Language Design and Evolution

Programming Languages

William Killian

Millersville University

What is a Language?

lan·guage

/ˈlaNGgwij/

noun

1. the method of human communication, either spoken or written, consisting of the use of words in a structured and conventional
2. a system of symbols and rules for writing programs or algorithms

Linguistics Definition

the method of human communication, either spoken or written, consisting of the use of words in a structured and conventional way

- Does “spoken” make sense for computers?
- Is “human communication” feasible with computer programs?

Computer Science Definition

a system of tokens and rules for writing programs or algorithms

- What tokens do we use?
- What rules do we have?

Tokens

- In the domain of language design, we will refer to each word as a unique ***token***
- Tokens can come in any order or be anything, but some tokens might have some special meaning
- Usually alphanumeric, numeric, or symbolic

What tokens might we have in Java? C++?

Keywords

- ***Keywords*** are a type of Token
- Usually reserved by the language
- Only be used in specific locations

Examples:

`int`	in Java
`void`	in C++
`const`	in Javascript
`def`	in Python
`let`	in OCaml

Others?

Symbols

- ***Symbols*** are another type of Token
- Usually combinations of punctuation characters
- Often used to indicate special operations

Examples:

==	in Java
|>	in OCaml
...	in Javascript
<<	in C++

Others?

Identifiers

- ***Identifiers*** refer to specific entities of our program
- Creating a new variable or function
- Accessing a data member
- Calling a function
- Using a library

Rules

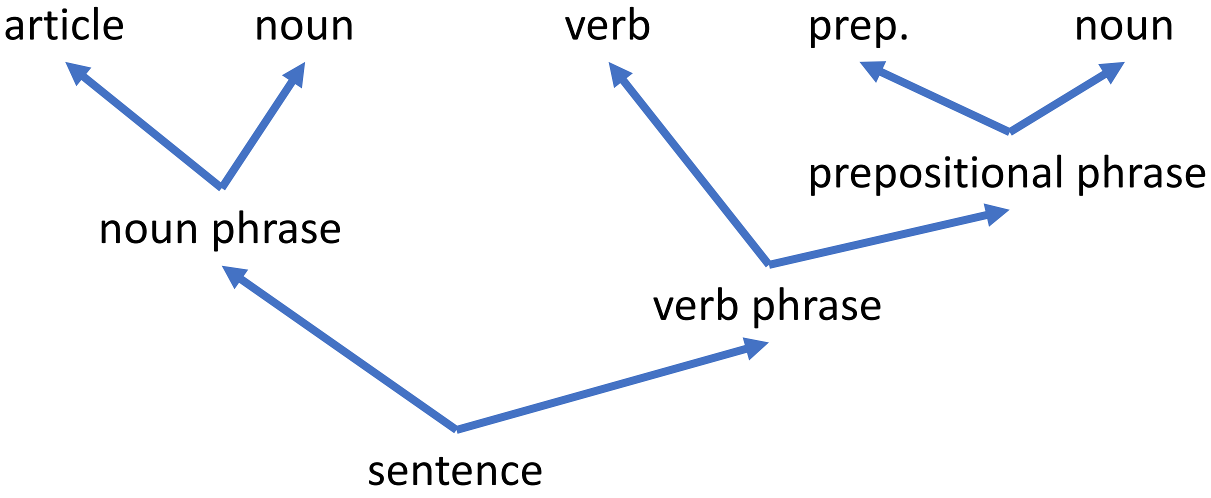
- Within language design, we will refer to the order in which ***symbols*** can be structured as ***rules***
- Two main types of rules:
  - Syntactic Rules

    The ***order*** of all symbols must be well-formed
  - Semantic Rules

    The ***meaning*** of all symbols must be well-formed
- We will talk about these in detail later in the class

Syntax Rules (in English)

The boy went to school.



article noun verb prep. noun

prepositional phrase

noun phrase

verb phrase

sentence

Syntax Rules (in Java)

class := CLASS ID classArg* LBRACE defs* RBRACE

classArg := (IMPLEMENTS|EXTENDS) nameList

nameList := ID [, nameList]*

defs := [visibility] [STATIC] (varDef SEMI| funDef)

visibility := (PUBLIC|PROTECTED|PRIVATE)

varDef := type ID

funDef := type ID LPAREN [paramList] RPAREN …

paramList := varDef [, paramList]*

type := (primitive | ID)

primitive := (INT | BOOLEAN | DOUBLE | SHORT | LONG | BYTE | FLOAT)

Syntax Rules (in Java)

CAPS means terminal (symbol)

italicized means rule

* - zero or more

[optional]

(choice1|choice2)

class := **CLASS ID** _classArg_* **LBRACE** _defs_* **RBRACE**

classArg := (**IMPLEMENTS**|**EXTENDS**) _nameList_

nameList := **ID** [, _nameList_]*

defs := [_visibility_] [**STATIC**] (_varDef_ **SEMI**| _funDef_)

visibility := (**PUBLIC**|**PROTECTED**|**PRIVATE**)

varDef := _type_ **ID**

funDef := _type_ **ID LPAREN** [_paramList_] **RPAREN …**

paramList := _varDef_ [, _paramList_]*

type := (_primitive_ | **ID**)

primitive := (**INT** | **BOOLEAN** | **DOUBLE** | **SHORT** | **LONG** | **BYTE** | **FLOAT**)

Syntax Rules (in Java)

class := **CLASS ID** *classArg** **LBRACE** *defs** **RBRACE**

classArg := (**IMPLEMENTS**|**EXTENDS**) *nameList*

nameList := **ID** [, *nameList*]*

defs := [*visibility*] [**STATIC**] (*varDef* **SEMI**| *funDef*)

visibility := (**PUBLIC**|**PROTECTED**|**PRIVATE**)

varDef := *type* **ID**

funDef := *type* **ID LPAREN** [*paramList*] **RPAREN …**

paramList := *varDef* [, *paramList*]*

type := (*primitive* | **ID**)

primitive := (**INT** | **BOOLEAN** | **DOUBLE** | **SHORT** | **LONG** | **BYTE** | **FLOAT**)

What are the **tokens** in the shown rules?

Syntax Rules (in Java)

class := **CLASS ID** _classArg_* **LBRACE** _defs_* **RBRACE**

classArg := (**IMPLEMENTS|EXTENDS**) _nameList_

nameList := **ID** [, _nameList_]*

defs := [_visibility_] [**STATIC**] (_varDef_ **SEMI**| _funDef_)

visibility := (**PUBLIC|PROTECTED|PRIVATE**)

varDef := _type_ **ID**

funDef := _type_ **ID LPAREN** [_paramList_] **RPAREN** …

paramList := _varDef_ [, _paramList_]*

type := (_primitive_ | **ID**)

primitive := (**INT** | **BOOLEAN** | **DOUBLE** | **SHORT** | **LONG** | **BYTE** | **FLOAT**)

Where are the rules?

Syntax Rules (in Java)

class := **CLASS ID** _classArg_* **LBRACE** _defs_* **RBRACE**

classArg := (**IMPLEMENTS**|**EXTENDS**) _nameList_

nameList := **ID** [, _nameList_]*

defs := [_visibility_] [**STATIC**] (_varDef_ **SEMI**| _funDef_)

visibility := (**PUBLIC**|**PROTECTED**|**PRIVATE**)

varDef := _type_ **ID**

funDef := _type_ **ID LPAREN** [_paramList_] **RPAREN …**

paramList := _varDef_ [, _paramList_]*

type := (_primitive_ | **ID**)

primitive := (**INT** | **BOOLEAN** | **DOUBLE** | **SHORT** | **LONG** | **BYTE** | **FLOAT**)

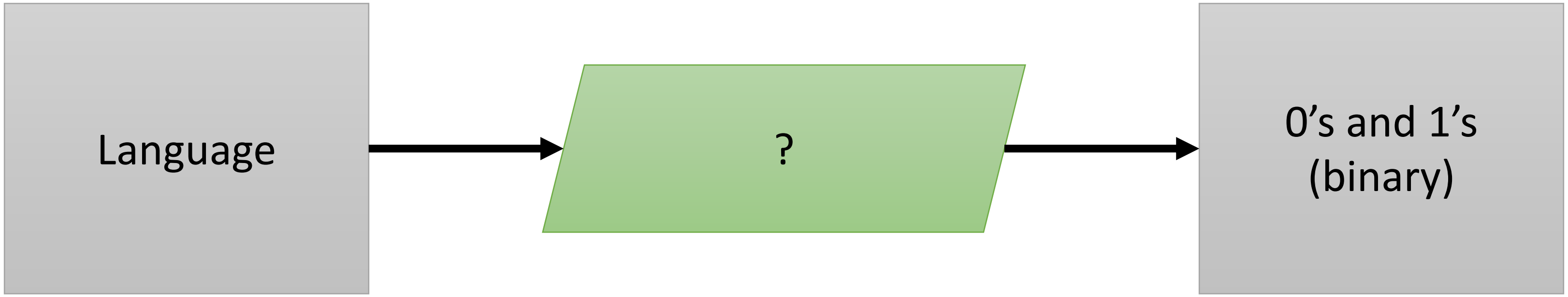
Exercise: Make a “Point” class with two public members of type double: x and y?

Rosetta Code Examples (Homework)

- What problems did you investigate?
- What languages did you see?
- What languages **DON’T** you want to see again?

Language Transformation

- Computers understand binary
  - Sequence of 0's & 1's
  - See: Computer Architecture
- Humans understand *languages*
  - See: Programming Languages



We need a *tool* to translate our language to binary

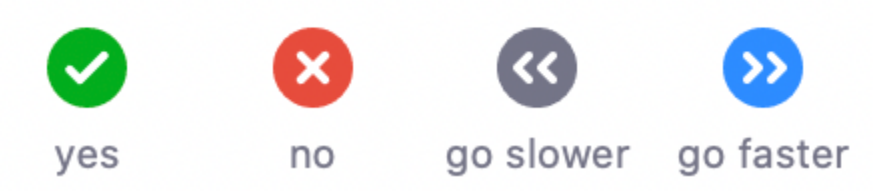
Compilers

- Input:
  - A program (sequence of instructions) written in a well-defined, predictable language
- Output:
  - A sequence of bits that a computer architecture can execute
- Task:
  - Compilers **translate** from an *input language* to an *output language* without loss of functionality
  - Mathematically correct

The First Programming Language

Discussion:

- What year do you think the first programming language came out?
- What did it look like?
- What did it run on?



Before 1930

1930 – 1940

1940 – 1950

After 1950

The First Program~~ming Language~~

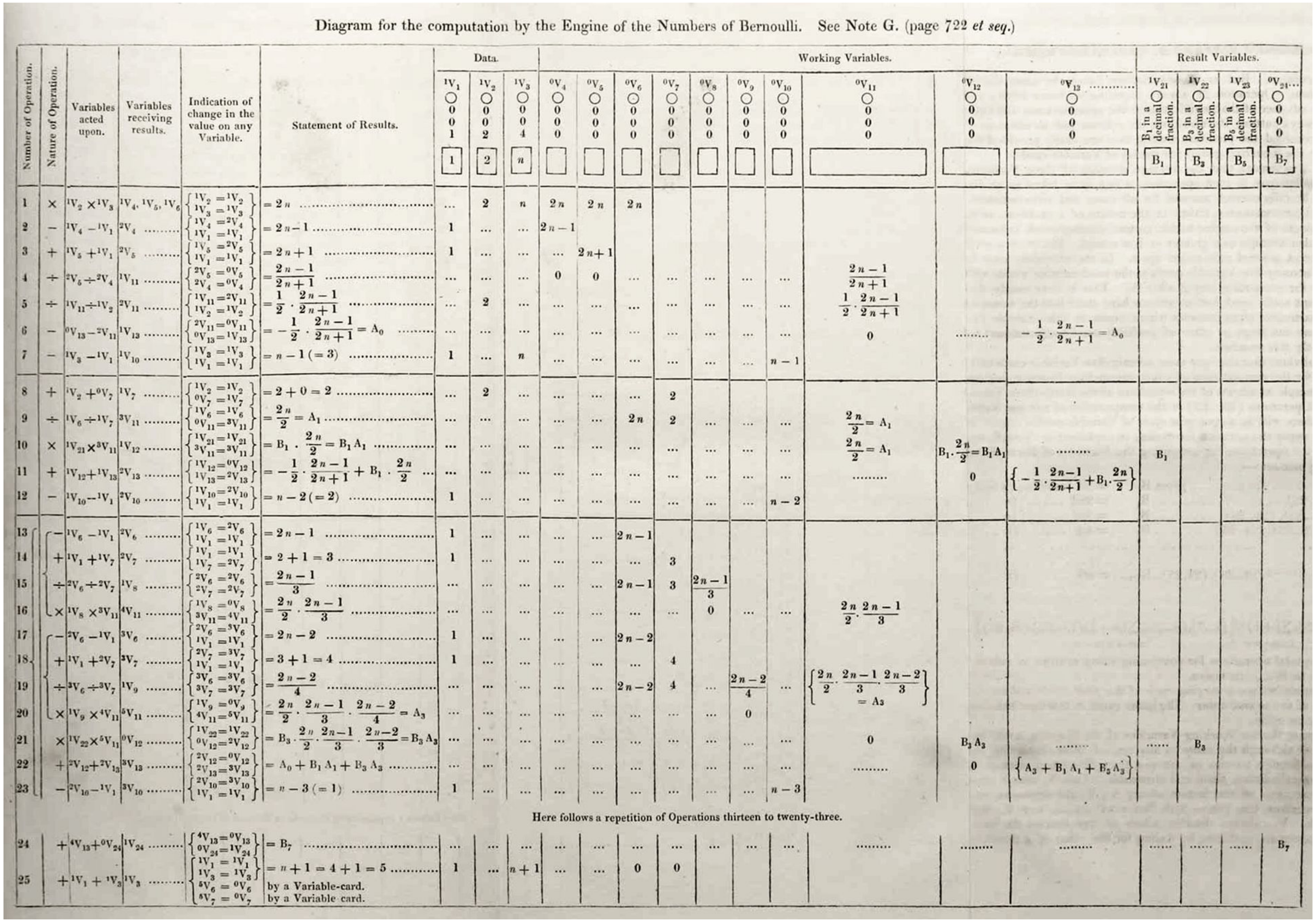


Diagram for the computation by the Engine of the Numbers of Bernoulli. See Note G. (page 722 *et seq.*)

Number of Operation.	Nature of Operation.	Variables acted upon.	Variables receiving results.	Indication of change in the value on any Variable.	Statement of Results.	Data.			Working Variables.										Result Variables.			
						$^1V_1$ ○ 0 0 0 1	$^1V_2$ ○ 0 0 0 2	$^1V_3$ ○ 0 0 0 4	$^0V_4$ ○ 0 0 0 0	$^0V_5$ ○ 0 0 0 0	$^0V_6$ ○ 0 0 0 0	$^0V_7$ ○ 0 0 0 0	$^0V_8$ ○ 0 0 0 0	$^0V_9$ ○ 0 0 0 0	$^0V_{10}$ ○ 0 0 0 0	$^0V_{11}$ ○ 0 0 0 0	$^0V_{12}$ ○ 0 0 0 0	$^0V_{13}$ .... ○ 0 0 0 0	$^1V_{21}$ ○ B_1 in a decimal fraction.	$^1V_{22}$ ○ B_3 in a decimal fraction.	$^1V_{23}$ ○ B_5 in a decimal fraction.	$^0V_{24}$... ○ 0 0 0
						1	2	n											B_1	B_3	B_5	B_7
1	×	$^1V_2 \times {}^1V_3$	$^1V_4, {}^1V_5, {}^1V_6$	$^1V_2 = {}^1V_2$ $^1V_3 = {}^1V_3$	$= 2n$	...	2	n	$2n$	$2n$	$2n$											
2	−	$^1V_4 - {}^1V_1$	$^2V_4$	$^1V_4 = {}^2V_4$ $^1V_1 = {}^1V_1$	$= 2n - 1$	1	...	...	$2n-1$													
3	+	$^1V_5 + {}^1V_1$	$^2V_5$	$^1V_5 = {}^2V_5$ $^1V_1 = {}^1V_1$	$= 2n + 1$	1	...	...	...	$2n+1$												
4	÷	$^2V_5 \div {}^2V_4$	$^1V_{11}$	$^2V_5 = {}^0V_5$ $^2V_4 = {}^0V_4$	$= \frac{2n-1}{2n+1}$	...	...	...	0	0	...	...	...	...	...	$\frac{2n-1}{2n+1}$						
5	÷	$^1V_{11} \div {}^1V_2$	$^2V_{11}$	$^1V_{11} = {}^2V_{11}$ $^1V_2 = {}^1V_2$	$= \frac{1}{2} \cdot \frac{2n-1}{2n+1}$	...	2	...	...	...	...	...	...	...	...	$\frac{1}{2} \cdot \frac{2n-1}{2n+1}$						
6	−	$^0V_{13} - {}^2V_{11}$	$^1V_{13}$	$^2V_{11} = {}^0V_{11}$ $^0V_{13} = {}^1V_{13}$	$= -\frac{1}{2} \cdot \frac{2n-1}{2n+1} = A_0$	...	...	...	...	...	...	...	...	...	...	0	....	$-\frac{1}{2} \cdot \frac{2n-1}{2n+1} = A_0$				
7	−	$^1V_3 - {}^1V_1$	$^1V_{10}$	$^1V_3 = {}^1V_3$ $^1V_1 = {}^1V_1$	$= n - 1 \, (= 3)$	1	...	n	...	...	...	...	...	...	$n-1$							
8	+	$^1V_2 + {}^0V_7$	$^1V_7$	$^1V_2 = {}^1V_2$ $^0V_7 = {}^1V_7$	$= 2 + 0 = 2$	...	2	...	...	...	...	2										
9	÷	$^1V_6 \div {}^1V_7$	$^3V_{11}$	$^1V_6 = {}^1V_6$ $^0V_{11} = {}^3V_{11}$	$= \frac{2n}{2} = A_1$	...	...	...	...	...	$2n$	2	...	...	...	$\frac{2n}{2} = A_1$						
10	×	$^1V_{21} \times {}^3V_{11}$	$^1V_{12}$	$^1V_{21} = {}^1V_{21}$ $^3V_{11} = {}^3V_{11}$	$= B_1 \cdot \frac{2n}{2} = B_1 A_1$	...	...	...	...	...	...	...	...	...	...	$\frac{2n}{2} = A_1$	$B_1 \cdot \frac{2n}{2} = B_1 A_1$	....	B_1			
11	+	$^1V_{12} + {}^1V_{13}$	$^2V_{13}$	$^1V_{12} = {}^0V_{12}$ $^1V_{13} = {}^2V_{13}$	$= -\frac{1}{2} \cdot \frac{2n-1}{2n+1} + B_1 \cdot \frac{2n}{2}$	...	...	...	...	...	...	...	...	...	...	....	0	$\left\{-\frac{1}{2} \cdot \frac{2n-1}{2n+1} + B_1 \cdot \frac{2n}{2}\right\}$				
12	−	$^1V_{10} - {}^1V_1$	$^2V_{10}$	$^1V_{10} = {}^2V_{10}$ $^1V_1 = {}^1V_1$	$= n - 2 \, (= 2)$	1	...	...	...	...	...	...	...	...	$n-2$							
13	−	$^1V_6 - {}^1V_1$	$^2V_6$	$^1V_6 = {}^2V_6$ $^1V_1 = {}^1V_1$	$= 2n - 1$	1	...	...	...	...	$2n-1$											
14	+	$^1V_1 + {}^1V_7$	$^2V_7$	$^1V_1 = {}^1V_1$ $^1V_7 = {}^2V_7$	$= 2 + 1 = 3$	1	...	...	...	...	...	3										
15	÷	$^2V_6 \div {}^2V_7$	$^1V_8$	$^2V_6 = {}^2V_6$ $^2V_7 = {}^2V_7$	$= \frac{2n-1}{3}$	...	...	...	...	...	$2n-1$	3	$\frac{2n-1}{3}$									
16	×	$^1V_8 \times {}^3V_{11}$	$^4V_{11}$	$^1V_8 = {}^0V_8$ $^3V_{11} = {}^4V_{11}$	$= \frac{2n}{2} \cdot \frac{2n-1}{3}$	...	...	...	...	...	...	...	0	...	...	$\frac{2n}{2} \cdot \frac{2n-1}{3}$						
17	−	$^2V_6 - {}^1V_1$	$^3V_6$	$^2V_6 = {}^3V_6$ $^1V_1 = {}^1V_1$	$= 2n - 2$	1	...	...	...	...	$2n-2$											
18	+	$^1V_1 + {}^2V_7$	$^3V_7$	$^2V_7 = {}^3V_7$ $^1V_1 = {}^1V_1$	$= 3 + 1 = 4$	1	...	...	...	...	...	4										
19	÷	$^3V_6 \div {}^3V_7$	$^1V_9$	$^3V_6 = {}^3V_6$ $^3V_7 = {}^3V_7$	$= \frac{2n-2}{4}$	...	...	...	...	...	$2n-2$	4	...	$\frac{2n-2}{4}$	...	$\left\{\frac{2n}{2} \cdot \frac{2n-1}{3} \cdot \frac{2n-2}{3} = A_3\right\}$						
20	×	$^1V_9 \times {}^4V_{11}$	$^5V_{11}$	$^1V_9 = {}^0V_9$ $^4V_{11} = {}^5V_{11}$	$= \frac{2n}{2} \cdot \frac{2n-1}{3} \cdot \frac{2n-2}{4} = A_3$	...	...	...	...	...	...	...	...	0								
21	×	$^1V_{22} \times {}^5V_{11}$	$^0V_{12}$	$^1V_{22} = {}^1V_{22}$ $^0V_{12} = {}^2V_{12}$	$= B_3 \cdot \frac{2n}{2} \cdot \frac{2n-1}{3} \cdot \frac{2n-2}{3} = B_3 A_3$	...	...	...	...	...	...	...	...	...	...	0	$B_3 A_3$	....	......	B_3		
22	+	$^2V_{12} + {}^2V_{13}$	$^3V_{13}$	$^2V_{12} = {}^0V_{12}$ $^2V_{13} = {}^3V_{13}$	$= A_0 + B_1 A_1 + B_3 A_3$	...	...	...	...	...	...	...	...	...	...	....	0	$\left\{A_3 + B_1 A_1 + B_3 A_3\right\}$				
23	−	$^2V_{10} - {}^1V_1$	$^3V_{10}$	$^2V_{10} = {}^3V_{10}$ $^1V_1 = {}^1V_1$	$= n - 3 \, (= 1)$	1	...	...	...	...	...	...	...	...	$n-3$							

Here follows a repetition of Operations thirteen to twenty-three.

24	+	$^4V_{13} + {}^0V_{24}$	$^1V_{24}$	$^4V_{13} = {}^0V_{13}$ $^0V_{24} = {}^1V_{24}$	$= B_7$	...	...	...	...	...	...	...	...	...	...	....	....	....	......	......	......	B_7
25	+	$^1V_1 + {}^1V_3$	$^1V_3$	$^1V_1 = {}^1V_1$ $^1V_3 = {}^1V_3$ $^5V_6 = {}^0V_6$ $^5V_7 = {}^0V_7$	$= n + 1 = 4 + 1 = 5$ by a Variable-card. by a Variable card.	1	...	$n+1$	...	...	0	0										

Ada Lovelace, 1843

The First Programming Language

Created

Plankalkül (1942-1945)

- Konrad Zuse

```
P1 max3 (V0[:8.0],V1[:8.0],V2[:8.0]) → R0[:8.0]
max(V0[:8.0],V1[:8.0]) → Z1[:8.0]
max(Z1[:8.0],V2[:8.0]) → R0[:8.0]
END
P2 max (V0[:8.0],V1[:8.0]) → R0[:8.0]
V0[:8.0] → Z1[:8.0]
(Z1[:8.0] < V1[:8.0]) → V1[:8.0] → Z1[:8.0]
Z1[:8.0] → R0[:8.0]
END
```

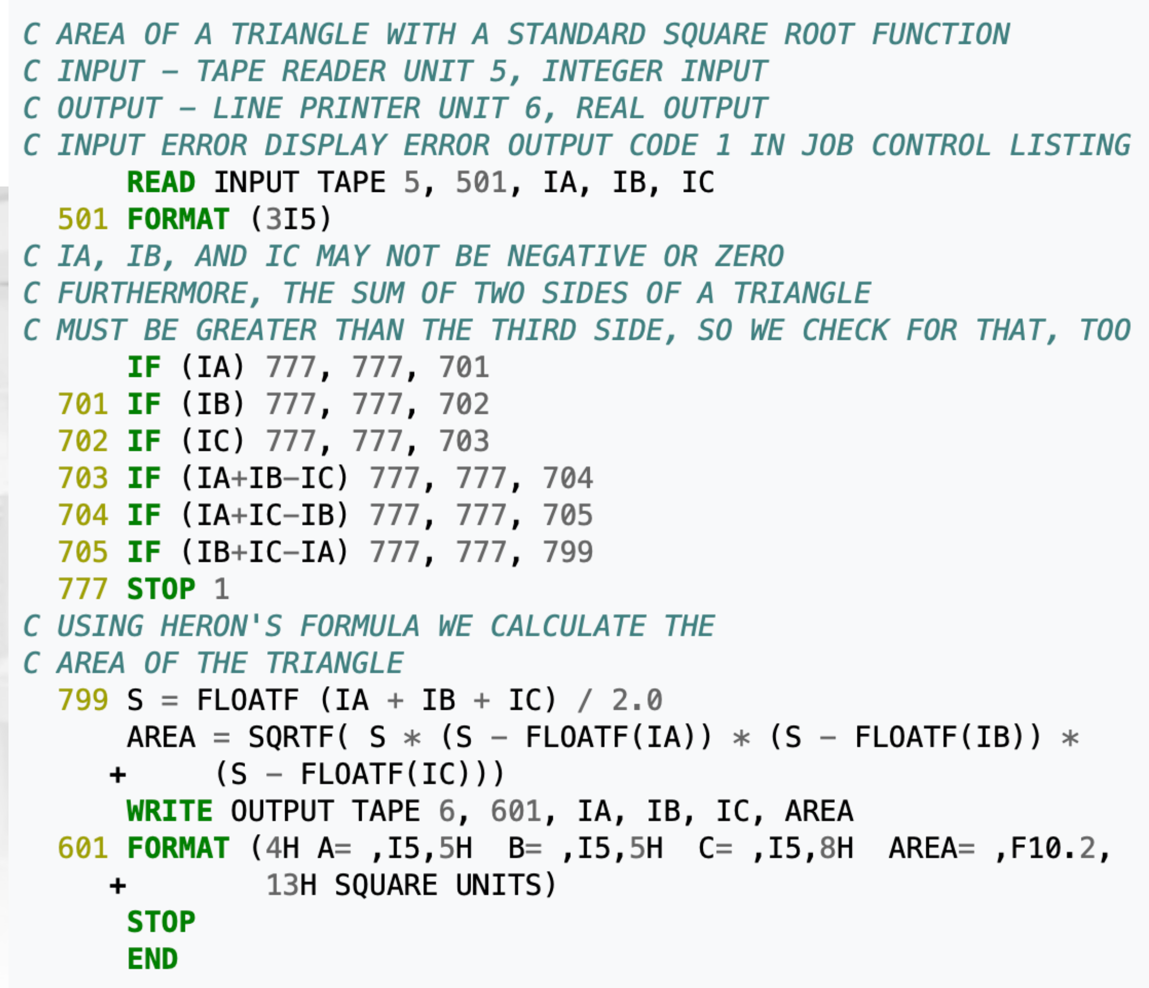
*function **max3** (in a linear transcription) that calculates the maximum of three variables*

The First Programming Language

Commercial

FORTRAN (1954)

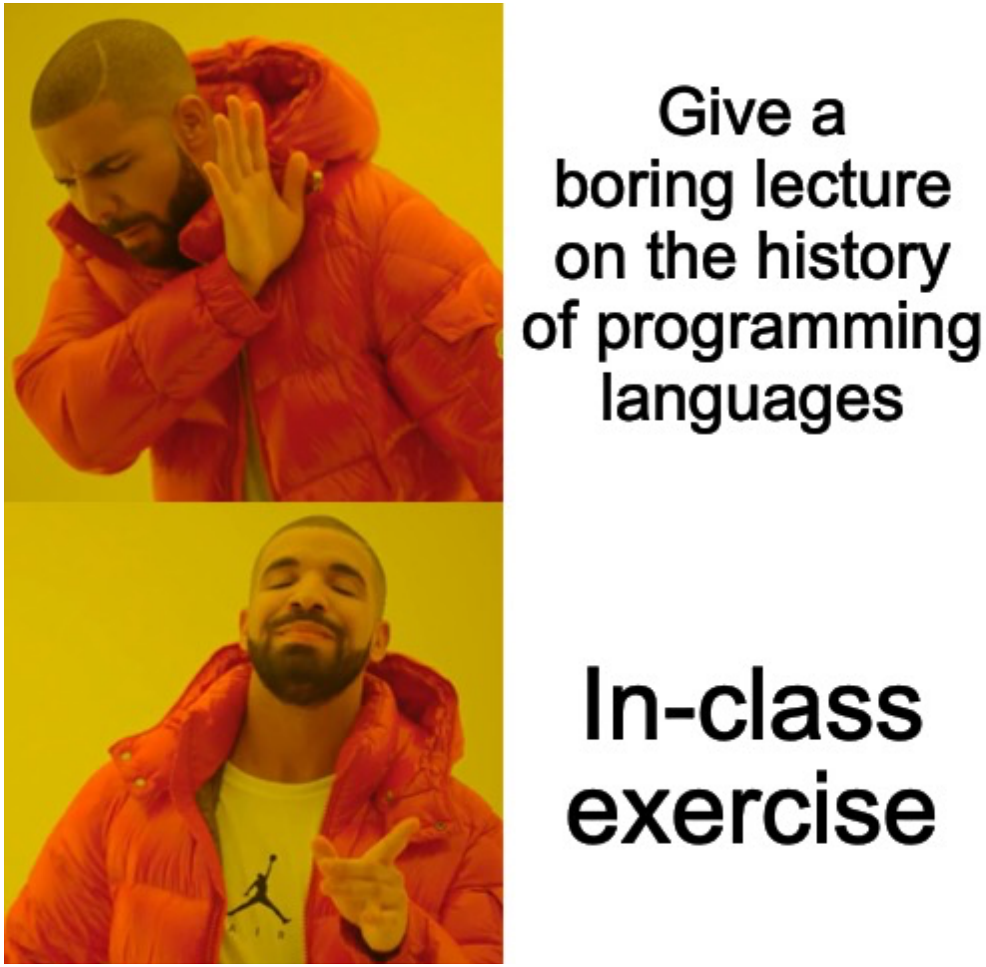
- John Backus + IBM



```
C AREA OF A TRIANGLE WITH A STANDARD SQUARE ROOT FUNCTION
C INPUT - TAPE READER UNIT 5, INTEGER INPUT
C OUTPUT - LINE PRINTER UNIT 6, REAL OUTPUT
C INPUT ERROR DISPLAY ERROR OUTPUT CODE 1 IN JOB CONTROL LISTING
     READ INPUT TAPE 5, 501, IA, IB, IC
 501 FORMAT (3I5)
C IA, IB, AND IC MAY NOT BE NEGATIVE OR ZERO
C FURTHERMORE, THE SUM OF TWO SIDES OF A TRIANGLE
C MUST BE GREATER THAN THE THIRD SIDE, SO WE CHECK FOR THAT, TOO
     IF (IA) 777, 777, 701
 701 IF (IB) 777, 777, 702
 702 IF (IC) 777, 777, 703
 703 IF (IA+IB-IC) 777, 777, 704
 704 IF (IA+IC-IB) 777, 777, 705
 705 IF (IB+IC-IA) 777, 777, 799
 777 STOP 1
C USING HERON'S FORMULA WE CALCULATE THE
C AREA OF THE TRIANGLE
 799 S = FLOATF (IA + IB + IC) / 2.0
     AREA = SQRTF( S * (S - FLOATF(IA)) * (S - FLOATF(IB)) *
    +     (S - FLOATF(IC)))
     WRITE OUTPUT TAPE 6, 601, IA, IB, IC, AREA
 601 FORMAT (4H A= ,I5,5H  B= ,I5,5H  C= ,I5,8H  AREA= ,F10.2,
    +        13H SQUARE UNITS)
     STOP
     END
```

What Was Your First Programming Language?

History of Programming Languages


Give a boring lecture on the history of programming languages
In-class exercise

History of Programming Languages

Split up into four groups (randomly)

- **Group 1:** 1960 – 1980
- **Group 2:** 1980 – 1995
- **Group 3:** 1995 – 2010
- **Group 4:** 2010 – Present

History of Programming Languages

1. Spend approximately 20 minutes in your group searching the internet (Google, Wikipedia).
2. Identify:
    - What languages seem important (that you've heard of)?
    - What is the "coolest" language?
    - What is the origin of the language?
        - Research (PhD)
        - Industry (IBM, Kodak, HP, Apple, Google, Microsoft, etc)
        - Hobby (someone's fun project)
    - Trends of languages during the time period

History of Programming Languages

3. Regroup after 20 minutes
4. Have each group give a short presentation with your findings
    - Don’t worry -- I’ll help you out