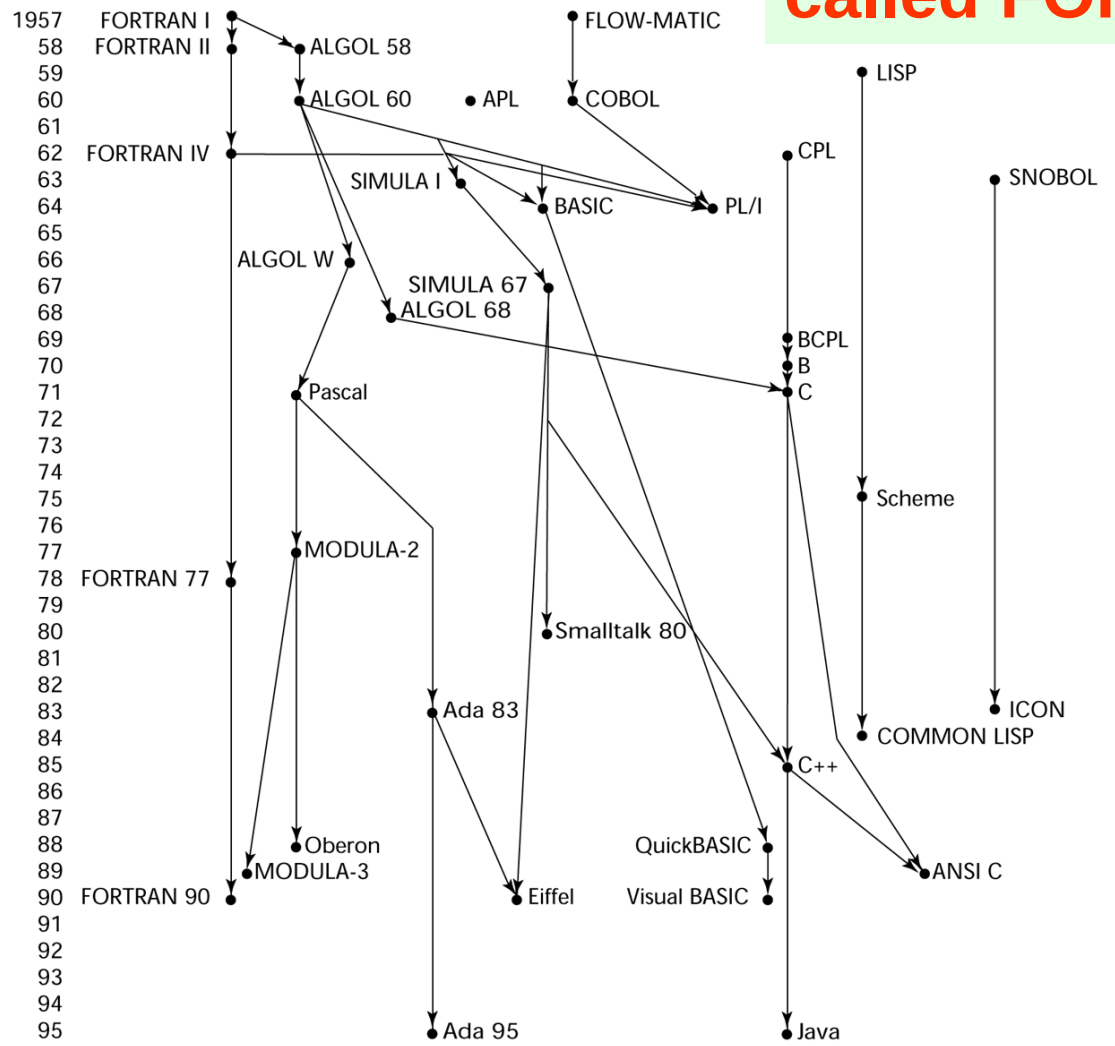
História das Linguagens de Programação

Rosana Braga

Links interessantes

- Timeline das linguagens de programação
  - https://en.wikipedia.org/wiki/Timeline_of_programming_languages
- Crash course computer science
  - Early programming - episódio 10
    - https://www.youtube.com/watch?v=nwDq4adJwzM&list=PL8dPuuaLjXtNlUrzyH5r6jN9ulIgZBpdo&index=11
  - First programming languages – episódio 11
    - https://www.youtube.com/watch?v=RU1u-js7db8&index=12&list=PL8dPuuaLjXtNlUrzyH5r6jN9ulIgZBpdo

» I don't know what the language of the year 2000 will look like, but I know it will be called FORTRAN « **C.A.R. Hoare**



Mas... E antes de termos linguagens?

Algoritmo – Ano 825

- Abdullah al-Khwarizmi (Persia)
- *Algoritmi de numero Indorum* →procedimento para realização de cálculos com os algarismos indianos (que tornaram-se os algarismos arábicos)
- Bem mais fácil do que somar números romanos!

A primeira programadora: Ada Lovelace

- Interessou-se pelo trabalho de Babbage (máquina analítica)
- Ao traduzir um artigo de francês para inglês adicionou várias notas que mostraram que ela entendeu o funcionamento da máquina
- Conceito de procedimento: sequência de cartões (programa) independente dos valores operados
- Noção de símbolos e variáveis de memória
- Noção da atribuição de valores a variáveis
- Algoritmo para computar a sequencia de Bernoulli: primeiro programa?

A primeira programadora: Ada Lovelace

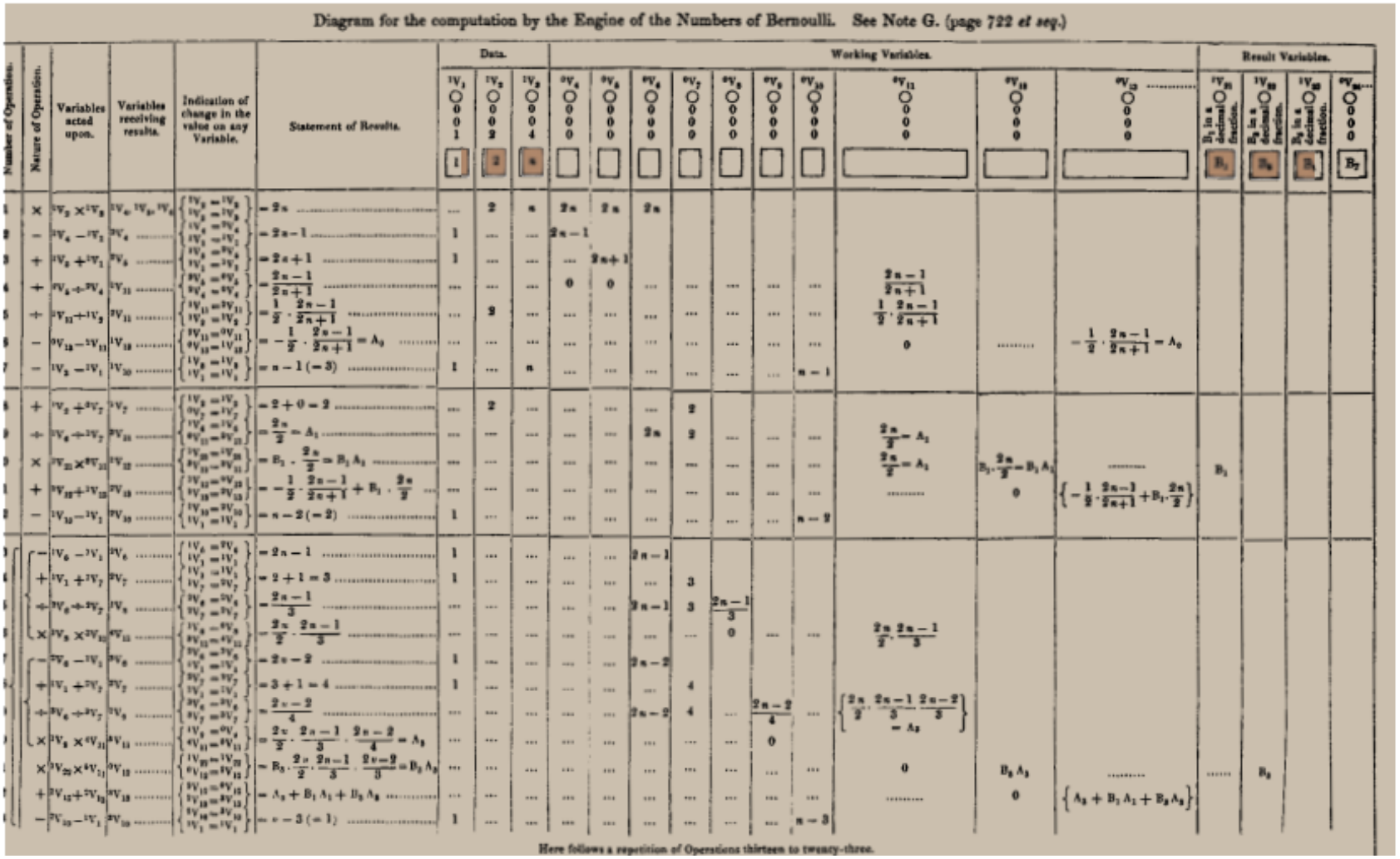


Diagram for the computation by the Engine of the Numbers of Bernoulli. See Note G. (page 722 *et seq.*)

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.			
						$^1V_1$ 1	$^1V_2$ 2	$^1V_3$ 4	$^0V_4$	$^0V_5$	$^0V_6$	$^0V_7$	$^0V_8$	$^0V_9$	$^0V_{10}$	$^0V_{11}$	$^0V_{12}$	$^0V_{13}$	$^0V_{14}$	$^1V_{20}$ B_1	$^1V_{22}$ B_3	$^1V_{24}$ B_5	$^0V_{26}$... B_7
1	×	$^2V_2 \times {}^2V_3$	$^2V_4$, $^2V_5$, $^2V_6$	$\left\{ {}^1V_2 = {}^1V_2 \atop {}^1V_3 = {}^1V_3 \right\}$	$= 2n$	...	2	n	$2n$	$2n$	$2n$											
2	−	$^2V_4 - {}^2V_1$	$^2V_4$	$\left\{ {}^1V_2 = {}^1V_2 \atop {}^1V_3 = {}^1V_3 \right\}$	$= 2n-1$	1	...	...	$2n-1$													
3	+	$^2V_3 + {}^2V_1$	$^2V_5$	$\left\{ {}^1V_2 = {}^1V_2 \atop {}^1V_3 = {}^1V_3 \right\}$	$= 2n+1$	1	...	...	...	$2n+1$												
4	+	$^2V_4 + {}^2V_4$	$^2V_{11}$	$\left\{ {}^2V_4 = {}^2V_4 \atop {}^2V_4 = {}^2V_4 \right\}$	$= \frac{2n-1}{2n+1}$	...	...	...	0	0	...	...	...	...	...	$\frac{2n-1}{2n+1}$						
5	+	$^2V_{11} + {}^1V_3$	$^2V_{11}$	$\left\{ {}^2V_{11} = {}^2V_{11} \atop {}^1V_3 = {}^1V_3 \right\}$	$= \frac{1}{2} \cdot \frac{2n-1}{2n+1}$	...	2	...	...	...	...	...	...	...	...	$\frac{1}{2} \cdot \frac{2n-1}{2n+1}$						
6	−	$^2V_{12} - {}^2V_{13}$	$^1V_{12}$	$\left\{ {}^2V_{11} = {}^2V_{11} \atop {}^1V_{13} = {}^1V_{13} \right\}$	$= -\frac{1}{2} \cdot \frac{2n-1}{2n+1} = A_0$	...	...	...	...	...	...	...	...	...	...	0	.....	$-\frac{1}{2} \cdot \frac{2n-1}{2n+1} = A_0$				
7	−	$^1V_2 - {}^1V_1$	$^1V_{30}$	$\left\{ {}^1V_2 = {}^1V_2 \atop {}^1V_1 = {}^1V_1 \right\}$	$= n-1\ (=3)$	1	...	n	...	...	...	...	...	...	$n-1$							
8	+	$^2V_2 + {}^2V_7$	$^2V_7$	$\left\{ {}^1V_2 = {}^1V_2 \atop {}^0V_7 = {}^1V_7 \right\}$	$= 2+0=2$	...	2	...	...	...	...	2										
9	÷	$^2V_4 \div {}^2V_7$	$^2V_{21}$	$\left\{ {}^1V_2 = {}^1V_2 \atop {}^0V_{21} = {}^0V_{21} \right\}$	$= \frac{2n}{2} = A_1$	...	...	...	...	...	$2n$	2	...	...	...	$\frac{2n}{2} = A_1$						
9	×	$^2V_{21} \times {}^2V_{22}$	$^2V_{12}$	$\left\{ {}^2V_{20} = {}^2V_{20} \atop {}^0V_{22} = {}^0V_{22} \right\}$	$= B_1 \cdot \frac{2n}{2} = B_1 A_1$	...	...	...	...	...	...	...	...	...	...	$\frac{2n}{2} = A_1$	$B_1 \cdot \frac{2n}{2} = B_1 A_1$	.....	B_1			
1	+	$^2V_{12} + {}^2V_{13}$	$^2V_{13}$	$\left\{ {}^2V_{12} = {}^2V_{12} \atop {}^2V_{13} = {}^2V_{13} \right\}$	$= -\frac{1}{2} \cdot \frac{2n-1}{2n+1} + B_1 \cdot \frac{2n}{2}$	...	...	...	...	...	...	...	...	...	...	.....	0	$\left\{ -\frac{1}{2} \cdot \frac{2n-1}{2n+1} + B_1 \cdot \frac{2n}{2} \right\}$				
2	−	$^1V_{13} - {}^1V_1$	$^1V_{31}$	$\left\{ {}^1V_{13} = {}^2V_{13} \atop {}^1V_1 = {}^1V_1 \right\}$	$= n-2\ (=2)$	1	...	...	...	...	...	...	...	...	$n-2$							
3	−	$^2V_6 - {}^2V_1$	$^2V_6$	$\left\{ {}^1V_2 = {}^2V_6 \atop {}^1V_1 = {}^1V_1 \right\}$	$= 2n-1$	1	...	...	...	...	$2n-1$											
4	+	$^2V_1 + {}^2V_7$	$^2V_7$	$\left\{ {}^1V_2 = {}^1V_2 \atop {}^1V_7 = {}^1V_7 \right\}$	$= 2+1=3$	1	...	...	...	...	...	3										
5	÷	$^2V_6 \div {}^2V_7$	$^2V_8$	$\left\{ {}^2V_6 = {}^2V_6 \atop {}^2V_7 = {}^2V_7 \right\}$	$= \frac{2n-1}{3}$	...	...	...	...	...	$2n-1$	3	$\frac{2n-1}{3}$									
6	×	$^2V_8 \times {}^2V_{12}$	$^2V_{13}$	$\left\{ {}^2V_8 = {}^2V_8 \atop {}^2V_{12} = {}^2V_{12} \right\}$	$= \frac{2n}{2} \cdot \frac{2n-1}{3}$	...	...	...	...	...	...	...	0	...	...	$\frac{2n}{2} \cdot \frac{2n-1}{3}$						
7	−	$^2V_6 - {}^2V_1$	$^2V_6$	$\left\{ {}^1V_2 = {}^2V_6 \atop {}^1V_1 = {}^1V_1 \right\}$	$= 2n-2$	1	...	...	...	...	$2n-2$											
8	+	$^2V_1 + {}^2V_7$	$^2V_7$	$\left\{ {}^1V_2 = {}^1V_2 \atop {}^1V_7 = {}^1V_7 \right\}$	$= 3+1=4$	1	...	...	...	...	...	4										
9	÷	$^2V_6 \div {}^2V_7$	$^2V_9$	$\left\{ {}^2V_6 = {}^2V_6 \atop {}^2V_7 = {}^2V_7 \right\}$	$= \frac{2n-2}{4}$	...	...	...	...	...	$2n-2$	4	...	$\frac{2n-2}{4}$	...	$\left\{ \frac{2n}{2} \cdot \frac{2n-1}{3} \cdot \frac{2n-2}{3} = A_2 \right\}$						
1	×	$^2V_8 \times {}^2V_{11}$	$^2V_{14}$	$\left\{ {}^2V_8 = {}^2V_8 \atop {}^2V_{11} = {}^2V_{11} \right\}$	$= \frac{2n}{2} \cdot \frac{2n-1}{3} \cdot \frac{2n-2}{4} = A_2$	...	...	...	...	...	...	...	...	0	...							
2	×	$^2V_{23} \times {}^2V_{11}$	$^2V_{15}$	$\left\{ {}^1V_{20} = {}^1V_{20} \atop {}^2V_{15} = {}^2V_{15} \right\}$	$= B_2 \cdot \frac{2n}{2} \cdot \frac{2n-1}{3} \cdot \frac{2n-2}{4} = B_2 A_2$	...	...	...	...	...	...	...	...	...	...	0	$B_2 A_2$	.....	.....	B_2		
3	+	$^2V_{15} + {}^2V_{12}$	$^2V_{16}$	$\left\{ {}^2V_{15} = {}^2V_{15} \atop {}^2V_{16} = {}^2V_{16} \right\}$	$= A_2 + B_1 A_1 + B_2 A_2$	...	...	...	...	...	...	...	...	...	...	.....	0	$\left\{ A_2 + B_1 A_1 + B_2 A_2 \right\}$				
4	−	$^2V_{16} - {}^1V_1$	$^2V_{32}$	$\left\{ {}^2V_{16} = {}^2V_{16} \atop {}^1V_1 = {}^1V_1 \right\}$	$= n-3\ (=1)$	1	...	...	...	...	...	...	...	...	$n-3$							

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

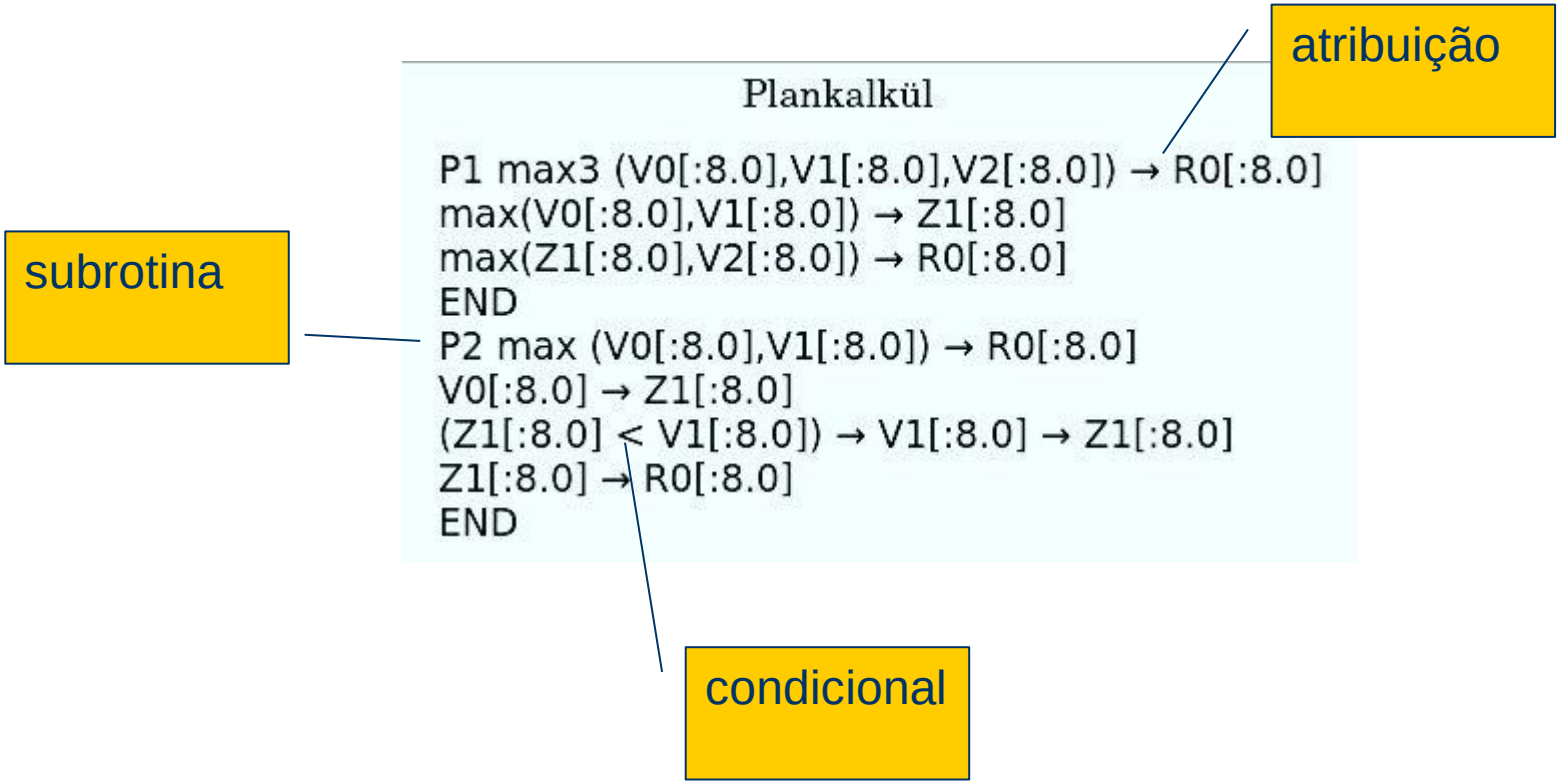
FROM *SCIENTIFIC MEMOIRS*, EDITED BY RICHARD TAYLOR, VOL. III. 1843

- http://celebratingada.com/
- http://factmyth.com/factoids/ada-lovelace-wrote-the-first-computer-program/

Primeira linguagem: Plankalkul (1946)

- Konrad Zuze, construtor do Z1, Z2, Z3...
- Concebeu uma linguagem em que pudesse ser empregado um nível de abstração mais alto
- Não chegou a ser implementada até 1975
- Considerada por Zuze como um exercício mental
- Primeira concepção de um compilador: leria comandos nessa linguagem e automaticamente perfuraria cartões com os comandos em linguagem de máquina

Primeira linguagem: Plankalkul (1946)



O problema da abstração

Programa em linguagem de máquina para adicionar dois números:

Location Hex	Instruction Code Binary	Instruction Code Hex	Instruction	Comments
100	0010 0001 0000 0100	2104	LDA 104	Load first operand into AC
101	0001 0001 0000 0101	1105	ADD 105	Add second operand to AC
102	0011 0001 0000 0110	3106	STA 106	Store sum in location 106
103	0111 0000 0000 0001	7001	HLT	Halt computer
104	0000 0000 0101 0011	0053	operand	83 decimal
105	1111 1111 1111 1110	FFFE	operand	-2 decimal
106	0000 0000 0000 0000	0000	operand	Store sum here

O problema da abstração

Programa Java™ para adicionar dois números:

```
int a, b, c;
a = 83;
b = -2;
c = a + b;
```

Programa em linguagem Assembly para adicionar dois números:

```
        ORG 100     /Origin of program is location 100
        LDA A       /Load operand from location A
        ADD B       /Add operation form location B
        STA C       /Store sum in location C
        HLT         /Halt computer
A,      DEC 83      /Decimal operand
B,      DEC –2      /Decimal operand
C,      DEC 0       /Sum stored in location C
        END
```

Compilação

(calendario.c) (gcc) (calendario.exe)

Compilador

- Traduz uma linguagem de alto nível na linguagem máquina da arquitetura destino
- Depois de compilado, o programa é específico da máquina onde corre

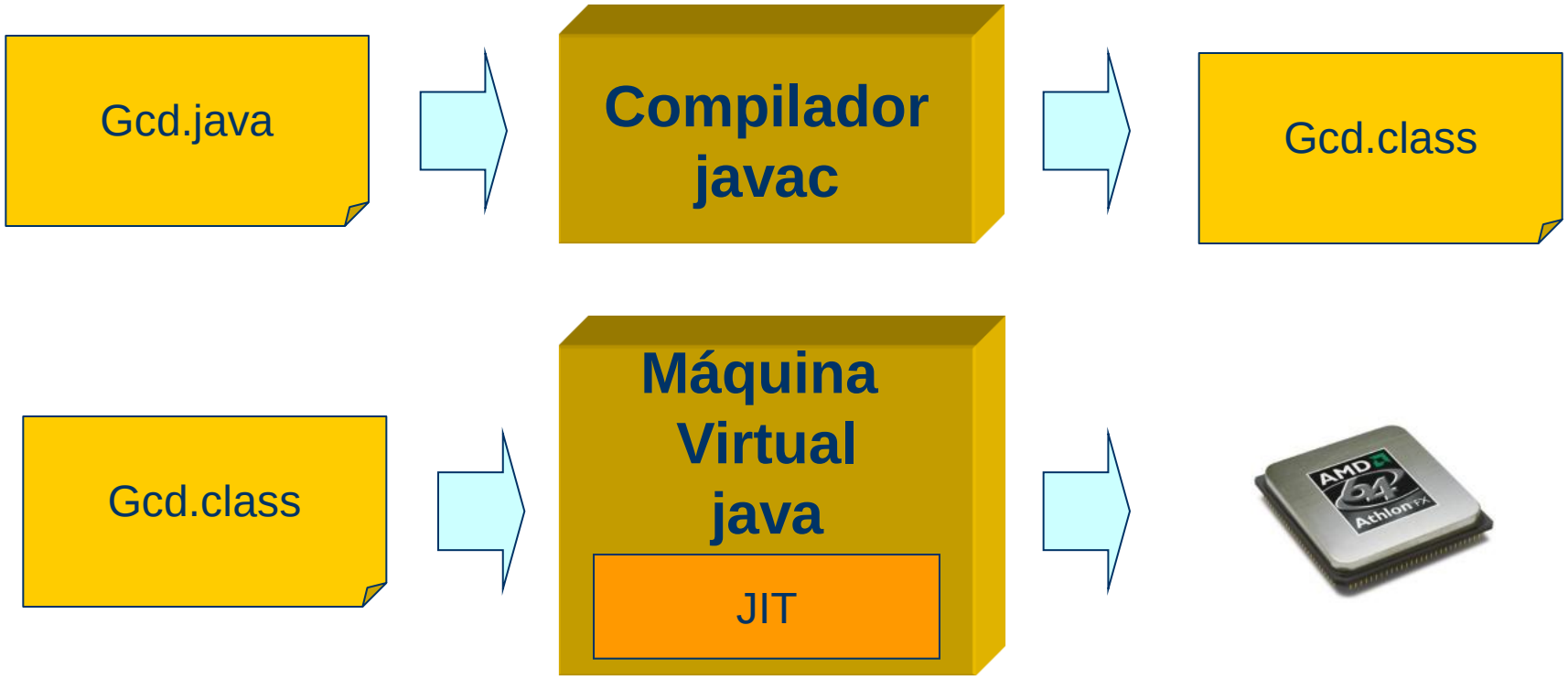
- As linguagens de programação de alto nível fornecem ao programador um conjunto de instruções que estão próximas da sua forma de pensar e do seu domínio de aplicação
  - Em 1995 estavam inventariadas cerca de 2300 linguagens (comp.lang.misc)

Primeiro compilador: A-0 (1952)

- Grace Hopper, programadora do Harvard Mark I
- Em 1949 contratada para desenvolver software para o UNIVAC
- A-0: programas que calculam seno, coseno, etc. gravados em fita. Programa desvia execução para posição na fita. Não é exatamente um compilador....mas deu grandes ideias para serem refinadas depois
- Evoluções: A1 até A-3 → pseudocódigo

Máquinas Virtuais

- Na máquina virtual existe um *just-in-time compiler* (JIT) que antes de executar o código o traduz em código máquina do processador alvo

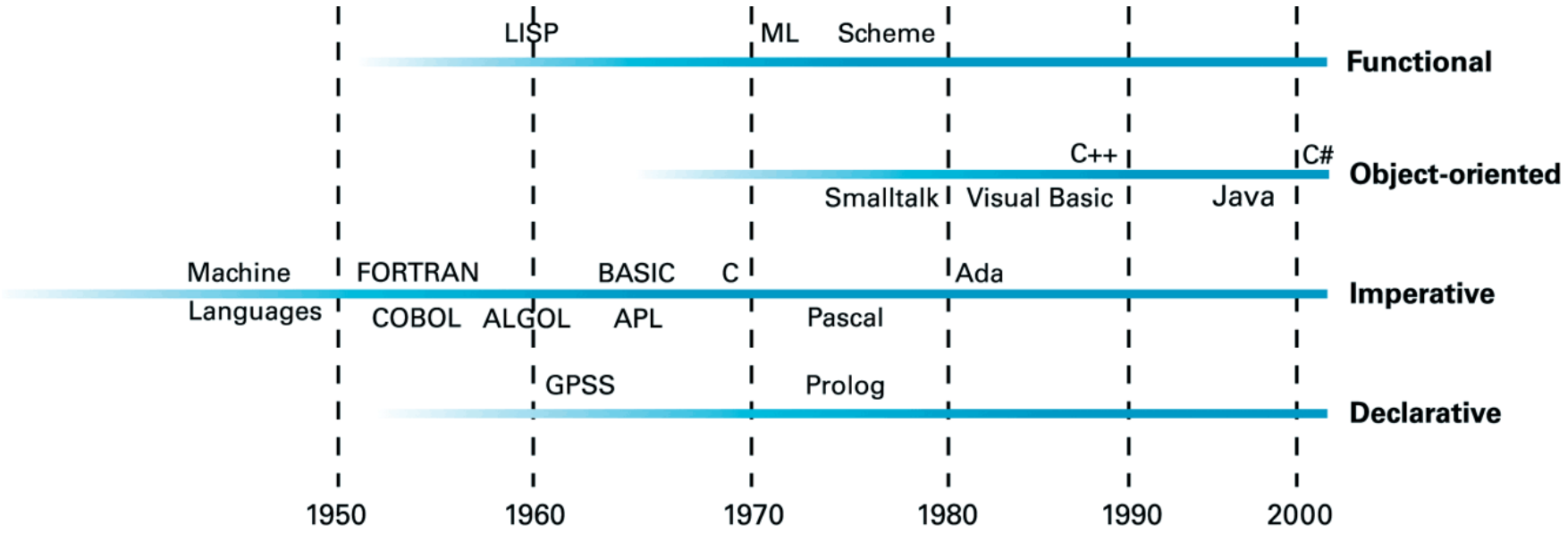


Paradigmas de Programação

- Atualmente existem quatro paradigmas de linguagens de programação em uso comum:
  - Imperativas (e.g. C, Pascal, Fortran)
  - Funcionais (e.g. LISP, Scheme)
  - Lógicas/Declarativas (e.g. Prolog)
  - Orientadas-a-Objetos (e.g. Java, C++, C#)

Hoje em dia a indústria é dominada pelos paradigmas Imperativo e Orientado-aos-Objetos

Evolução das Linguagens de Programação



Linguagens Imperativas

- Para programar um computador diz-se que…
  - PROGRAMA =
    ESTRUTURAS DADOS + ALGORITMOS

- No programa existem variáveis que representam os dados
- Existe um conjunto de instruções que sucessivamente, a cada instrução, altera o valor das variáveis, manipulando os dados
- Segue de forma bastante próxima o modelo básico de funcionamento do processador
- Exemplos: C, Pascal, Fortran

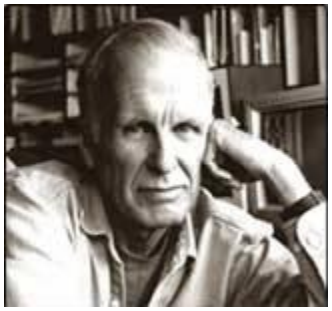
Estrutura Típica de uma Linguagem Imperativa

```
int fatorial;
int n;
int i;

void main()
{
  scanf("%d", &n);

  fatorial = 1;
  for (i=1; i<=n; i++)
    fatorial = fatorial*i;

  printf("%d", fatorial);
}
```

A primeira parte do programa consiste na declaração dos dados

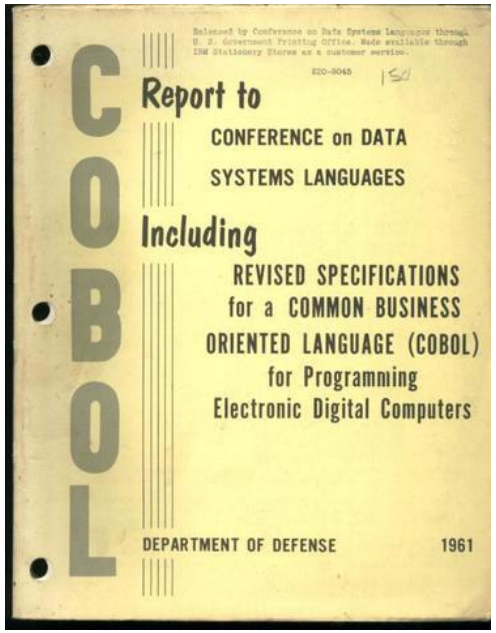
A segunda parte do programa consiste nas instruções que manipulam os dados

Personalidades… John Backus



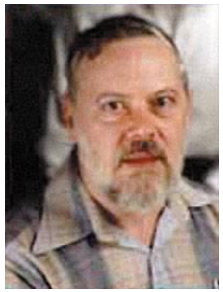
- John Backus, desenvolveu o FORTRAN *circa* 1957, na IBM
- Foi a primeira linguagem de alto nível digna desse nome
- FORmula TRANslating
- Era previsto levar 6 meses a fazer, levou mais de 2 anos – ninguém sabia as técnicas básicas de implementar um compilador, aprenderam aqui.
  - BNF: Backus-Naur Form -> usada para descrever Algol no ano seguinte

COBOL (1960)

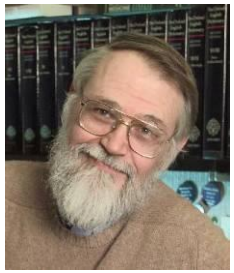


- Necessidade de linguagem para facilitar a escrita de programas para a indústria: comitê CODASYL
- Programas em COBOL tinham 3 grandes divisões: dados, procedimentos e ambiente
- Em dezembro de 1960 ocorreu um marco histórico: dois computadores de fabricantes diferentes rodaram o mesmo programa, modificando apenas a divisão de ambiente
- Portabilidade!!

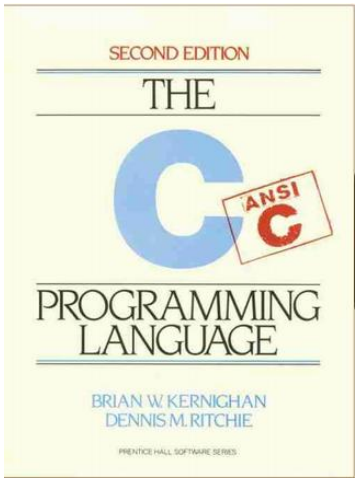
Dennis Ritchie (& Brian Kernighan)



Dennis Ritchie



Brian Kernighan



- Em 1967, M. Richards desenvolve a linguagem BCPL
- Em 1970, Ken Thompson implementa o núcleo sistema operacional UNIX em Assembly!
  - A primeira linguagem/compilador escrita para Unix foi a B, uma variante do BCPL
- Dennis Ritchie (Bell Labs) reconhece a necessidade de implementar o próprio sistema operacional usando uma linguagem de alto nível: inventa o C, uma evolução do B
- A linguagem C é altamente apropriada para programação de sistema
- Dennis Ritchie e Brian Kernighan escrevem a bíblia do C: “The C Programming Language”

Linguagens Funcionais

- Não existem atribuições de variáveis: tudo é feito invocando funções
- Tradicionalmente são utilizadas em cálculo simbólico / Inteligência Artificial
- Tipicamente tem suporte direto para trabalharem com Listas de Símbolos
- Exemplos: LISP, ML, Scheme

- Em termos de indústria não tiveram grande aceitação, embora alguns software a utilizem (e.g. AutoCAD, Emacs ☺)

Exemplo: Fatorial em LISP

```
(defun fatorial (x)
      (if (<= x 0)
            1
            (* x (fatorial (- x 1)))
      )
)
```

- Não tem definição de tipos/variáveis
- Não tem instruções de iteração
- Não tem instruções de atribuição
- (Quase) Tudo são definições de funções
- Uso forte de recursividade

- A primitiva básica é a lista!

```
(+ (* 2 4) (/ 4 3))
```

LISP

- Criado por John McCarthy em 1959
- A principal ideia era a manipulação de símbolos utilizando listas diretamente na linguagem
  - LISP = LIST PROCESSING
    (+ 5 (* 2 5))
  - A primeira tentativa chamava-se FLPL (Fortran List Processing Language)
- As funcionalidades que McCarthy queria eram:
  - Expressões condicionais (ifs)
  - Recursividade
  - Listas
  - Garbage Collection
- Escreveu um artigo onde definia o LISP e a sua função base *eval*
  - Uma aluno dele notou que era possível implementar o *eval* na prática, o que deu origem ao LISP!

Programação Lógica

- O programador não diz como é que se resolve um problema. Apenas diz:
  - Quais são os fatos
  - Quais são os teoremas que descrevem o sistema
  - O interpretador/compilador encarrega-se de encontrar a solução para as interrogações feitas ao programa
- Isto implica que na sua forma pura:
  - Não existem atribuições
  - Não existe controle de fluxo
- Linguagem mais conhecida: PROLOG
  - Também é fortemente baseada em listas
  - Utilização: Inteligência Artificial

PROLOG – Raciocinar sobre Famílias

- Fatos e teoremas (o que é dado ao sistema):

```
pai(carlos, antonio).
pai(antonio, jose).
pai(miguel, antonio).

avo(X,Y) :- pai(X,Z), pai(Z,Y).
irmao(X,Y) :- pai(X,Z), pai(Y,Z).
```

- Interrogações (o que perguntamos ao sistema):

```
?- pai(carlos, X).
X = antonio;
no

?- avo(carlos, X).
X = jose;
no

?- irmao(carlos, X).
X = miguel;
no
```

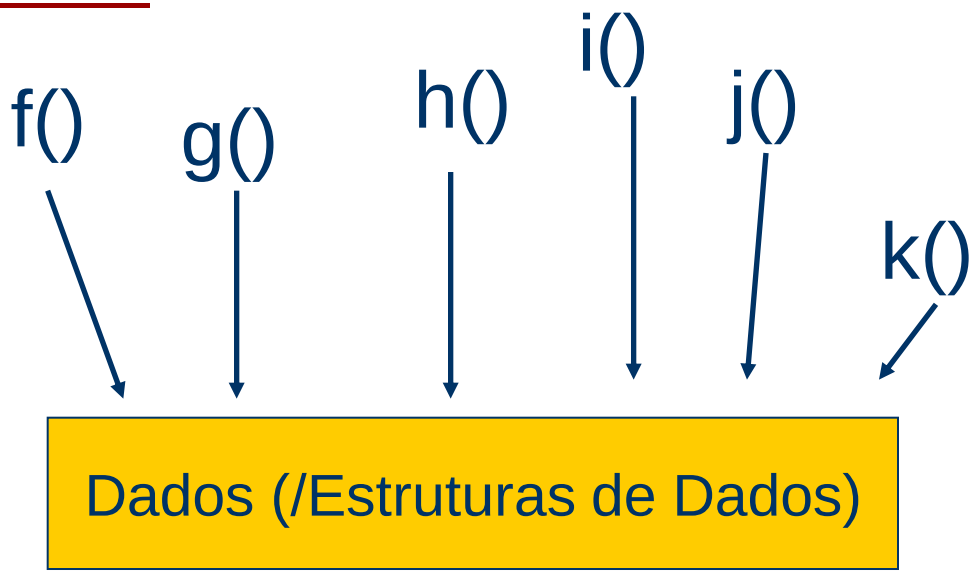
PROLOG – Calcular um Fatorial

- Fatos e Teoremas

```
fat(N,1) :-
    N =:= 1.

fat(N, Resultado) :-
    N > 1,
    K is N-1,
    fat(K, FatK),
    Resultado is N*FatK.
```

Programação Orientada a Objetos

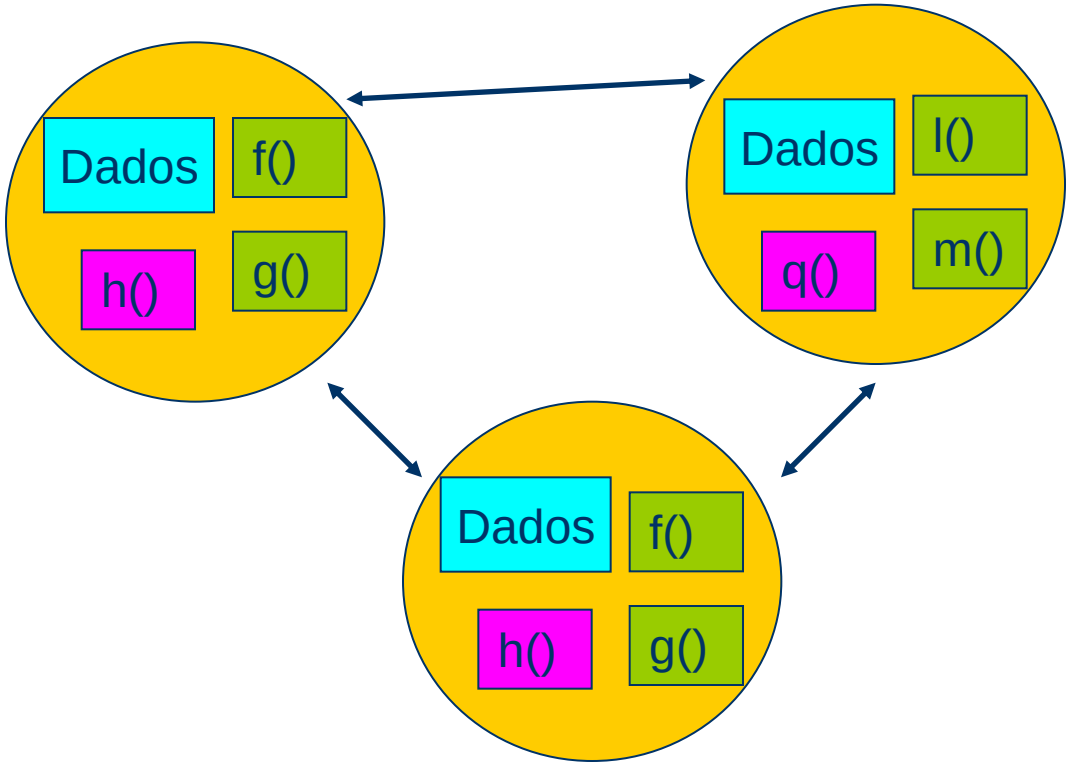
- Motivação: Os grandes problemas da programação imperativa, estruturada:
  - Grande Acoplamento!
  - Baixa Coesão!



Temos os dados, e o programa é constituído por milhares de funções que…
-- Ou manipulam diretamente esses dados
-- Ou trocam muitos valores por parâmetro

Programação Orientada a Objetos

- Em OOP (Object-Oriented Programming), as funções estão encapsuladas juntamente com os dados a que podem (e devem acessar)



Programação Orientada a Objetos

- A principal ideia dos objetos é que:
  - Apenas as funções relacionadas com os dados os acessam
  - Reduzir o acoplamento e aumentar a coesão, isto é, permitir a construção de software em projetos de larga escala, de forma consistente e fácil de gerenciar
  - Além disso, é muito mais natural pensar em termos de objetos e suas relações do que em termos de dados e algoritmos

- Programação estruturada:
  - PROGRAMA = DADOS + ALGORITMOS
- Programação orientada a Objetos
  - PROGRAMA = OBJETOS + RELAÇÕES

Classes e Objetos (Java)

```
class Pessoa {
  private String nome;
  private int idade;

  Pessoa(String nomePessoa, int idadePessoa) {
    nome = nomePessoa;
    idade = idadePessoa;
  }

  public void identifica() {
    System.out.println(nome + ": " + idade);
  }
}
```

```
Pessoa cliente1 = new Pessoa("Antonio", 32);
Pessoa cliente2 = new Pessoa("José", 23);

cliente1.identifica();
cliente2.identifica();
```

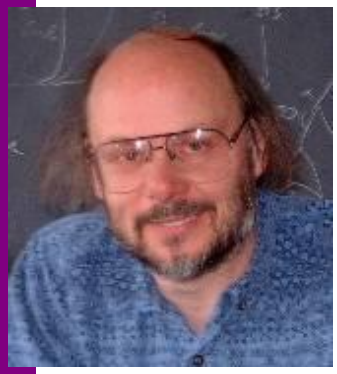
Um pouco de História… Alan Kay

- Um dos pais da programação orientada a objetos
  - SmallTalk, Laboratórios da XEROX em Palo Alto, 1972
  - Também inventou o conceito de computador pessoal, GUI e Portátil… (sim, para além da programação orientada aos objetos…)
  - A ideia de computador pessoal era RADICAL!

    "There is no reason anyone would want a computer in their home." (Ken Olsen, Digital Equipment Corp, **1977**)

- As ideias da programação orientada a objetos, de Kay, vêm da Biologia!!!

Um pouco de História… Bjarne Stroustrup



- Bjarne Stroustrup queria ter classes e objetos na linguagem C
- Criou um pré-processador que compilava a sua linguagem “C with Classes” para C
  - 1984, Bell Labs, C++
- Alguns dos problemas do C++ é que é muito grande, complicada de utilizar e muito fácil de cometer erros/gerar código de baixa qualidade

Um pouco de História… JAVA

- Em 1991 a Sun começa um projeto para construir uma linguagem para sistemas embarcados
  - Linguagem Oak, James Gosling, Sun Microsystems
- Em 1994 a Internet começava a mostrar sinais promissores.
- O projeto Oak é adaptado para a Internet → Nasce o Java em 1995
- Filosofia do JAVA:
  - Ser parecido com o C/C++
  - Eliminar radicalmente tudo o que há de mau (ou considerado mau) no C++
  - Adicionar ideias brilhantes de outros sistemas (Carregamento dinâmico de código, Máquina Virtual, Garbage Collection, Threads, …)