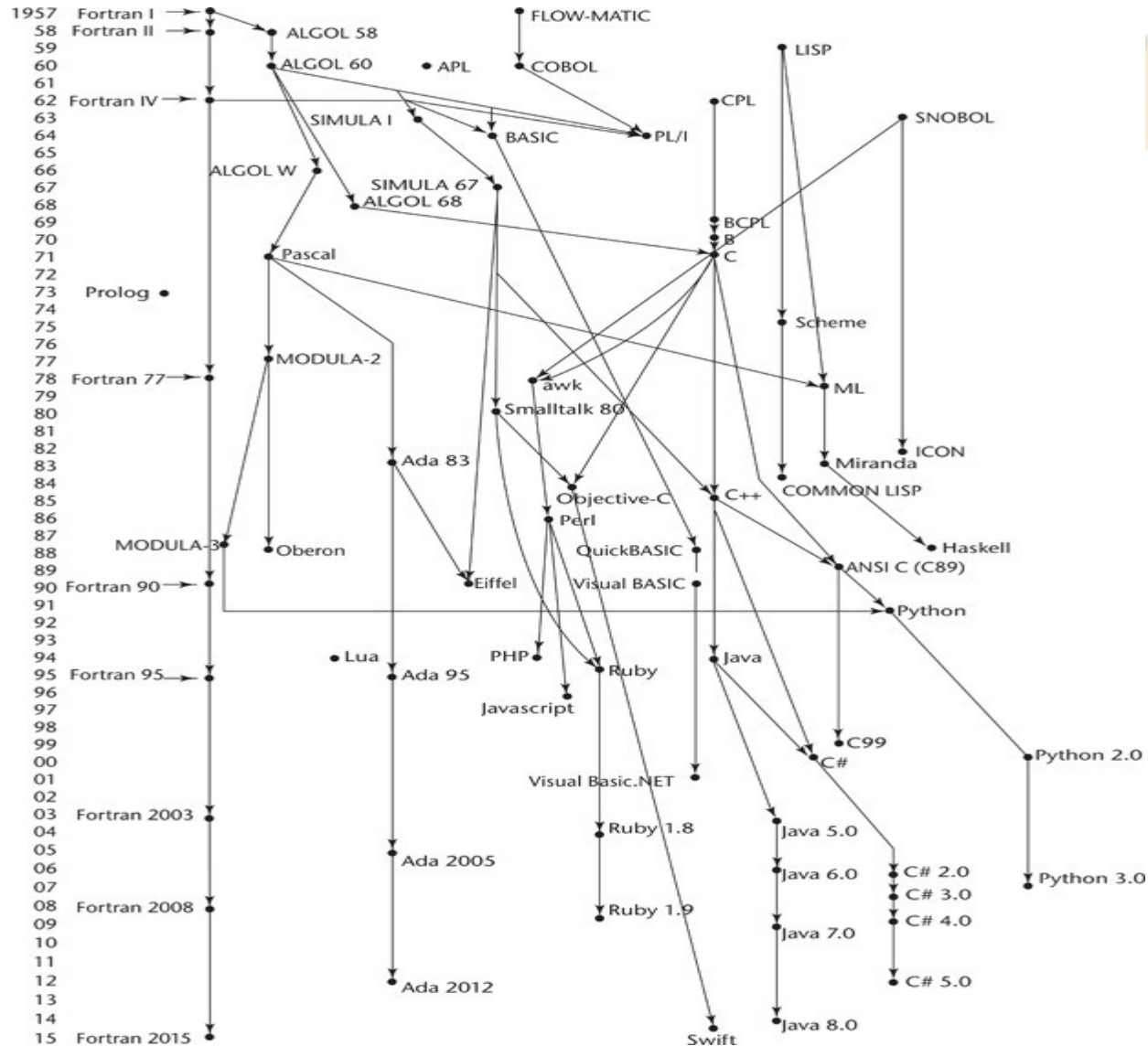


Paradigmas de Linguagens de Programação

Evolução das Principais Linguagens de Programação

Cristiano Lehrer, M.Sc.

Genealogia das Linguagens de Programação



Evolução das Principais Linguagens de Programação • Paradigmas de Linguagens de Programação

Copyright © 2009/2024 • ybadoo.com.br



Linguagem Plankalkül de Zuse

- Criada por Konrad Zuse, em 1945:
 - Publicada somente em 1972.
- Nunca foi implementada.
- Estruturas avançadas:
 - Estruturas de dados: ponto flutuante, *arrays* e registros.
 - Controle: *loops*, sem **goto**, instruções de seleção.
 - Notação: atribuir o valor da expressão $A(4) + 1$ para $A(5)$
 - | $A + 1 \rightarrow A$
 - V | 4 5 (subscritos)
 - S | 1.n 1.n (tipos de dados)

Programação de Hardware Mínima (1/3)

- Qual o problema em usar código de máquina:
 - Legibilidade pobre.
 - Difícil de modificar:
 - Problema de endereçamento.
 - Codificação de expressões penosa.
 - Deficiências da máquina:
 - Sem indexação ou ponto flutuante.

Programação de Hardware Mínima (2/3)

- Short Code – 1949:
 - Proposta por John Mauchly para o BINAC.
 - Descrição original não publicada:
 - Versão UNIVAC I, em 1952.
 - Expressões eram codificadas da esquerda para a direita.
 - Código em pares de bytes.
 - Implementação como interpretador.
 - Algumas operações:

01 → -	02 →)	03 → =
04 → /	07 → +	09 → (

Programação de Hardware Mínima (3/3)

- Speedcoding – 1954:
 - Proposta por John Backus, para o IBM 701.
 - Interpretador implementado pseudocódigo para 4 operações aritméticas e funções matemáticas em ponto flutuante.
 - Ramificação condicional e incondicional.
 - Conversões de I/O.
 - Incremento automático de endereço de registradores.
 - Lento (4.2 ms para executar uma soma).
 - Somente 700 palavras para programa do usuário.

FORTRAN (1/8)

- Proposta por John Backus, em 1954, para o IBM 704.
- IBM *Mathematical Formula Traslating System*.
- Ambiente de desenvolvimento:
 - Computadores eram pequenos e não confiáveis.
 - Aplicações eram científicas.
 - Sem metodologias ou ferramentas de programação.
 - Eficiência (da máquina) era o mais importante.
- Impacto do ambiente no projeto:
 - Sem necessidade de armazenamento dinâmico.
 - Necessitava de um bom manuseio de *array* e *loops* de contagem.
 - Sem manuseio de strings, aritmética decimal ou entrada/saída poderosa.

FORTRAN (2/8)

- **FORTRAN I – 1957:**
 - Primeira versão implementada do **FORTRAN**.
 - I/O formatado.
 - Nomes poderiam ter até seis caracteres.
 - Subprogramas definidos pelo usuário.
 - Comando de seleção de três modos (**if** aritmético).
 - *Loop* de contagem pós-testado (**do**).
 - Todas as estruturas de controle baseados no 704.
 - Compilador lançado depois de 18 anos de esforço.
 - Programas maiores de 400 linhas raramente compilavam.
 - Código era muito rápido.
 - Rapidamente tornou-se popular.

FORTRAN (3/8)

- **FORTRAN II** – 1958:
 - Corrigiu vários bugs do **FORTRAN I**.
 - Compilação independente de sub-rotinas.
 - 50% do código escrito para o 704 era **FORTRAN**.

FORTRAN (4/8)

- FORTRAN IV – 1960 – 1962:
 - FORTRAN III nunca foi largamente distribuído.
 - Declaração de tipos explícita.
 - Comando de seleção lógica.
 - Nomes de subprogramas podiam ser parâmetros.
 - Padrão ANSI em 1966.

FORTRAN (5/8)

- FORTRAN 77 – 1978:
 - Manuseio de **character string**.
 - Comando de controle de *loop* lógico.
 - Comando **if-then-else**.

FORTRAN (6/8)

- FORTRAN 90 – 1990:
 - Funções *built-in* para operações com *arrays*.
 - *Arrays* dinâmicos.
 - Ponteiros.
 - Tipo registro.
 - Recursão.
 - Novas estruturas de controle → **case**.
 - Checagem de tipo de parâmetro.
 - Módulos.

FORTRAN (7/8)

- Avaliação:
 - Primeira linguagem de alto nível.
 - Mudou dramaticamente e permanentemente o modo como os computadores são usados.

FORTRAN (8/8)

```
C EXEMPLO DE PROGRAMA FORTRAN 90
C ENTRADA: UM NÚMERO INTEIRO (INTEGER), LIST_LEN, EM QUE LIST_LEN É MENOR QUE 100,
          SEGUIDO DE VALORES LIST_LEN_INTEGER
C SAÍDA:  O NÚMERO DE VALORES DE ENTRADA QUE SÃO MAIORES DO QUE A MÉDIA DE TODOS OS
          VALORES DE ENTRADA
          INTEGER INTLIST(99)
          INTEGER LIST_LEN, CONTADOR, SOMA, MEDIA, RESULTADO
          RESULTADO = 0
          SOMA = 0
          READ *, LIST_LEN
          IF((LIST_LEN .GT. 0) .AND.
             (LIST_LEN. LT. 100)) THEN
C LEIA DADOS DE ENTRADA EM UM ARRAY E COMPUTE SUA SOMA
          DO 10 CONTADOR = 1, LIST_LEN
             READ *, INTLIST(CONTADOR)
             SOMA = SOMA + INTLIST(CONTADOR)
10          CONTINUE
C COMPUTE A MEDIA
          MEDIA = SOMA / LIST_LEN
C CONTE OS VALORES QUE SÃO MAIORES DO QUE A MÉDIA
          DO 20 CONTADOR = 1, LIST_LEN
             IF (INTLIST(CONTADOR) .GT. MEDIA) THEN
                 RESULTADO = RESULTADO + 1
             END IF
20          CONTINUE
C IMPRIMA O RESULTADO
          PRINT *, 'NÚMERO DE VALORES > QUE A MÉDIA:', RESULTADO
          ELSE
             PRINT *, 'ERRO: VALOR DO COMPRIMENTO DA LISTA NÃO É VÁLIDO'
          END IF
          STOP
          END
```

LISP (1/5)

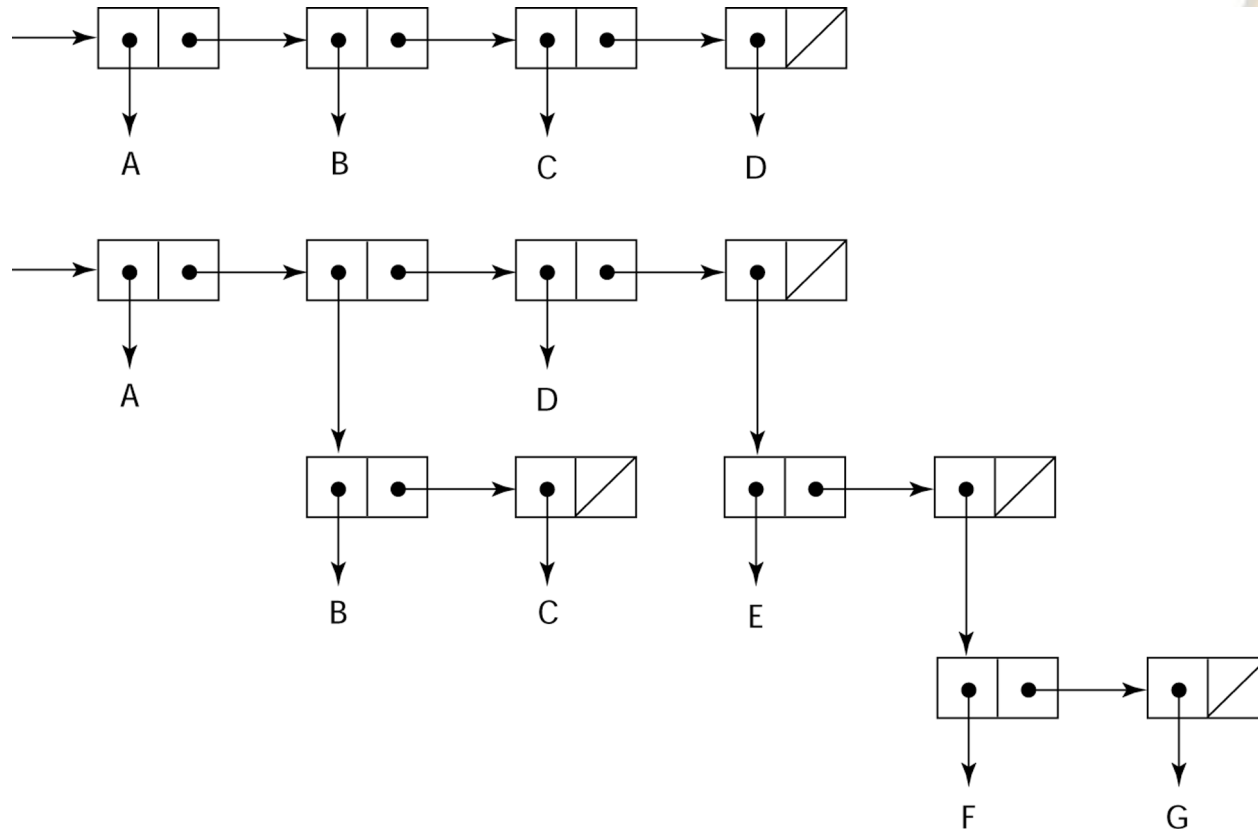
- Proposta por John McCarthy no MIT, em 1959.
- *List Processing Language*.
- Interesse em Inteligência Artificial (IA) surgiu nos anos 50:
 - Processamento de listas:
 - Allen Newell, J. C. Shaw e Hebert Simon.
- Tentativa:
 - **FLPL**
 - *Fortran List Processing Language*.

LISP (2/5)

- Pesquisa em IA necessitava de uma linguagem que:
 - Processasse dados em listas (ao invés de *arrays*).
 - Computação simbólica (ao invés de numérica).
- Características:
 - Somente dois tipos de dados:
 - Átomos.
 - Listas.
 - Sintaxe baseada no *lambda calculus*.
 - Pioneira da programação funcional:
 - Não necessita variáveis ou atribuição.
 - Controle via recursão e expressões condicionais.

LISP (3/5)

- (A B C D)
- (A (B C) D (E (F G)))



LISP (4/5)

- Avaliação:
 - Paradigma alternativo ao modelo imperativo.
 - Ainda uma das linguagens dominantes em IA.
 - **COMMON LISP** e **SCHEME** são dialetos contemporâneos do **LISP**.
 - **ML**, **MIRANDA** e **HASKELL** são linguagens mais modernas que herdam aspectos do **LISP**.

LISP (5/5)

```
; Função de exemplo LISP
; O código seguinte define uma função predicada LISP
; que pega duas listas como argumentos e retorna (True)
; se as duas listas forem iguais, e NIL (falso)
; caso contrário.
(DEFUN listas_iguais (lis1, lis2)
  (COND
    ((ATOM lis1)(EQ lis1 lis2))
    ((ATOM lis2) NIL)
    ((EQUAL (CAR lis1) (CAR lis2))
     (EQUAL (CDR lis1)(CDR lis2)))
    (T NIL)
  )
)
```

ALGOL (1/7)

- ALGOL 58 – 1958:
 - *Algorithmic Language*;
 - Ambiente de desenvolvimento:
 - FORTRAN era recém chegado para IBM 70x.
 - Muitas outras linguagens estavam sendo desenvolvidas, todas para máquinas específicas.
 - Não havia linguagem portátil, todas dependiam da máquina.
 - Nenhuma linguagem universal para representar (comunicar) algoritmos.
 - ACM e GAMM encontram-se por quatro dias para o projeto:
 - Association for Computing Machinery (ACM).
 - Sociedade para Matemática Aplicada e Mecânica da Alemanha (GAMM).

ALGOL (2/7)

- ALGOL 58 – 1958 (continuação):
 - Objetivos da linguagem:
 - Proximidade da notação matemática.
 - Boa para descrever algoritmos.
 - Deve ser traduzível para código de máquina.
 - Conceito de tipo foi formalizado.
 - Nomes podiam ter qualquer tamanho.
 - *Arrays* podiam ter um número qualquer de índices.
 - Parâmetros eram separados por modo (entrada/saída).
 - Índices eram colocados entre colchetes.
 - Comandos de composição de blocos (**begin...end**).
 - Ponto e vírgula como separador de comando.
 - Operador de atribuição era **:=**
 - **if** possuía uma cláusula **else-if**

ALGOL (3/7)

- ALGOL 58 – 1958 (continuação):
 - Observações:
 - Não era a intenção implementá-la, mas variações dela foram:
 - MAD e JOVIAL.
 - Apesar da animação inicial da IBM, o apoio foi retirado em meados de 1959.

ALGOL (4/7)

- ALGOL 60 – 1960:
 - Modificou o **ALGOL 58** durante um encontro de seis dias em Paris.
 - Novas características:
 - Estrutura de blocos (escopo local).
 - Dois modos de passagem de parâmetros:
 - Subprogramas recursivos.
 - Arrays em pilhas dinâmicas.
 - Ainda nenhum I/O nem manuseio de strings.

ALGOL (5/7)

- ALGOL 60 – 1960 (continuação):
 - Sucessos:
 - Foi o padrão para publicação de algoritmos por mais de 20 anos.
 - Todas as linguagens imperativas são baseadas nela.
 - Primeira linguagem independente de máquina.
 - Primeira linguagem cuja sintaxe foi definida formalmente (BNF):
 - Forma de Backaus-Naur.

ALGOL (6/7)

- ALGOL 60 – 1960 (continuação):
 - Falhas:
 - Nunca foi amplamente utilizada, especialmente nos EUA.
 - Motivos:
 - Nenhuma I/O e o conjunto de caracteres faziam com que os programas não fossem muito portáteis.
 - Muito flexível, portanto difícil de implementar.
 - Entrincheiramento do **FORTRAN**.
 - Sintaxe de descrição formal:
 - Estranho e complicado para a época.
 - Falta de apoio da IBM.

ALGOL (7/7)

comment Programa de exemplo ALGOL 60

Entrada: um número inteiro, listlen, em que listlen é menor do que 100, seguido de valores listlen-integer

Saída: o número de valores de entrada que são maiores do que a média de todos os valores de entrada;

begin

```
integer array intlist [1:99];  
integer listlen, contador, soma, media, resultado;  
soma := 0;  
resultado := 0;  
readint (listlen);  
if (listlen > 0) L (listlen < 100) then  
begin  
  comment leia a entrada em um array e compute a média;  
  for contador := 1 step 1 until listlen do  
  begin  
    readint(intlist[contador]);  
    soma := soma + intlist[contador];  
  end;  
  comment compute a média;  
  media := soma / listlen;  
  comment conte os valores de entrada que são > do que a média;  
  for contador := 1 step 1 until listlen do  
    if intlist[contador] > media  
      then resultado := resultado + 1;  
  comment imprima o resultado;  
  printstring("O número de valores > média é:");  
  printint(resultado);  
end  
else  
  printstring("Erro - tamanho da lista de entrada não é legal");  
end
```

end

COBOL (1/6)

- Common Business Oriented Language – 1960.
- Ambiente de desenvolvimento:
 - UNIVAC começava a usar o **FLOW-MATIC**.
 - USAF começava a usar o **AIMACO**.
 - IBM desenvolvia o **COMTRAN** (commercial translator).
- “Programas matemáticos deveriam ser escritos usando a notação matemática, programas para processamento de dados deveriam ser escritos usando assertivas em inglês”.
 - Grace Hopper, 1953.

COBOL (2/6)

- Definição do **COBOL** foi baseado no **FLOW-MATIC**:
 - Nomes de até 12 caracteres, com hifenação.
 - Nomes em inglês para operadores aritméticos.
 - Dados e código eram completamente separados.
 - Verbos eram a primeira palavra em cada comando.
- Primeiro encontro de projeto:
 - Maio de 1959.

COBOL (3/6)

- Objetivos do projeto:
 - Deve parecer com inglês simples.
 - Deve ser fácil de usar, mesmo que signifique menos recursos (poder).
 - Deve estender a base de usuários de computadores.
 - Não deve ser influenciado por problemas de implementação (compiladores).
- Comitê do projeto eram todos de fabricantes de computadores e agências do Departamento de Defesa.

COBOL (4/6)

- Problemas de projeto:
 - Expressões aritméticas.
 - Indexação.
 - Brigas entre fabricantes.
- Contribuições:
 - Primeiro recurso macro em uma linguagem de alto nível.
 - Estruturas de dados hierárquicas (registros).
 - Comandos de seleção aninhados.
 - Nomes longos (até 30 caracteres), com hifens.
 - Data division.

COBOL (5/6)

- Avaliação:
 - Primeira linguagem solicitada pelo Departamento de Defesa Americano (DoD):
 - Teria falhado sem o DoD.
 - Linguagem de aplicação em negócios ainda amplamente utilizada.
 - Apesar de ter sido uma das linguagens mais utilizadas, teve pouco efeito (aparte do [PL/I](#)) no projeto de outras linguagens.

COBOL (6/6)

```
IDENTIFICATION DIVISION.  
PROGRAM-ID. Multiplier.
```

```
DATA DIVISION.
```

```
WORKING-STORAGE SECTION.
```

```
01 Num1 PIC 9 VALUE ZEROS.  
01 Num2 PIC 9 VALUE ZEROS.  
01 Result PIC 99 VALUE ZEROS.
```

```
PROCEDURE DIVISION.
```

```
    DISPLAY "Enter first number (1 digit) : " WITH NO ADVANCING.  
    ACCEPT Num1.  
    DISPLAY "Enter second number (1 digit) : " WITH NO ADVANCING.  
    ACCEPT Num2.  
    MULTIPLY Num1 BY Num2 GIVING Result.  
    DISPLAY "Result is = ", Result.  
    STOP RUN.
```


BASIC (1/4)

- Beginner's All-purpose Symbolic Instruction Code.
- Projetado no Dartmouth College por John Kemeny e Thomas Kurtz, em 1964.
- Objetivos do projeto:
 - Fácil de aprender e usar:
 - Por outros que não fossem estudantes de programação.
 - Deve ser agradável e amigável.
 - Rápido de usar em trabalhos domésticos.
 - Acesso livre e particular.
 - Tempo do usuário mais importante que tempo de máquina.

BASIC (2/4)

- Versão inicial era compilada, sem entrada pelo terminal.
- Simples:
 - 14 tipos de comandos.
 - 1 tipo de dado.
- Dialetos populares atualmente:
 - QuickBASIC.
 - Visual BASIC.

BASIC (3/4)

- Avaliação:
 - Primeiro método de acesso via terminal remoto.
 - Algumas aplicações de porte foram desenvolvidas:
 - Sistema operacional do PDP-11, da DIGITEL.
 - Estrutura de programação pobre.
 - Não destinadas a projetos sérios de qualquer porte.
 - Facilidade para aprendizado e implementação.
 - Popularização com o computador pessoal.

BASIC (4/4)

REM Exemplo de programa em QuickBASIC

INPUT A, B

FOR I=A TO B

 IF MOD(I,2)=0 THEN PRINT I

NEXT

PL/I (1/6)

- Projetado pela IBM, em 1965.
- Situação da computação no início dos anos 60 (IBM):
 - Computação científica:
 - Computadores IBM 1620 e 7090.
 - **FORTRAN**.
 - *SHARE user group*.
 - Computação comercial:
 - Computadores IBM 1401 e 7080.
 - **COBOL**.
 - *GUIDE user group*.

PL/I (2/6)

- Em 1963:
 - Usuários da computação científica começavam a precisar de I/O mais elaboradas, que **COBOL** possuía.
 - Usuários da computação comercial começavam a necessitar de ponto flutuante e arrays, que **FORTRAN** possuía.
- Parecia que muitas lojas começavam a precisar de dois tipos de computadores, linguagens e pessoal de suporte:
 - Muito caro!
- Solução óbvia:
 - Construir um novo computador para ambas aplicações:
 - IBM System/360.
 - Projetar uma nova linguagem para fazer os dois tipos de aplicações:
 - Substituir **FORTRAN**, **COBOL**, **LISP** e os **ASSEMBLY**;

PL/I (3/6)

- Incluía o que era considerado o melhor:
 - ALGOL 60:
 - Recursão.
 - Estrutura de blocos.
 - FORTRAN IV:
 - Compilação em separado.
 - Comunicação por dados globais.
 - COBOL 60:
 - Estrutura de dados.
 - I/O.
 - Facilidades para geração de relatórios.

PL/I (4/6)

- Avaliação:
 - Contribuições:
 - Primeira linguagem a permitir a criação de tarefas com execução concorrente.
 - Primeiro tratamento de exceção (32 tipos).
 - Recursão selecionáveis.
 - Tipo de dado ponteiro.
 - Referência a seções de arrays.

PL/I (5/6)

- Avaliação (continuação):
 - Comentários:
 - Muitas características novas tinham problemas de projeto.
 - Muito grande e complexo.
 - Foi de fato usado para aplicações científicas e comerciais.
 - No mínimo, um sucesso parcial.

PL/I (6/6)

```
HELLO:PROCEDURE OPTIONS (MAIN);  
  
    /* A PROGRAM TO OUTPUT HELLO WORLD */  
    FLAG = 0;  
  
LOOP: DO WHILE (FLAG = 0);  
        PUT SKIP LIST('HELLO WORLD!');  
    END;  
  
END HELLO;
```

APL e SNOBOL (1/4)

- Caracterizadas por tipagem dinâmica e alocação de memória dinâmica.
- Não foram baseadas em outras linguagens.
- Não tiveram grande influência em linguagens posteriores.

APL e SNOBOL (2/4)

- APL – A Programming Language – 1962:
 - Projetada como uma linguagem para descrição de hardware:
 - Na IBM por Ken Iverson.
 - Altamente expressiva:
 - Muitos operadores, tanto para escalares quando arrays de várias dimensões.
 - Programas são muito difíceis de ler:
 - Geração dos números primos de 1 até R:
 - $(\sim R \in R \circ . \times R) / R \leftarrow 1 \downarrow \uparrow R$

APL e SNOBOL (3/4)

- SNOBOL – String Oriented Symbolic Language, 1964:
 - Projetada como uma linguagem de manipulação de strings.
 - Operadores poderosos para *pattern matching* de strings.
 - Usada para escrever editores.

APL e SNOBOL (4/4)

```
* WORDSIZE.SNO
* Program to read a file and display the number of words of
* various word lengths. To make the program more interesting,
* we shall only consider word lengths between 3 and 9. This allows
* us to demonstrate the use of an array with subscripts offset from
* 1, as well as array failure.
&TRIM = 1
WORDPAT = BREAK(&LCASE &UCASE) SPAN(&LCASE &UCASE "'-") . WORD
COUNT = ARRAY('3:9',0)
READ LINE = INPUT :F(DONE)
NEXTW LINE WORDPAT = :F(READ)
COUNT<SIZE(WORD)> = COUNT<SIZE(WORD)>+ 1 :(NEXTW)
DONE OUTPUT= "WORD LENGTH NUMBER OF OCCURRENCES"
I = 2
PRINT I = I + 1
OUTPUT = LPAD(I,5) LPAD(COUNT<I>,20) :S(PRINT)
END
```

SIMULA 67 (1/2)

- Baseada no **SIMULA I**, criada por Kristen Nygaard e Ole-John Dahal, entre 1962 e 1964 na Noruega.
- Projetada inicialmente para simulações de sistemas.
- Baseada no **ALGOL 60** e **SIMULA I**.
- Contribuições:
 - Co-rotinas:
 - Um tipo de subprograma que pode ser reiniciado a partir do ponto onde previamente havia parado.
 - Implementada em uma estrutura chamada classe:
 - Classes são a base para abstração de dados.
 - Classes são estruturas que incluem dados locais e funcionalidade.

SIMULA 67 (2/2)

```
Begin
  Class Glyph;
    Virtual: Procedure print Is Procedure print;;
  Begin
  End;

  Glyph Class Char (c);
    Character c;
  Begin
    Procedure print;
      OutChar(c);
  End;

  Glyph Class Line (elements);
    Ref (Glyph) Array elements;
  Begin
    Procedure print;
    Begin
      Integer i;
      For i:= 1 Step 1 Until UpperBound (elements, 1) Do
        elements (i).print;
      OutImage;
    End;
  End;

  Ref (Glyph) rg;
  Ref (Glyph) Array rgs (1 : 4);

  ! Main program;
  rgs (1):- New Char ('A');
  rgs (2):- New Char ('b');
  rgs (3):- New Char ('b');
  rgs (4):- New Char ('a');
  rg:- New Line (rgs);
  rg.print;
End;
```


ALGOL 68 (1/2)

- A partir do desenvolvimento do [ALGOL 60](#), mas não é um superconjunto daquela linguagem.
- Projeto é baseado no conceito de ortogonalidade.
- Fonte de várias idéias adotadas por outras linguagens.
- Contribuições:
 - Estruturas de dados definidos pelo usuário.
 - Referência de tipos.
 - Arrays dinâmicos (chamados FLAX arrays).
- Comentários:
 - Usado menos ainda que o [ALGOL 60](#).
 - Teve forte influência nas linguagens subseqüentes, especialmente [PASCAL](#), [C](#) e [ADA](#).

ALGOL 68 (2/2)

```
int sum sq:=0;
for i
while
    sum sq≤1000
do
    sum sq+:=i**2
od
```

PASCAL (1/2)

- Criada por Niklaus Wirth, em 1971.
- Baseada no [ALGOL-W](#), proposta alternativa apresentada pelos que consideraram muito complexa, tanto a linguagem quanto a metalinguagem, do [ALGOL 68](#).
- Simplicidade e tamanho reduzido eram objetivos do projeto.
- Projetada para o ensino de programação estruturada.
- Ainda é uma das mais usadas linguagens para o ensino de programação estruturada nas universidades.

PASCAL (2/2)

```
program name;
uses
  crt; (* biblioteca do pascal *)
var
  n1,n2,q1,q2:string; (*variáveis criadas pelo usuário do tipo string*)
begin
  {Limpa a tela}
  clrscr;
  writeln('Digite o primeiro nome:');
  readln(n1);
  writeln('Digite o segundo nome:');
  readln(n2);
  writeln('Digite uma qualidade para o primeiro nome:');
  readln(q1);
  writeln('Digite uma qualidade para o segundo nome:');
  readln(q2);
  writeln(n1, ' ', q1);
  writeln(n2, ' ', q2);
  {Aguarda até uma tecla ser pressionada}
  readkey;
end.
```

C (1/2)

- Projetada para a programação de sistemas:
 - Dennis Richie, em 1972.
- Ancestrais incluem **CPL**, **BCPL**, **B** e **ALGOL 68**:
 - **CPL** foi desenvolvida em Cambridge em 1963.
 - **BCPL** com o objetivo de programação de sistemas.
 - **B** primeira linguagem de alto nível do UNIX.
 - **BCPL** nem **B** eram tipadas.
- Operadores poderosos, checagem de tipos pobres.
- Inicialmente propagada através do UNIX.

C (2/2)

```
#include <stdio.h>

struct pessoa
{
    unsigned short int idade;
    char nome[51]; /* vetor de 51 chars para o nome */
    unsigned long int rg;
}; /* estrutura declarada */

int main(void)
{
    /* declaração de uma variável tipo struct pessoa */
    struct pessoa exemplo = {16, "Fulano", 123456789};

    printf("Idade: %hu\n", exemplo.idade);
    printf("Nome: %s\n", exemplo.nome);
    printf("RG: %lu\n", exemplo.rg);

    return 0;
}
```

PROLOG (1/2)

- Desenvolvido na Universidade de Aix-Marseille, por Comerauer e Roussel, com ajuda de Kowalski da Universidade de Edinburgh, em 1972:
 - Programmation en Logique.
- Baseada na lógica formal.
- Não procedural.
- Comparativamente poderia ser definida como:
 - Um sistema de banco de dados inteligente que usa processos de inferência para inferir a verdade de expressões fornecidas.

PROLOG (2/2)

```
hanoi(N) :- move(N, left, centre, right).
move(0, _, _, _) :- !.
move(N, A, B, C) :- M is N-1,
                    move(M, A, C, B),
                    inform(A, B),
                    move(M, C, B, A).
inform(X, Y) :- write('move a disc from the '),
                write(X),
                write(' pole to the '),
                write(Y),
                write(' pole'),
                nl.
```


ADA (1/4)

- Enorme esforço de projeto, envolvendo centenas de pessoas, muito dinheiro e cerca de oito anos:
 - Projeto iniciado no meio dos anos 70.
 - Projeto concluído em 1983.
- Financiado pelo Departamento de Defesa dos Estados Unidos – DoD.
- Homenagem a Ada Lovelace, considerada a primeira mulher programadora da história da computação.

ADA (2/4)

- Contribuições:
 - Packages:
 - Suporte para abstrações de dados.
 - Manuseio de exceção:
 - Elaborado.
 - Unidades de programas genéricas.
 - Concorrência:
 - Através do modelo de tarefas.

ADA (3/4)

- Comentários:
 - Projeto competitivo.
 - Incluiu tudo que se conhecia sobre engenharia de software e projeto de linguagens.
 - Primeiros compiladores eram muito difíceis:
 - O primeiro compilador surgiu cinco anos depois que o projeto da linguagem tinha sido completado.
- **ADA 95** – iniciado em 1988:
 - Suporte para POO através de derivação de tipos.
 - Melhores mecanismos de controle para dados compartilhados:
 - Novas características de concorrência.
 - Bibliotecas mais flexíveis.

ADA (4/4)

```
With text_IO; Use text_IO;
With Ada.Integer_Text_IO; Use Ada.Integer_Text_IO;
Procedure usando_if is
    x, y : integer;
begin
    New_Line(3);
    Put("Digite o primeiro valor: ");
    Get(x);
    Put("Digite o segundo valor: ");
    Get(y);
    New_Line;

    if x>y then
        Put_Line("O Primeiro valor digitado e' maior!");
    elsif x<y then
        Put_Line("O Segundo valor digitado e' maior!");
    else
        Put_line("Os dois valores são iguais.");
    end if;
    New_Line(3);
end usando_if;
```

SMALLTALK (1/2)

- Desenvolvida pela Xerox PARC, em 1972 a 1980:
 - Inicialmente por Alen Kay, depois por Adele Goldberg.
- Primeira implementação completa de uma linguagem orientada por objetos:
 - Abstração de dados.
 - Herança.
 - Polimorfismo.
- Pioneira no tipo de interface gráfica que hoje todos utilizam.

SMALLTALK (2/2)

"Exemplo de Programa Smalltalk"

"A seguinte é uma definição de classe, cujas instanciações podem desenhar polígonos equiláteros de qualquer número de lados"

```
class name Polígono
superclass Object
instance variable names   nossaCaneta
    numLados
    comprimentoLado
" Métodos de classe "
" Crie uma instância "
new
    ^ super new getPen*
" Pegue uma caneta para desenhar polígonos "
getPen
    NossaCaneta <- Pen new defaultNib: 2
" Método de instâncias "
" Desenhe um polígono "
draw
    numLados vezesRept: [nossaCaneta go: comprimentoLado;
                        turn: 360 // numLados]
" Defina o tamanho dos lados "
comprimento: len
    comprimentoLado <- len
" Defina o número de lados "
lados: num
    numLados <- num
```

C++ (1/2)

- Desenvolvida nos laboratórios Bell, por Stroustrup, em 1985.
- Baseada no C e SIMULA 67.
- Facilidades para programação orientada por objetos, tiradas do SIMULA 67, foram adicionadas ao C.
- Possui tratamento de exceção.
- Linguagem grande e complexa, em parte por suportar programação procedural e orientada por objetos.
- Popularizou-se rapidamente, junto com a POO.
- Padrão ANSI aprovado em novembro de 1997.

C++ (2/2)

```
#include <iostream>

class Passaro // classe base
{
public:
    virtual void MostraNome()
    {
        std::cout << "um pássaro";
    }
    virtual ~Passaro() {}
};

class Cisne: public Passaro // Cisne é um pássaro
{
public:
    void MostraNome()
    {
        std::cout << "um cisne"; // sobrecarrega a função virtual
    }
};

int main()
{
    Passaro* passaro = new Cisne;

    passaro->MostraNome(); // produz na saída "um cisne", e não "um pássaro"

    delete passaro;
}
```


Eiffel (1/2)

- Criada por Bertrand Meyer, em 1992.
- Uma linguagem relacionada que suporta POO.
- Não foi derivada diretamente de qualquer outra linguagem.
- Menor e mais simples que C++:
 - Contudo ainda possui a maioria dos recursos do C++.

EIFFEL (2/2)

```
class
  HELLO_WORLD
create
  make
feature
  make
    do
      print("Hello World!%N")
    end
end
```

JAVA (1/2)

- Desenvolvida por uma equipe de programadores, chefiada por James Gosling, na empresa Sun Microsystems, em 1995.
- Baseada no C++:
 - Significativamente simplificada.
 - Suporta apenas POO.
 - Tem referência, mas não ponteiros.
 - Inclui suporte a *applets* e uma forma de concorrência.

JAVA (2/2)

```
import javax.swing.JOptionPane;
public class Soma {
    public static void main(String[] args) {

        //declaração das variáveis
        String numeroA, numeroB;
        int numero1, numero2, soma;

        //pede dois números inteiros
        numeroA = JOptionPane.showInputDialog("Entre com o primeiro número inteiro");
        numeroB = JOptionPane.showInputDialog("Entre com o segundo número inteiro");

        //converte os números de string para inteiro
        numero1 = Integer.parseInt(numeroA);
        numero2 = Integer.parseInt(numeroB);

        //efetua a soma dos números
        soma = numero1 + numero2;

        //mostra o resultado da soma para o usuário
        JOptionPane.showMessageDialog(null, "A soma dos números é: " +
        soma, "Resultado", JOptionPane.PLAIN_MESSAGE);
    }
}
```