História das Linguagens de Programação

Alexsandro Santos Soares
prof.asoares@gmail.com

Universidade Federal de Uberlândia
Faculdade de Computação

História das Linguagens de Programação

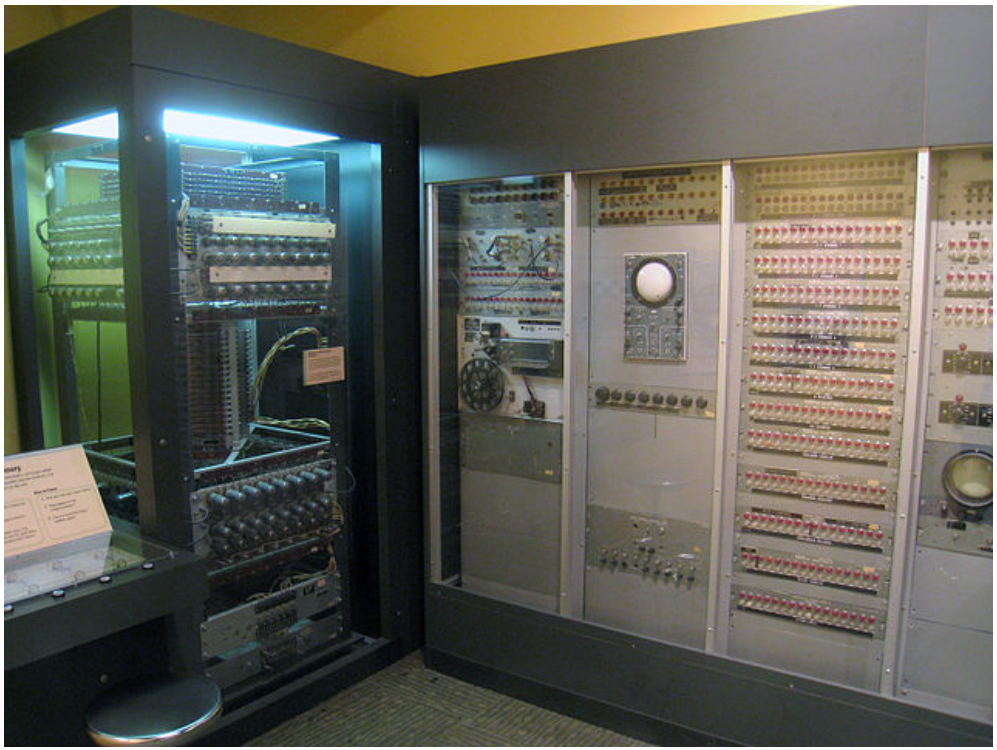
- Como vimos, os primeiros computadores modernos (1950 à 1957) eram programados usando códigos binários de máquina, um para cada tipo de computador.
- Nessas máquinas registradores tinham papéis diferentes:
  - R1 pode ser somado com os registradores R2 a R4, os registradores R3 a R6 podem descolar bits, etc.
- E os programadores deveriam saber tudo isso!

Assembler

- Uma das primeiras melhorias foi o montador (*assembler*).
- Ele era um programa que traduzia `add r1,10` no código de máquina apropriado.
- O programador fazia o mesmo trabalho complicado de antes: para cada tipo de computador ele tinha que conhecer sua arquitetura.
- Mas, com a tradução automatizada, a leitura e depuração do código ficava mais fácil.
- Dois projetos governamentais dos EUA nas décadas de 50 e 60 foram responsáveis pelo treinamento de programadores: o Whirlwind e o SAGE.

Whirlwind

- O Whirlwind da Marinha que iniciou em 1944 e cujo propósito era construir simuladores de vôo melhores e de propósito mais geral.



- Após saber do sucesso do ENIAC, a Marinha decidiuconstruir um computador eletrônico próprio.
- O hardware era um desafio, mas a *configuração* (a programação) foi mais difícil do que esperado.

SAGE

- O SAGE (Semi-Automatic Ground Environment) da Força Aérea, iniciado em 1955, tinha como objetivo interligar todo o sistema de radares dos EUA, visando evitar ataques aéreos da antiga União Soviética.
- Por volta de 1956 o hardware estava pronto mas a agência quase governamental que o supervisionava, a RAND, achou que não haviam programadores suficientes.

SAGE

- Eles iniciaram uma grande campanha para recrutamento e treinamento.
  - Os anúncios em revistas procuram jogadores de Xadrez e pessoa boas em resolver quebra-cabeças.
  - Depois eles concluíram que professores de música também se tornariam bons programadores.
- Em 1963 o SAGE ainda estava em execução mas, no mínimo 6000 programadores treinados pelo SAGE o tinham deixado para trabalhar na iniciativa privada.

Flow-Matic

- De 1952 a 1955 Grace Hopper trabalhava em uma forma mais fácil de configurar computadores e propôs uma *linguagem* geral de computador, a qual ela deu o nome de Flow-Matic.
- Os benefícios de tal abordagem eram:
  - Menos tempo de treinamento dos programadores.
  - Escrita mais rápida de código.
  - O mesmos programas poderiam ser executados em máquinas diferentes.

Exemplo de código Flow-Matic

- Abaixo está um trecho de código em Flow-Matic

```
INPUT INVENTORY FILE A;
PRICE FILE B;
OUTPUT PRICED INVENTORY FILE C.
COMPARE PRODUCT #A WITH PRODUCT #B.
IF GREATER, GO TO OPERATION 10;
IF EQUAL, GO TO OPERATION 5;
OTHERWISE GO TO OPERATION 2.
TRANSFER A TO D;
WRITE ITEM D;
JUMP TO OPERATION 8.
```

- A ideia era que uma vez escrito o código fonte em Flowmatic, o compilador o transformaria em um executável (um novo arquivo, repleto de código assembly). Este arquivo poderia, depois, ser executado.

Flow-Matic

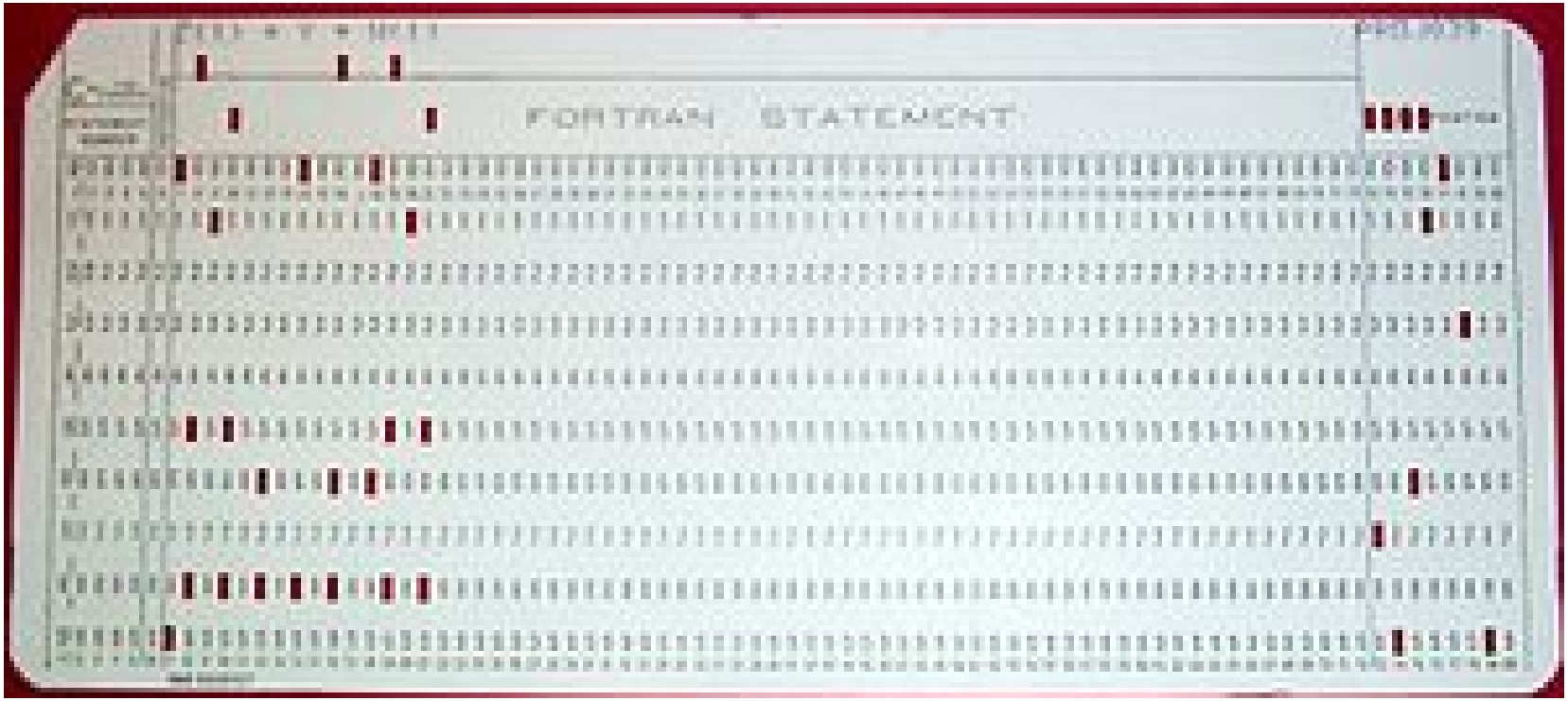
- Em 1958 Flow-matic estava sendo usado para escrever programas reais no UNIVAC.
  - Ela não é usada hoje em dia, mas sim sua versão melhorada, COBOL.

Fortran

- De 1954 a 1957 a IBM, John Backus e uma equipe de cerca de 10 pessoas estavam trabalhando no que viria a ser a primeira linguagem de programação prática, FORTRAN.

- O ponto essencial era que o compilador poderia verter linhas de FORTRAN em código assembly que era tão bom, ou ainda melhor, que o código assembly escritos por programadores *de verdade*.

Fortran



- Fortran permitia que mais pessoas escrevessem programas e em menor tempo.
- Ele também facilitava a adaptação de um programa antigo ou a execução do mesmo programa em diferentes computadores.
- Tudo que era necessário era que houvesse um compilador para a nova máquina.

Exemplo de código em Fortran II

```
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
      READ INPUT TAPE 5, 501, IA, IB, IC
  501 FORMAT (3I5)
      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 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
```

Cobol

- Fortran era muito boa para cálculos matemáticos, mas não tão boa com nomes e endereços ou grandes arquivos.
- Por isso, muitas empresas, como a IBM, estavam trabalhando em um linguagem comercial.
- Fortran foi pensada como uma linguagem científica e incluir estas demandas comerciais poderiam torná-la mais lenta.
  - Assim ela não seria bom em coisa alguma.
- Um consórcio formado pelo pessoas do governo dos EUA e de várias empresas escreveram COBOL, que foi baseada em FLOW-MATIC.

Cobol

- Uma das pessoas que especificaram o Cobol foi Jean Sammet, que trabalhou para Grace Hopper.

- COBOL tornou-se uma das linguagens mais populares para bancos, companias de seguro, etc.
  - Embora muitos cientistas da computação veem Cobol como a pior linguagem de programação já inventada.
- No proxímo slide está um trecho de um programa Cobol.que, entre as linhas 110 a 120, realiza um laço que lê e adiciona todos os salários de empregados.

Exemplo de programa em Cobol

- Entre as linhas 110 a 120, ele realiza um laço que lê e adiciona todos os salários dos funcionários.

```
1  00050 TOTAL-SALARY PIC 9999.
2  00100 READ EMPLOYEE-RECORD.
3  00103 MOVE 0 TO TOTAL-SALARY.
4  00105
5  00110 PERFORM ACCUMULATE-SALARY
6  00120   UNTIL NO-MORE-EMPLOYEES EQUAL-TO "YES".
7  00130
8  00200 ACCUMULATE-SALARY.
9  00210   ADD EMP-SALARY TO TOTAL-SALARY GIVING TOTAL-SALARY.
10 00220   READ EMPLOYEE-RECORD
11 00230     AT END
12 00240       MOVE "YES" TO NO-MORE-EMPLOYEES.
```

- Cobol foi feita para ler cartões perfurados e por isso as linhas numeradas eram importantes.
- As colunas de 1 a 6 eram reservadas para números de linhas.
- Note também que a função ACCUMULATE-SALARY não recebia nenhuma entrada.

Cobol

- Muito embora Cobol seja uma tecnologia dos anos 1960, ela ainda está em uso hoje.
- Existem muitos programas antigos usados por bancos e companias de seguro que são escritos em COBOL.
  - É mais fácil e muito mais seguro atualizar um programa do que escrever um novo do zero.
- Em parte, o bug do milênio, foi causado pelos muitos programas escritos em Cobol que usavam dois dígitos para datas.
  - Ninguém poderia imaginar que programas escritos em 1970 poderiam ainda estar em uso no ano 2000.

Linguagens de alto nível

- Linguagens como o Cobol e o Fortran vieram a ser chamadas como linguagens de alto nível, contrastando com o assembly que era chamado de linguagem de baixo nível.
- A ideia era que no nível mais baixo estaria a máquina real e quanto mais alto na hierarquia de abstrações se está, menos se pode ver como ela realmente funciona.
- Um grande número de linguagens de programação foram propostas ainda nesse período.
  - Muitas delas não pegaram.
  - Não eram práticas, eram apenas experimentos em projetos de compiladores.

Outras Linguagens

- LISP era para processamento de listas e popular em inteligência artificial.
- SnoBol esta para processamento de strings.
- Algol, escrito na Europa, tinha grandes ideias mas ninguém conseguia escrever um bom compilador para ela.
- O PL/1 da IBM tinha tantas características que ninguém poderia aprendê-las na totalidade, ou escrever um compilador que tratasse todas elas.
- JOVIAL foi uma das muitas primerias centenas de linguagens para uso militar.
- Smalltalk tinha todas as variáveis como *objetos* que enviavam *mensagens* uns para os outros.
  - Ela inspirou linguagens como C++ e Java.

Exemplos de programas em Algol e PL/1

Algol

```
procedure Absmax(a)
    Size:(n, m) Result:(y) Subscripts:(i, k);
    value n, m; array a; integer n, m, i, k;
    real y;
comment The absolute greatest element of the
     matrix a, of size n by m is transferred
    to y, and the subscripts of this element
    to i and k;
begin integer p, q;
    y := 0; i := k := 1;
    for p:=1 step 1 until n do
    for q:=1 step 1 until m do
        if abs(a[p, q]) > y then
            begin y := abs(a[p, q]);
            i := p; k := q
            end
end Absmax
```

PL/I

```
PROCEDURE(ARRAY,N); /* BUBBLE SORT*/
DECLARE (I,J) FIXED BIN(15);
DECLARE S BIT(1);          /* SWITCH */
DECLARE Y FIXED BIN(15); /* TEMPO */
DO I = N-1 BY -1 TO 1;
  S = '1'B;
  DO J = 1 TO I;
    IF X(J)>X(J+1) THEN DO;
      S = '0'B;
      Y = X(J);
      X(J) = X(J+1);
      X(J+1) = Y;
      END;
    END;
  IF S THEN RETURN;
  END;
RETURN;
END SRT;
```

BASIC

- BASIC foi escrita em Dartmouth em 1964 como uma linguagem mais fácil para que engenheiros, químicos, etc, pudessem usar computadores para resolverem suas equações.
- Ao final da década de 70, quando os fabricantes de microcomputadores estavam procurando por uma linguagem simples e pequena, eles adotaram BASIC.
  - Em 1975, Ed Roberts usou uma versão mais enxuta da BASIC para o seu kit de computador pessoal Altair. Bill Gates e Paul Allen, então estudantes em Harvard, foram os primeiros a responder ao anúncio.
  - Em 1977, Steve Wozniak escreveu uma versão da BASIC para o seu Apple II.
  - Era só ligar o computador e começar a digitar um programa em Basic. Depois podia-se salvá-lo em um disquete de 5 1/4 polegadas com um comando.
  - Livros e revistas com jogos escritos em Basic, se tornaram populares nesta época.

Exemplo em Basic

- Abaixo esta uma programa escrito em uma versão de Basic:

```
1  10 for ST = 1 to 80
2  20 print "*"
3  30 endfor
4  40 printline
5  45 MY=100
6  50 printline "   Welcome to BlackJack. Enter bet:"
7  55 read BT
8  70 RL=rand(6)+rand(6)
9  80 if RL=7 or RL=11
10 85 printline "You win"
11 90 let MY = MY + BT
12 100 goto 1000
13 105 endif
14 110 print "Point is " RL " Press space to roll again"
15 ....
16 1000 print "Play again?"
17 1010 read AG
18 1020 if AG = "Y"
19 1030 goto 50
```

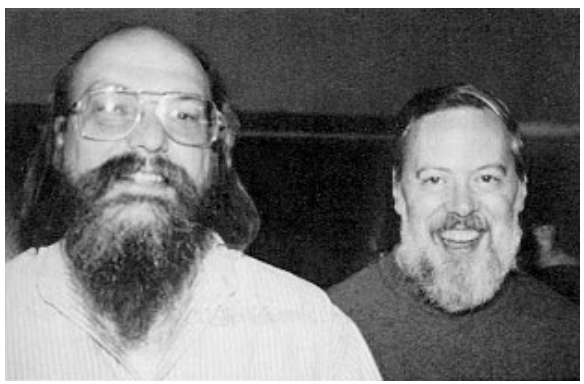
Pascal

- Pascal foi escrita em 1970 como uma linguagem moderna (no seu tempo) e enxuta, sem casos especiais ou coisas interessantes que poucos realmente compreendiam e que causavam erros.
- Ela se tornou popular como uma linguagem de ensino

```
1  procedure stupidsort (var a: array of integer);
2   var i: integer;
3   begin
4     i := 0;
5     repeat
6       inc (i);
7       if (a[i+1] < a[i]) then begin
8         exchange (a[i], a[i+1]);
9         i:=0;
10      end;
11    until i = length(A) - 2;
12  end;
```

C

- O sistema operacional UNIX foi projetado pela AT&T para os computadores que gerenciavam telefones.
  - Ele foi escrito em linguagem assembly para o computador PDP por Ken Thompson.
- Em 1972 ele precisou ser reescrito.
  - Assembly levaria muito tempo e FORTRAN não possuia a habilidade de mover bits de memória.
- Assim, Dennis Ritchie criou C para reescrever o UNIX.



- C foi beneficiada pelos 15 anos de experiência desde que FORTRAN fora introduzido e era perfeitamente boa para aplicações matemáticas.

Exemplo de programa em C

- O seguinte código realiza seis operações matemáticas e envia os seus respectivos resultados para a saída padrão (em geral, a tela).

```c
# include <math.h> // necessária para pow() e sqrt()
# include <stdio.h> // necessária para printf()

int main()
{
   int a = 2, b = 3;

   printf("%d + %d = %d\n", a, b, a + b);
   printf("%d - %d = %d\n", a, b, a - b);
   printf("%d x %d = %d\n", a, b, a * b);
   printf("%d / %d = %0.1f\n", a, b, (float) a / b);
   printf("%d elevado a %d = %0.1f\n", a, b, pow(a, b));
   printf("raiz quadrada de %d = %0.1f\n", a, sqrt(a));

   return 0;
}
```

C++

- Em 1984 Bjarne Stroustrup, também da AT&T, projetou uma versão orientada a objetos de C, que ele chamou de C++.

- No início, C++ era um pré-processador que compilava tudo para C.
- Hoje, ela possui um compilador próprio.
  - Na verdade, vários de muitos fabricantes.

Exemplo de programa em C++

```
1  #include <iostream>
2
3  class Passaro {                    // classe base
4  public:
5     virtual void MostraNome() {
6        std::cout << "um passaro";
7     }
8     virtual ~Passaro() {}
9  };
10
11 class Cisne: public Passaro { // Cisne éum pássaro
12 public:
13    void MostraNome() {
14       std::cout << "um cisne"; // sobrecarrega a função virtual
15    }
16 };
17
18 int main() {
19    Passaro* passaro = new Cisne;
20    passaro->MostraNome(); // imprime "um cisne", e não "um pássaro"
21    delete passaro; }
```

Java

- Por volta de 1994 a internet tournou-se popular e as linguagens distribuídas também.
- A ideia original era que torradeiras, geladeiras, etc poderiam precisar de uma linguagem de programação.
  - Elas todas precisariam se comunicar em rede usando alguma linguagem em comum, independentemente do chip por elas usados.
- Uma linguagem como FORTRAN não era equipada com material para *redes* e, além disso, ela era compilada para uma linguagem assembly específica de um processador.
- Java foi escrita pela SUN em 1992 para resolver este problema.
  - Se parece com C/C++.
  - Tentou eliminar o que se eram considerados erros de projeto em C++.
  - Adiciona ideias de outros sistemas tais como: carregamento dinâmico de código, máquina virtual, gerenciamento automático de memória, etc.)

Exemplo de programa em Java

```
1  public class Animal{
2     public void correr(){ System.out.println("correndo..."); }
3  }
4  public Cachorro extends Animal{
5     public void latir(){ System.out.println("Latindo..."); }
6  }
7  public Gato extends Animal{
8     public void miar(){ System.out.println("Miando..."); }
9  }
10 class Programa {
11     public static void main(String[] args) {
12       Cachorro c = new Cachorro();
13       c.correr();
14       c.latir();
15
16       Gato g = new Gato();
17       g.correr();
18       g.miar();
19
20       Animal a = new Animal();
21       a.correr(); }
```

Linguagens Imperativas vs. Linguagens funcionais

- Nos dias atuais, COBOL e Fortran são bastante similares.
- Elas são exemplos do que chamamos de linguagens *imperativas*. Elas possuem:
  - Variáveis que representam os dados: `a=b+c*7`.
  - Estruturas condicionais `if-else`.
  - Estruturas de iteração como `while(x<10)`.
- Esse conjunto de instruções é executado uma linha por vez até que se alcançe o fim.
  - Essas instruções podem alterar o valor das variáveis.
- Quase todas as linguagens vistas aqui são imperativas.
  - C++ e Java são chamadas de linguagens orientadas a objetos mas, elas são linguagens imperativas que possuem orientação a objetos.
- As linguagens imperativas seguem o modelo básico de funcionamento do processador.

Linguagens Imperativas vs. Linguagens funcionais

- Um outra família de linguagens de programação são as chamadas linguagens funcionais, pois elas usam funções para quase tudo.
- Haskell, LISP, Scheme e ML enquadram-se nesta categoria.
- Embora elas estejam por aí faz tempo e funcionem muito bem, elas não são tão populares.
  - Muito usadas em Inteligência Artifical.
  - Vários programas as utilizam tais como AutoCAD, Emacs, etc.

Exemplo em LISP

- Abaixo está um trecho de programa LISP que conta o números de a's em uma lista:

```
1 (defun countas (alist)
2  (if(= alist ()) 0
3     (if (= (car alist) 'a)
4         (+ 1 (countas(cdr alist)))
5         (countas(cdr alist))
6  )))
7
8 >(countas '(h a y a a b a))
9 4
```

- Ele diz que a resposta é 0 se `alist` é vazia. Em caso contrário, verifique se a primeira coisa é um `a`. Se for, a resposta é 1 mais o número de `a`'s no resto da lista. Se o primeiro elemento *não* for `a`, a resposta é o número de de `a`'s no resto da lista.

Exemplo em C

- Abaixo está um programa em C, uma linguagem imperativa, que faz a mesma coisa:

```
1 numa=0;
2 index=0;
3 while(index < listsize)
4  { if (alist[index] == 'a')
5        numa=numa+1;
6    index=index+1;
7  }
```

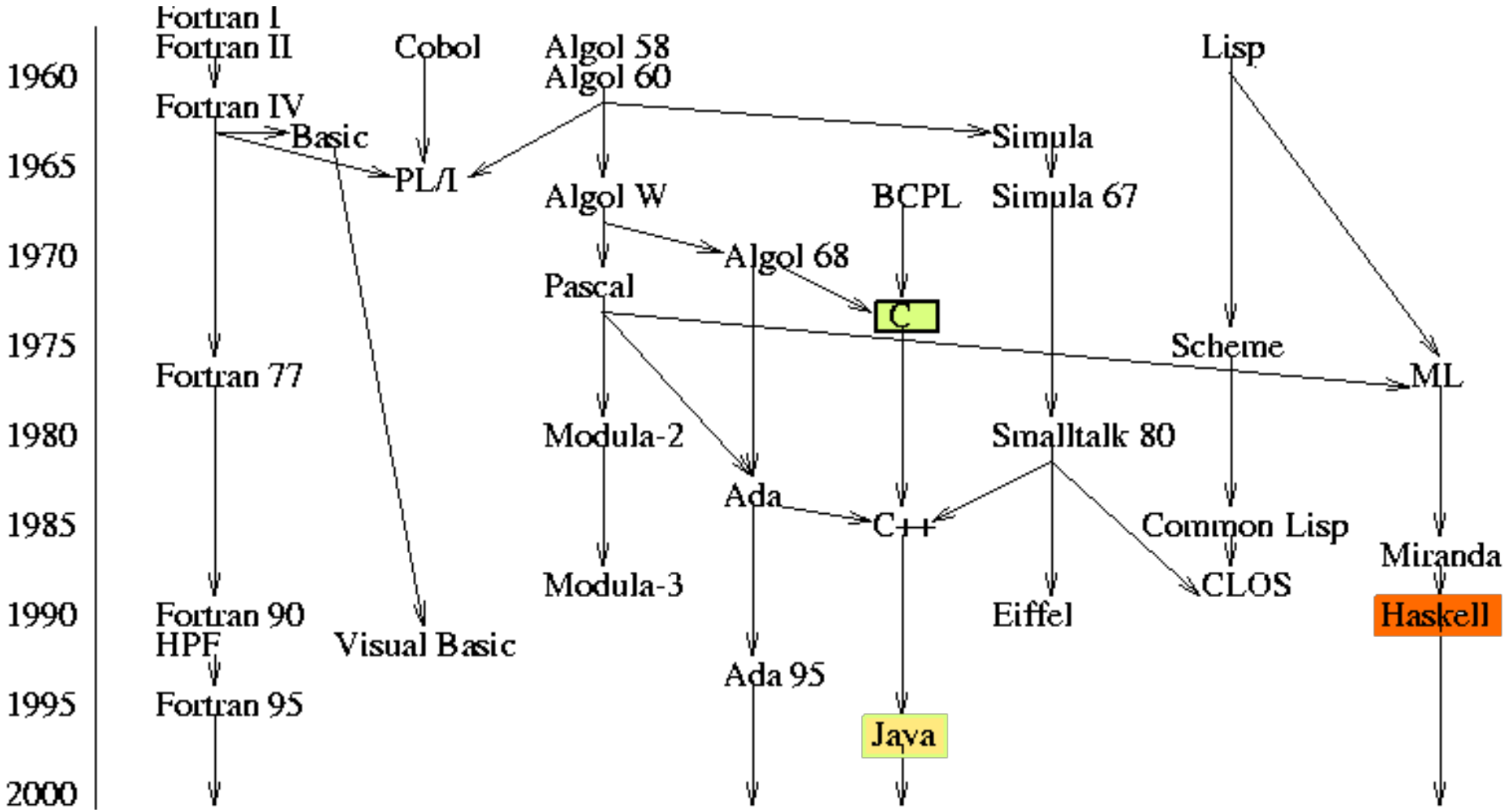
Linguagens interpretadas

- De forma prática, um programa poderia ser escrito, salvo em uma fita magnética ou de papel, compilada e depois salvo em uma fita magnética diferente.
- Em um computador pessoal, isso não erá prático.
  - Um compilador era um programa muito grande e os computadores domésticos não tinha muito espaço de armazenamento.
  - Seria melhor apenas digitar o programa e mandar executá-lo.
- A solução era usar um intérprete que é um programa que lê linhas de código e as executa imediatamente.
- Alguém poderia escrever um intérprete em assembly e colocá-lo em chip.
- Os programas interpretados eram mais lentos que aqueles compilados.

Linguagens interpretadas

- Para os primeiros PCs, os intérpretes funcionavam bem.
  - Eles eram pequenos.
  - Os programas ocupam menos espaço que os compilados.
  - Os usuários domésticos não se importavam em esperar algumas centenas de segundos.
- Jogos em PC's eram todos escritos em assembly, devido à velocidade de execução.
- Scripts para o console (*Shell scripts*) são interpretados.

Árvore genealógica simplificada das linguagens de programação



Para saber mais

- Owen Reynolds. Overview of programming languages. Disponível em `http://web.cs.iastate.edu/~cs104/notes/histlanguage.html`

Fontes

- História das linguagens de programação. In: *Wikipédia, a enciclopédia livre*.