4 008      mov    eax, esi
CODE:00405CE6 008      call  @@FreeMem              ; __linkproc__ FreeMem
CODE:00405CEB
CODE:00405CEB      Clear_eax:                   ; CODE XREF: Login_php+A7j
CODE:00405CEB                ; Login_php+DAj
CODE:00405CEB 008      xor    eax, eax
CODE:00405CED 008      pop    edx
CODE:00405CEE 004      pop    ecx
CODE:00405CEF 000      pop    ecx
CODE:00405CF0 -04      mov    fs:[eax], edx
CODE:00405CF3 -04      push  offset cleanup
```

Mnemonic Instructions (CPU Specific)

Fortran, ALGOL, COBOL (3GL)

Program: P R O G 1		Requested by:		Page 2 of 3		
Programmer ROBERT LIEB		Date 1-15-1998		Identification		
Sequence		COBOL Statement				
(Page)	(Set)	A	B			
01		WORKING-STORAGE SECTION.				
02	01	DATA-REMAINS-SWITCH PIC X(02) VALUE SPACES.				
03						
04	01	HEADING-LINE.				
05		05	FILLER			PIC X(10) VALUE SPACES.
06		05	FILLER			PIC X(12) VALUE 'STUDENT NAME'.
07		05	FILLER			PIC X(10) VALUE SPACES.
08						
09	01	DETAIL-LINE.				
10		05	FILLER			PIC X(08) VALUE SPACES
11		05	PRINT-NAME			PIC X(25)
12		05	FILLER			PIC X(10) VALUE SPACES.
13						
14		PROCEDURE DIVISION.				
15		PREPARE-SENIOR-REPORT.				
16		OPEN INPUT STUDENT-FILE				
17		OUTPUT PRINT-FILE.				
18		READ STUDENT-FILE				
19		AT END MOVE 'NO' TO DATA-REMAINS-SWITCH				
20		END-READ.				
21		PERFORM WRITE-HEADING-LINE.				
22		PERFORM PROCESS-RECORDS				
23		UNTIL DATA-REMAINS-SWITCH = 'NO'.				
24		CLOSE STUDENT-FILE				
25		PRINT-FILE.				
26		STOP-RUN.				
27						
28						
29						
30						

COBOL Coding Form

Instruction Explosion, Machine Independence

Modern Programming Languages

- Visual Basic, SQL (4GL): Higher Instruction Explosion, database support
- Lisp, Prolog (5GL): Expert Systems, Artificial Intelligence applications
- C++, Java (Object-Oriented (OOP)): code reuse
- VBScript, JavaScript, PHP (Scripting): web development

Family Tree of Programming Languages

Mother Tongues

Tracing the roots of computer languages through the ages

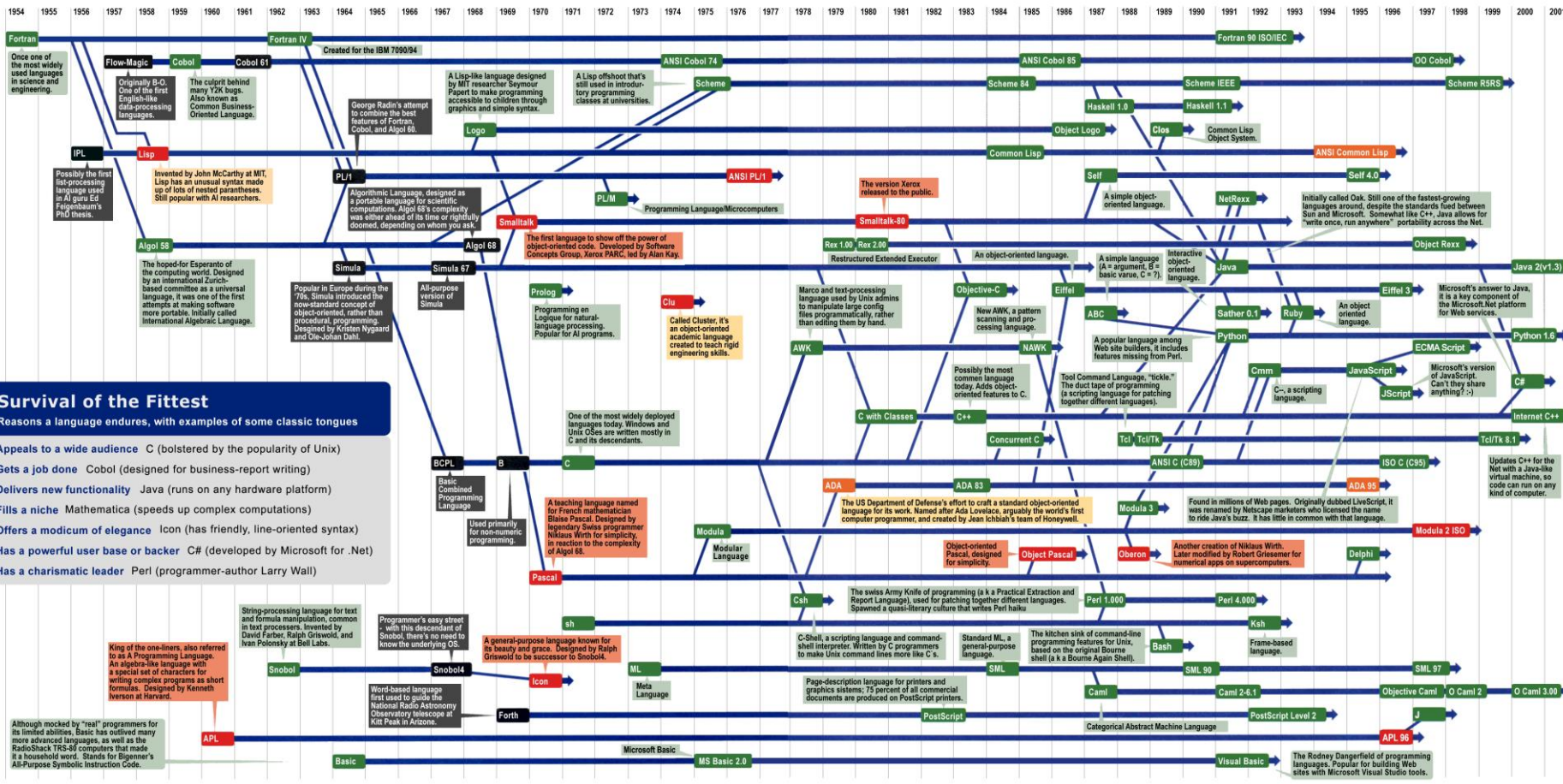
Just like half of the world's spoken tongues, most of the 2,300-plus computer programming languages are either endangered or extinct. As powerhouses C/C++, Visual Basic, Cobol, Java and other modern source codes dominate our systems, hundreds of older languages are running out of life.

An ad hoc collection of engineers-electronic scribes, if you will—will save, or at least document the lingo of classic software. They're combing the globe's 9 million developers in search of coders still fluent in these nearly forgotten lingua frangas. Among the most endangered are Ada, APL, B (the predecessor of C), Lsp, Oberon, Smalltalk, and Simula.

Code-raker Grady Booch, Rational Software's chief scientist, is working with the Computer History Museum in Silicon Valley to record and, in some cases, maintain languages by writing new compilers so our ever-changing hardware can grok the code. Why bother? "They tell us about the state of software practice, the minds of their inventors, and the technical, social, and economic forces that shaped history at the time," Booch explains. "They'll provide the raw material for software archaeologists, historians, and developers to learn what worked, what was brilliant, and what was an utter failure." Here's a peek at the strongest branches of programming's family tree. For a nearly exhaustive rundown, check out the Language List at [HTTP://www.informatik.uni-freiburg.de/java/misc/lang_list.html](http://www.informatik.uni-freiburg.de/java/misc/lang_list.html). - Michael Mendeno

Key

- 1954 Year Introduced
- Active: thousands of users
- Protected: taught at universities; compilers available
- Endangered: usage dropping off
- Extinct: no known active users or up-to-date compilers
- Lineage continues



Sources: Paul Boutin; Brent Halpern, associate director of computer science at IBM Research; The Retrocomputing Museum; Todd Proebsting, senior researcher at Microsoft; Gio Wiederhold, computer scientist, Stanford University

Programming Concepts

Variables	Examples
VARCHAR / CHAR	"John Smith", "TEST", "ABC123"
INTEGER / DECIMAL / FLOAT	1234, 1000101, 3.14159, -400, 10.01, -.001
BOOLEAN	True/False, 0/1
DATE / TIME	03/14/2010 01:59:26
ARRAY	colors{red, orange, yellow, green, blue, indigo, violet}

Conditional Logic

IF / THEN

SWITCH / CASE

FOR LOOP

WHILE LOOP

Sample Program: Bubble Sort

```
var cards=new Array();
cards[0] = 4;
cards[1] = 12; //Queen
cards[2] = 8;
cards[3] = 5;
cards[4] = 11; //Jack
cards[5] = 10;
cards[6] = 14; //Ace
cards[7] = 9;
cards[8] = 13; //King
cards[9] = 6;
cards[10]= 3;
cards[11]= 7;
cards[12]= 2;
```

```
function swapCards(cards, firstIndex, secondIndex)
{
    var temp = cards[firstIndex];
    cards[firstIndex] = cards[secondIndex];
    cards[secondIndex] = temp;
}
```

```
function bubbleSort(cards)
{
    var len = cards.length;
    var cardsSorted = false;

    while (cardsSorted == false)
    {
        cardsSorted = true;

        for (i=0; i < len; i++)
        {
            if (cards[i] > cards[i+1])
            {
                swapCards(cards, i, i+1);
                cardsSorted = false;
            }
        }
    }
    return cards;
}
```

FUNCTION
INTEGER
BOOLEAN
LOOP
IF/THEN
ARRAY

Cut and Paste Programming

- TIC TAC TOE
 - Number Version
 - Hard Conditional Logic Version
 - Intelligent Version
- Game of Life
- Super Mario World in JavaScript

JavaScript Game Resources

- <http://javascript.internet.com/games/>
- http://www.jsmadeeasy.com/javascripts/Games/1ist_test.asp
- <http://www.javascriptkit.com/script/cutindex22.shtml>
- <http://www.javascriptgaming.com/>