



Linguagens Formais e Programação

Aula 1 – Linguagens de Programação

Prof. Flávio Ceci, Dr.

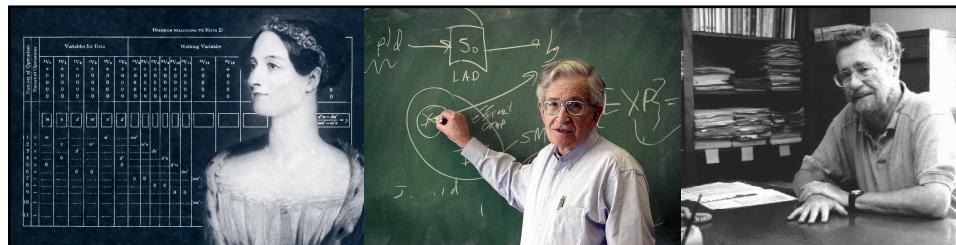
flavio.cec@unisul.br

1

Motivação

- Capacidade aumentada para expressar ideias;
 - Conhecendo a natureza das linguagens de programação, pode-se escolher a melhor para a resolução de um problema apresentado.
- Habilidade aumentada para aprender novas linguagens;
- Melhor entendimento da importância da implementação;
- Melhor uso de linguagens já conhecidas.

2



Evolução das Linguagens de Programação

Fonte: Braga, R. – Aula 8



3

A primeira programadora

Ada Lovelace (1815 ~ 1852):

- Ela desenvolveu os algoritmos que permitiriam à máquina analítica de Charles Babbage computar os valores de funções matemáticas;
- A Máquina Analítica (1837), foi um projeto de um computador **mecânico** moderno de uso geral, realizado pelo professor de matemática britânico Charles Babbage.



4

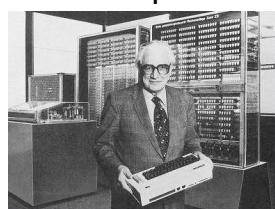
A primeira programadora

Number of Operation.	Nature of Operation.	Variables used upon.	Variables receiving results.	Indication of change in the value on any Variable.	Data.		Working Variables.								Result Variables.					
					1	2	n	2n	2n	V_{11}	V_{12}	V_{13}	V_{14}	V_{15}	V_{16}	V_{17}	V_{18}	V_{19}	V_{20}	V_{21}
1	\times	$V_{12} \times V_{13}$	V_{14}, V_{15}, V_{16}	$\{V_{12}=V_{13}\}$	$= 2n$					1	2n	2n	0	0	0	0	0	B_1
2	$-$	$V_{12} - V_{13}$	V_{14}, V_{15}	$\{V_{12} > V_{13}\}$	$= 2n+1$					1	2n+1	0	0	0	0	0	0	B_2
3	$+$	$V_{12} + V_{13}$	V_{14}, V_{15}	$\{V_{12} < V_{13}\}$	$= 2n-1$					1	2n-1	0	0	0	0	0	0	B_3
4	\times	$V_{12} \times V_{14}$	V_{15}, V_{16}	$\{V_{12}=V_{14}\}$	$= 2n-1$					1	2n-1	0	0	0	0	0	0	B_4
5	$+$	$V_{12} + V_{14}$	V_{15}, V_{16}	$\{V_{12} < V_{14}\}$	$= 2$					1	2	0	0	0	0	0	0	B_5
6	$-$	$V_{12} - V_{14}$	V_{15}, V_{16}	$\{V_{12} > V_{14}\}$	$= 2$					1	2	0	0	0	0	0	0	B_6
7	$-$	$V_{12} - V_{14}$	V_{15}, V_{16}	$\{V_{12} < V_{14}\}$	$= n-1-(\beta)$					1	n-1	0	0	0	0	0	0	B_7
8	$+$	$V_{12} + V_{15}$	V_{17}, V_{18}	$\{V_{12}=V_{15}\}$	$= 2 + 0 = 2$					1	2	0	0	0	0	0	0	B_8
9	$+$	$V_{12} + V_{15}$	V_{17}, V_{18}	$\{V_{12} < V_{15}\}$	$= 2 = A_1$					1	2	0	0	0	0	0	0	B_9
10	\times	$V_{12} \times V_{15}$	V_{17}, V_{18}	$\{V_{12}=V_{15}\}$	$= B_1 \cdot 1 = B_1 A_1$					1	2	0	0	0	0	0	0	B_{10}
11	$+$	$V_{12} + V_{15}$	V_{17}, V_{18}	$\{V_{12} < V_{15}\}$	$= 1 \cdot 2n-1 + B_1 \cdot \frac{2n}{2}$					1	2	0	0	0	0	0	0	B_{11}
12	$+$	$V_{12} + V_{15}$	V_{17}, V_{18}	$\{V_{12} > V_{15}\}$	$= n + 2 - (\beta)$					1	n-2	0	0	0	0	0	0	B_{12}
13	$(-$	$V_{12} - V_{15}$	V_{17}, V_{18}	$\{V_{12}=V_{15}\}$	$= 2n-1$					1	2n-1	0	0	0	0	0	0	B_{13}
14	$+$	$V_{12} + V_{15}$	V_{17}, V_{18}	$\{V_{12} < V_{15}\}$	$= 2 + 1 = 3$					1	3	0	0	0	0	0	0	B_{14}
15	$+$	$V_{12} + V_{15}$	V_{17}, V_{18}	$\{V_{12} > V_{15}\}$	$= 2 = A_2$					1	2	0	0	0	0	0	0	B_{15}
16	\times	$V_{12} \times V_{15}$	V_{17}, V_{18}	$\{V_{12}=V_{15}\}$	$= B_2 \cdot 1 = B_2 A_2$					1	2	0	0	0	0	0	0	B_{16}
17	$-$	$V_{12} - V_{15}$	V_{17}, V_{18}	$\{V_{12} < V_{15}\}$	$= 2 - 2 = 0$					1	2n-2	0	0	0	0	0	0	B_{17}
18	$+$	$V_{12} + V_{15}$	V_{17}, V_{18}	$\{V_{12} > V_{15}\}$	$= -3 + 1 = 4$					1	4	0	0	0	0	0	0	B_{18}
19	$+$	$V_{12} + V_{15}$	V_{17}, V_{18}	$\{V_{12} < V_{15}\}$	$= 2 = -2$					1	2n-2	4	0	0	0	0	0	B_{19}
20	\times	$V_{12} \times V_{15}$	V_{17}, V_{18}	$\{V_{12}=V_{15}\}$	$= 2n-2 - 2 = A_2$					1	2n-2	0	0	0	0	0	0	B_{20}
21	\times	$V_{12} \times V_{15}$	V_{17}, V_{18}	$\{V_{12} < V_{15}\}$	$= B_2 \cdot 2 - 2 = B_2 - B_2 A_2$					1	2	0	0	0	0	0	0	B_{21}
22	$+$	$V_{12} + V_{15}$	V_{17}, V_{18}	$\{V_{12} > V_{15}\}$	$= A_2 + B_2 A_2 + B_2 A_2$					1	2	0	0	0	0	0	0	B_{22}
23	$-$	$V_{12} - V_{15}$	V_{17}, V_{18}	$\{V_{12} < V_{15}\}$	$= n-3-(\beta)$					1	n-3	0	0	0	0	0	0	B_{23}
24	$+$	$V_{12} + V_{15}$	V_{19}, V_{20}	$\{V_{12}=V_{15}\}$	$\rightarrow R_1$					1	0	0	0	0	0	0	0	B_{24}
25	$+$	$V_{12} + V_{15}$	V_{19}, V_{20}	$\{V_{12} < V_{15}\}$ by a Variable cond. $V_{12} = V_{15}$ by a Variable cond.	$\rightarrow R_2$					1	0	0	0	0	0	0	0	B_{25}

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

A Primeira Linguagem: Plankalkül (1946)

- Desenvolvida por: Konrad Zuse (1910 – 1995)
- Não chegou a ser implementada até 1975
- Considerada por Zuse como um exercício mental
- Primeira concepção de um compilador: Ieria comandos nessa linguagem e automaticamente perfuraria cartões com os comandos em linguagem de máquina



subrotina

```

Plankalkül
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
attribution
conditional

```

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

7

O problema da abstração

Programa em linguagem Assembly:

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		

8

O problema da abstração

Seria importante ter algum programa traduzisse códigos em linguagem humana (alto nível) para linguagem de máquina!



9

Compilador

Programa
numa
Linguagem de
Alto Nível

(calendario.c)



Compilador
(gcc)



Programa
em
Código
Máquina

(calendario.exe)



10

Compilador

- Traduz uma **linguagem de alto nível** na **linguagem maquina** da arquitetura destino;
- Depois de compilado, o **programa é específico** da **máquina** bem como ambiente executado;
- 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.

11

Compilador



- Primeiro Compilador: A-0 (1952)
- Desenvolvido por: **Grace Hopper**, programadora do Harvard Mark I;
- 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

12



13

Gerações das Linguagens de Programação

- 1ª Geração (Baixo Nível)
 - Linguagens em Nível de Máquina (Código binário)
- 2ª Geração (Baixo Nível)
 - Linguagens de Montagem (Assembly);
 - A linguagem **ASSEMBLY** é única para cada tipo de CPU (Específica para arquitetura da CPU).

```

; Programa p/ iniciacao do porto B e pôr os seus pinos a '1' lógico
; Version 1.0 Date: 10.10.1999. MCU:PIC16F84 Written by: John Smith

; Declaração e configuração do processador
PROCESSOR 16F84 ; Tipo de processador
#include "p16f84.inc" ; Inclui ficheiro

; Directiva
_CONFIG _CP_OFF & _WDT_OFF & _PWRTE_ON & _XT_OSC

; Instrução
org 0x00 Main ; Início do programa
goto Main ; Vector de reset
; Ir p/ o inicio do programa Main
;

; Comentário
; Início do programa main
; Selecionar banco 1 de memória
; Pinos do porto B de saída
; Selecionar banco 0 de memória

; Operando
Main
BANK1
movwf 0x00
movwf TRSB
BANK0
movwf 0xFF
movwf PORTB ; Tudo a '1' lógico no porto B
; O programa permanece no loop
Loop
goto Loop ; Para assinalar o fim do programa
end
  
```

Fonte: <http://camilahamdan.wikidot.com/wiki:lp-aula-ii-geracoes-da-lp>

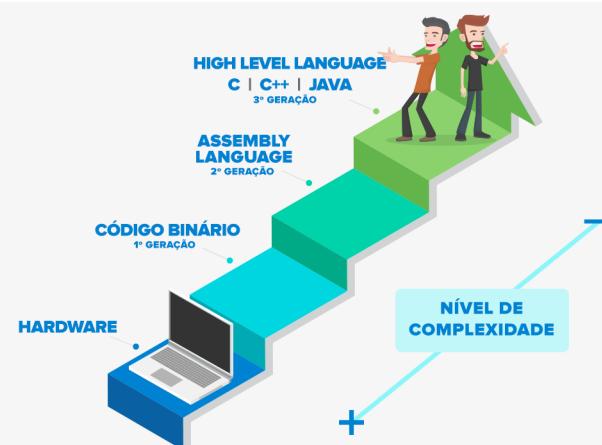
14

Gerações das Linguagens de Programação

- 3^a Geração (Alto Nível)
 - Linguagens procedurais (C, Pascal, Basic, Cobol, ...)
- 4^a Geração (Alto Nível)
 - Linguagens de Aplicação (SQL, PL-SQL, Progress 4GL, ...)
- 5^a Geração (Alto Nível)
 - Linguagens para Inteligência Artificial (Prolog) e linguagens funcionais (LISP)
- 6^a Geração (Alto Nível)
 - Linguagens para Redes Neurais

15

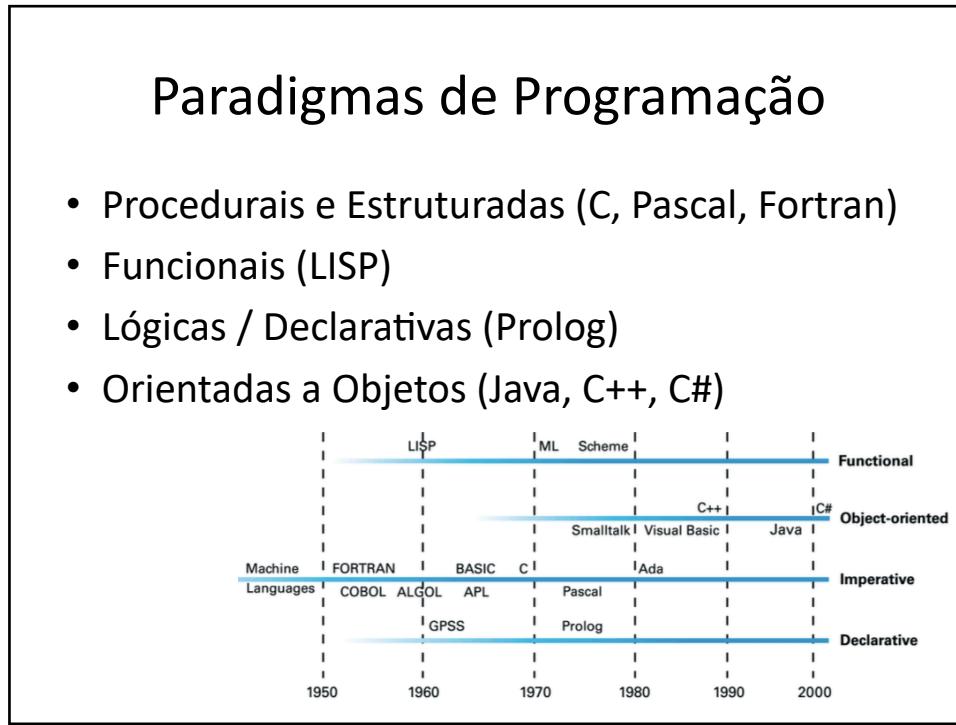
Gerações das Linguagens de Programação

Fonte: <http://encurtador.com.br/qvAV4>

16



17



18

Paradigmas de Programação

- Linguagens Procedurais e Estruturadas:
 - Para programar um computador
 - PROGRAMA – Estrutura de 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.

19

Paradigmas de Programação

- Linguagens Imperativas (C, Pascal, Fortran)

```

int factorial;
int n;
int i;

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

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

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

A primeira parte do programa consiste na declaração dos dados

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

20

Paradigmas de Programação

- Linguagens Funcionais (LISP):
 - 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;
 - Em termos de indústria não tiveram grande aceitação, embora alguns software a utilizem (AutoCAD)

21

Paradigmas de Programação

- Linguagens Lógicas (Prolog):
 - 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;
 - Utilizada para inferências em IA

22

Paradigmas de Programação

- Linguagens Lógicas (Prolog):

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

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

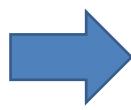
```
avo(X,Y) :- pai(X,Z), pai(Z,Y).
irmao(X,Y) :- pai(X,Z), pai(Y,Z).
```

Interrogações (o que perguntamos ao sistema):

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

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

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



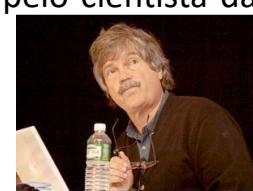
SPARQL – Simple Protocol And Rdf Query Language

```
1. PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
2. PREFIX an: <http://www.exemplo.com.br/animais#>
3.
4. SELECT ?a ?e
5. WHERE
6. {
7.   ?a rdf:Class an:animal;
8.   an:carnívoro ?e.
9. }
10.
11. ASK
12. WHERE
13. {
14.   ?a rdf:Class an:animal;
15.   an:carnívoro ?e.
16. }
```

23

Paradigmas de Programação

- Linguagens Orientada a objetos (Java,...):
 - Tem como objetivo reaproveitar ao máximo o código desenvolvido;
 - É baseada em elementos do mundo real;
 - Trabalha com os conceitos de: classes, instâncias, métodos, atributos e relacionamentos;
 - Este paradigma foi desenvolvido pelo cientista da computação Alan Kay.



24

Paradigmas de Programação

- Classificação Detalhada:
 - Linguagens Imperativas:
 - Procedurais;
 - Estruturadas; e
 - Orientada a Objetos
 - Linguagens Declarativas:
 - Lógicas;
 - Funcionais.

25



Tipos de Linguagem de Programação



Fonte: CAMARGO, Heloisa de Arruda

26

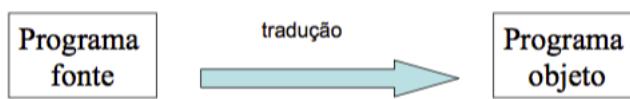
Tipos de Linguagem de Programação

- Pode-se entender esses tipos de linguagens de programação como:
 - Métodos de Implementação de Linguagem
- É como uma linguagem de programação se comunica com o computador;
- Tipos:
 - Compilada;
 - Interpretada; e
 - Implementação híbrida.

27

Tipos de Linguagem de Programação

- Linguagens Compiladas:
 - Programas são traduzidos para linguagem de máquina e são executados diretamente no computador;
 - Envolve dois processos distintos:
 - Tradução (compilação);
 - Execução.
 - A execução é iniciada depois que a tradução é concluída;
 - A execução não tem acesso ao programa fonte;
 - Vantagem: execução rápida.

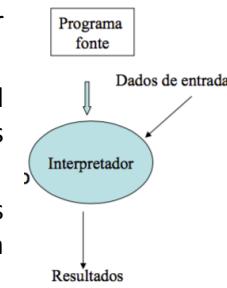


28

Tipos de Linguagem de Programação

- **Linguagens Interpretada:**

- O interpretador “executa” diretamente as instruções do programa fonte, sem traduzir para linguagem de máquina
- Simula, por software, uma máquina virtual onde o ciclo de execução entende os comandos da linguagem de alto nível
- Desvantagem: execução de 10 a 100 vezes mais lenta, devido ao passo de decodificação da instrução de alto nível, que é mais complexa
- Tem acesso ao programa fonte, para depuração ou mesmo para alterar o código sendo executado



29

Tipos de Linguagem de Programação

- **Linguagens Híbridas:**

- Mescla compilação com interpretação;
- Programas fonte são traduzidos para uma linguagem intermediaria que é interpretada;
- Tem maior portabilidade que uma linguagem compilada;
- São mais rápidas que uma linguagem interpretada
 - instruções intermediarias são projetadas para serem interpretadas facilmente.

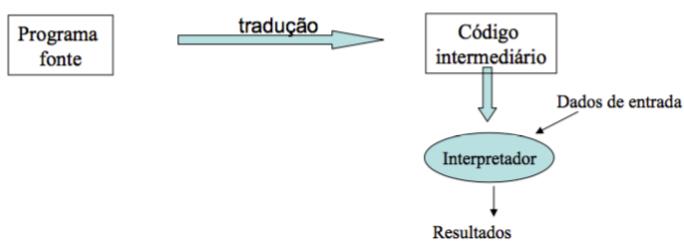
30

Tipos de Linguagem de Programação

- Linguagens Híbridas:

- Exemplo:

- Pearl
- Java (primeiras versões)
 - » Código intermediário :bytecodes
 - » Algumas implementações usam compilação JIT
 - » Alguns sistemas implementam compilação de bytecodes



31

Links Interessantes

- Early Programming: Crash Course Computer Science #10

<https://www.youtube.com/watch?v=nwDq4adJwzM&list=PL8dPuuaLjXtNIUrzyH5r6jN9ullgZBpdo&index=11>

- The First Programming Languages: Crash Course Computer Science #11

<https://www.youtube.com/watch?v=RU1u-j57db8&list=PL8dPuuaLjXtNIUrzyH5r6jN9ullgZB+pdo&index=12>

40

Referencial Teórico

BRAGA, R. T. V. **História das Linguagens de Programação** – Slides da Aula 8 - USP.

CAMARGO, Heloisa de Arruda. **Paradigmas de Linguagens de Programação** - Slides.

SEBESTA, R. W. **Conceitos de Linguagens de Programação**, 9 edição, Bookman, São Paulo, 2011.