Compiladores

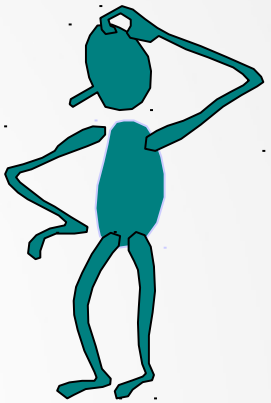
Introdução

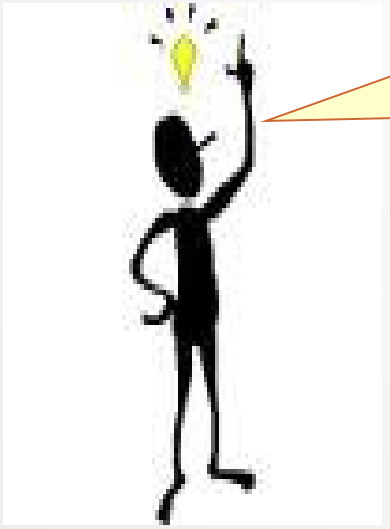
Profa Ana Carolina Lorena

2º semestre 2016

Linguagens de Programação

Como *softwares* são construídos?





Linguagens de programação são utilizadas para a construção de programas em computadores

Linguagens de Programação

Uma linguagem de programação pode ser definida como:

- Conjunto limitado de símbolos e comandos, utilizados para criar programas;

- Método padronizado para expressar instruções para um computador

Por meio dela se estabelece uma comunicação com o computador, fazendo com que ele compreenda e execute o que o programador determinar

Linguagens de Programação

Uma linguagem (natural ou de programação) é formada por:

- Parte léxica: conjunto de termos permitidos;
- Sintaxe: a forma ou estrutura das expressões;
- Semântica: o significado das expressões.

Parte léxica

Determina os símbolos que compõem a linguagem

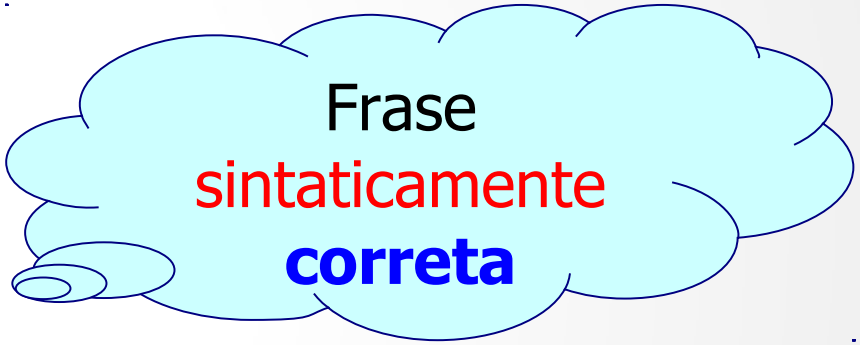
```
int i = 0;
```

Possui os componentes:
`int`: palavra reservada que designa o tipo inteiro
`i`: identificador de uma variável
`=`: operador de atribuição
`0`: valor inteiro
`;`: símbolo que designa o final da sentença

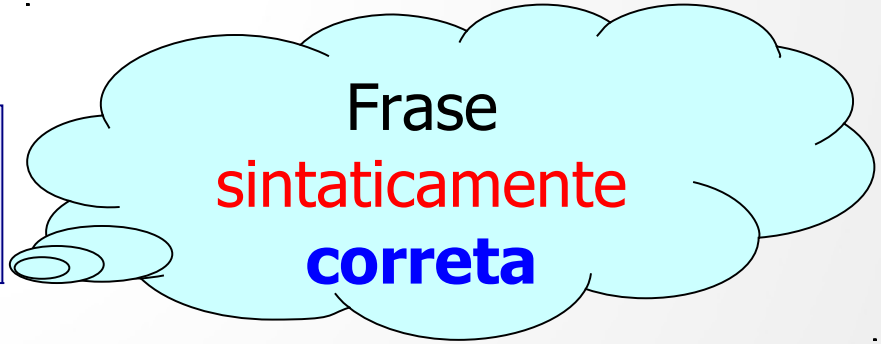
Sintaxe

Sintaxe determina regras de como se escreve de forma correta em uma linguagem (regras de escrita)

Os seguintes países fazem parte do Mercosul: Brasil, Argentina, Paraguai, Uruguai e Venezuela

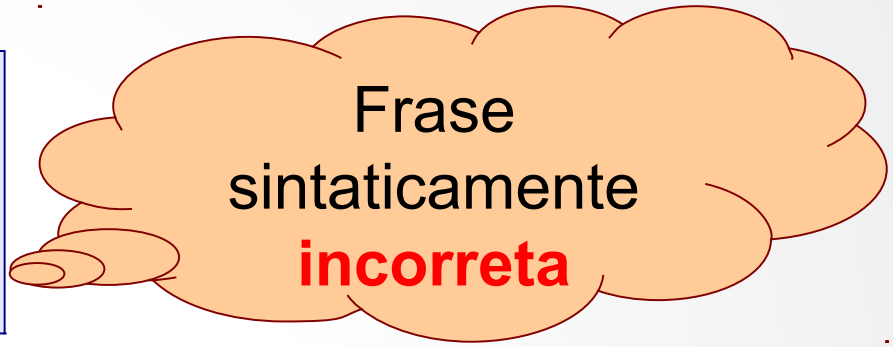


O Brasil está localizado na América Central



Sintaxe

Os serguintes países faz parte do Mercosul: Brasil, Argentina, Paraguai, Uruguai e Venezuela



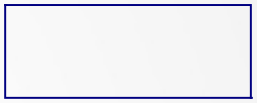
Em linguagem natural a sintaxe é conhecida como Gramática

Sintaxe - Programação

Considere o comando para a criação e declaração de uma variável, em Java

```
int idade;
```

idade



Considere o comando para a atribuição de valor à uma variável, em Java

```
idade = 10;
```

idade

Estes comandos estão sintaticamente **corretos**, na linguagem de programação Java

Sintaxe - Programação

Considere os comandos para a criação e declaração de uma variável, em Java

```
Int idade;          Erro: Int
int idade           Erro: Falta ;
```

Considere o comando para a atribuição de valor à uma variável, em Java

```
idade := 10;        Erro: :=
idade = 10          Erro: Falta ;
```

Estes comandos estão sintaticamente **incorretos**, na linguagem de programação Java

Semântica

Semântica estuda o significado das palavras ou frases

Considere as frases:

- O Sol é uma estrela
- Brasil está localizado na América do Sul

Semanticamente corretas

- Tem dia que de noite é assim mesmo
- Pá daqui, pá dali

Semanticamente incorretas

Semântica - Programação

Considere os comandos, em Java:

```
int idade;
idade = 10;
```

Comandos sintática e semanticamente corretos

Considere os comandos, em Java:

```
int idade;
idade = 10.7;
```

Comando de atribuição semanticamente incorreto

Classificação Ling. de Programação

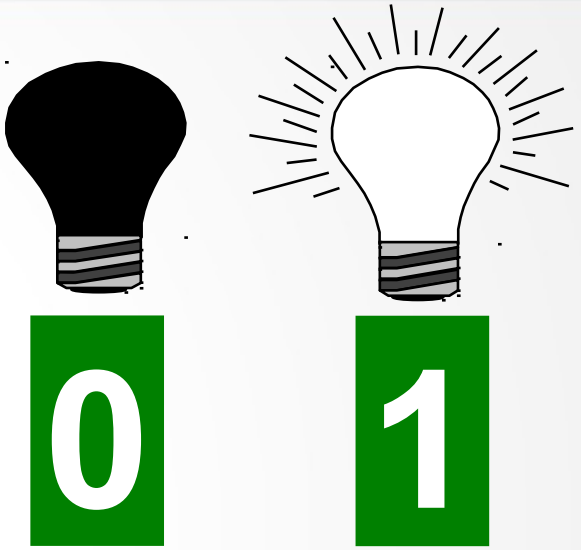
A proximidade que a linguagem de programação tem com a humana determina sua classe:

- Linguagem de máquina (primeira geração)
- Linguagem *assembly* - de montagem (segunda geração)
- Linguagem de alto nível (terceira geração)
- Linguagem de muito alto nível (quarta geração)

Ling. de Máquina (1ª geração)

- Linguagem de máquina é o código que o computador executa diretamente

- É composta de 0s e 1s, e neste caso é conhecida como linguagem binária



Instruções de linguagem de máquina são representadas por códigos na forma de palavras binárias

Exemplo:

```
01000101000111010101010000100101010...
```

Ling. de Máquina (1ª geração)

Um programa em linguagem de máquina é uma série de 0s e 1s, ordenados de forma que alguns representam:

- **códigos de instruções**
- **os dados que serão processados**
- **ou indicam onde esses dados estão armazenados**

Ling. de Máquina (1ª geração)

- Cada família de computadores possui sua própria linguagem de máquina
- Para atender às características de um determinado processador
- Por isso é totalmente dependente do *hardware*, e do fabricante do processador

Ling. de Máquina (1ª geração)

- Ex. Linguagem Hexadecimal: sequência de bits pode ser representada por números hexadecimais
  - Transforma cada 4 bits em um dígito hexadecimal

binário	hexadecimal	binário	hexadecimal
0000	0	0001	1
0010	2	0011	3
0100	4	0101	5
0110	6	0111	7
1000	8	1001	9
1010	A	1011	B
1100	C	1101	D
1110	E	1111	F

Ling. de Máquina (1ª geração)

Exemplo de Programa Escrito na Linguagem Hexadecimal

```
0000000 feed face 0000 0012 0000 0000 0000 0001
0000010 0000 0003 0000 016c 0000 2000 0000 0001
0000020 0000 0104 0000 0000 0000 0000 0000 0000
0000030 0000 0000 0000 0000 0000 0090 0000 0188
0000040 0000 0090 0000 0007 0000 0007 0000 0003
0000050 0000 0000 5f5f 7465 7874 0000 0000 0000
0000060 0000 0000 5f5f 5445 5854 0000 0000 0000
0000070 0000 0000 0000 0000 0000 008c 0000 0188
0000080 0000 0002 0000 0218 0000 000c 8000 0400
0000090 0000 0000 0000 0000 5f5f 7069 6373 796d
00000a0 626f 6c73 7475 6231 5f5f 5445 5854 0000
00000b0 0000 0000 0000 0000 0000 008c 0000 0000
00000c0 0000 0214 0000 0000 0000 0000 0000 0000
00000d0 8000 0008 0000 0000 0000 0020 5f5f 6e6c
00000e0 5f73 796d 626f 6c5f 7074 7200 5f5f 4441
00000f0 5441 0000 0000 0000 0000 0000 0000 008c
0000100 0000 0004 0000 0214 0000 0002 0000 0000
0000110 0000 0000 0000 0006 0000 0000 0000 0000
0000120 0000 0002 0000 0018 0000 027c 0000 0003
0000130 0000 02a0 0000 0014 0000 000b 0000 0050
0000140 0000 0000 0000 0000 0000 0000 0000 0002
0000150 0000 0002 0000 0001 0000 0000 0000 0000
0000160 0000 0000 0000 0000 0000 0000 0000 0000
0000170 0000 0278 0000 0001 0000 0000 0000 0000
0000180 0000 0000 0000 0000 bfc1 fff8 9421 ffd0
0000190 7c3e 0b78 7c08 02a6 429f 0005 7d48 02a6
00001a0 7c08 03a6 3c4a 0000 8042 0078 8002 0000
00001b0 7c03 0378 8021 0000 bbc1 fff8 4e80 0020
00001c0 bfc1 fff8 9421 ffd0 7c3e 0b78 7c08 02a6
00001d0 429f 0005 7d48 02a6 7c08 03a6 907e 0048
00001e0 3c4a 0000 8042 0040 8042 0000 801e 0048
00001f0 7f80 1000 409d 0014 3c4a 0000 8042 0040
0000200 801e 0048 9002 0000 8021 0000 bbc1 fff8
0000210 4e80 0020 0000 0000 ab00 0074 0000 008c
0000220 a100 0000 0000 004c ac00 0070 0000 008c
0000230 a100 0040 0000 004c ab00 005c 0000 008c
0000240 a100 0000 0000 004c ac00 0058 0000 008c
0000250 a100 0040 0000 004c ab00 0020 0000 008c
0000260 a100 0000 0000 0014 ac00 001c 0000 008c
0000270 a100 0078 0000 0014 0000 0002 0000 0006
0000280 0f01 0000 0000 0038 0000 0001 0f01 0000
0000290 0000 0000 0000 000e 0100 0000 0000 0000
00002a0 005f 6d61 7800 5f69 6e73 6572 7400 5f5f
00002b0 6d61 7800
00002b4
```

Programação ainda impraticável... O que fazer??

Ling. *Assembly* (2ª geração)

- A linguagem *assembly* (linguagem de montagem) usa nomes (mnemônicos) e símbolos em lugar dos números
  - Utiliza palavras abreviadas (mnemônicos) indicando operações

`MOV R1, R2`

- mnemônico MOV (abreviação de MOVE)
- dois registradores como parâmetros: R1 e R2
- processador comanda o movimento do conteúdo de R2 para R1
- equivalente à instrução Java `R1 = R2;`

Ling. *Assembly* (2ª geração)

- *Assembly* é uma linguagem simbólica
  - Utiliza símbolos em sua estrutura
  - Não é composta de números binários ou hexadecimais

`ADD R1, R2`

- mnemônico ADD (abreviação de ADDITION)
- dois registradores como parâmetros: R1 e R2
- processador comanda a adição do conteúdo de R1 ao conteúdo de R2, e o resultado é armazenado em R1
- equivalente à instrução em Java `R1=R1+R2;`

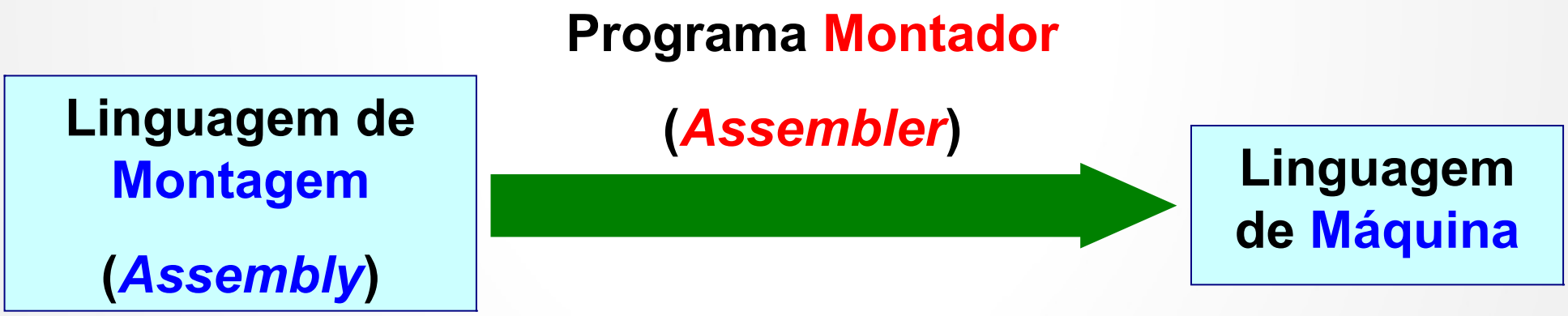
Ling. *Assembly* (2ª geração)

Simplificações da linguagem *assembly*:

- Escolhe nomes descritivos para as posições de memória
  - Exemplo: arquitetura x86 (Intel) são:
    - EAX - registrador acumulador
    - ECX - registrador contador
    - EDX - registrador de dados
- Usa mnemônicos para representar códigos de operação
  - ADD
  - MOV
  - etc

Ling. *Assembly* (2ª geração)

- A tradução/conversão da linguagem *assembly* para a linguagem de máquina se chama montagem
  - E é feita por um programa chamado montador (*assembler*)



Ling. *Assembly* (2ª geração)

Microprocessador 8086

Ling. de máquina Assembly
B0 FF MOV AL, 0FFh
A2 00 20 MOV [2000h], AL

Microprocessador 6800

Ling. de máquina Assembly
86 FF LDA A, # FF
97 00 20 STA A, 00

Microprocessador 6502

Ling. de máquina Assembly
A9 FF LDA # FF
8D 00 20 STA 00

- A linguagem *assembly* é única para cada tipo de CPU
  - Cada processador possui um conjunto de instruções próprio
- Utiliza instruções de baixo nível que operam diretamente com registros e memórias.
  - Não pode ser reutilizada em famílias de processadores diferentes

Tarefa: colocar o valor 255 na posição de memória 2000h

Ling. *Assembly* (2ª geração)

- Famílias de processadores mantém um certo nível de interoperabilidade
  - Quem sabe programar em *Assembly* em um 8086 saberá programar em um 80286, pois esta é a família de microprocessadores da Intel
  - Outros microprocessadores como 68000, 68020, etc, constituem a família de microprocessadores da Motorola
    - Os processadores Motorola geralmente estão presentes em microcomputadores como o MacIntosh

Ling. de Alto Nível (3ª geração)

- Apresentam uma sintaxe mais próxima da linguagem natural
  - Usam palavras reservadas extraídas do vocabulário corrente (`int`, `public`, `if`, ...)

```
class HelloWorld
{
   public static void main (String[] args)
   {
      System.out.println ("Bem Vindos ao ICT");
   }
}
```

Ling. de Alto Nível (3ª geração)

Linguagens de alto nível foram estruturadas buscando refletir melhor os processos humanos de solução de problemas

Exemplos de linguagens de alto nível:

- Fortran: voltada para a área de engenharia
- Cobol: voltada para a área comercial
- Pascal: voltada para a área acadêmica

Ling. de Alto Nível (3ª geração)

Algumas linguagens de programação, e o ano em que foram desenvolvidas:

1957	FORTRAN
1958	ALGOL
1960	LISP
1960	COBOL
1962	APL
1962	SIMULA
1964	BASIC
1964	PL/1
1966	ISWIM
1970	Prolog
1972	C

1975	Pascal
1975	Scheme
1977	OPS5
1978	CSP
1978	FP
1980	dBase II
1983	Smalltalk
1983	Ada
1983	Parlog
1984	Standard ML
1986	C++

1986	CLP(R)
1986	Eiffel
1988	CLOS
1988	Mathematica
1988	Oberon
1990	Haskell
1995	Delphi
1995	Java

Ling. de Alto Nível (3ª geração)

Os programas escritos em linguagens de alto nível são denominados código fonte

- Os códigos fonte devem ser convertidos para a linguagem de máquina

Ling. Muito Alto Nível (4ª geração)

- As linguagens de quarta geração têm uma estrutura ainda mais próxima da linguagem humana
- Definem “o que” deve ser feito, e não “como” deve ser feito
- Exemplo: linguagens de consulta a banco de dados, como SQL

Ling. Muito Alto Nível (4ª geração)

Ex. SQL, linguagem de consulta para manipular bases de dados

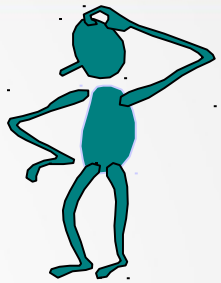
nome	email	telefone	salario	cargo	*id
João da Silva	jsilva@swhere.com	7363-2334	2300	Gerente	1034
Carlos Ribas	cribas@cblanca.org	8334-3238	1800	Auxiliar	2938
Madalena Reis	mreis@portal.com	6451-5672	2000	Contador	7567
Patrícia Horws	phorws@mail.com	4513-6564	2900	Gerente	2314
Carlito Fox	cfox@uol.com.br	5642-7873	1500	Auxiliar	5622
Ricardo Alves	ralves@portal.com	9302-4320	2000	Programador	6762

Apresentar os dados dos campos **nom**e e **telefone** da tabela Funcionario:

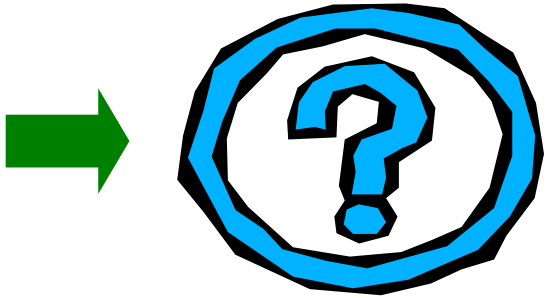
```
select nome, telefone FROM Funcionario;
```

Ling. Programação: Tradução

Como fazer a tradução do código fonte para o código executável?



Linguagem de Alto Nível (Código Fonte)



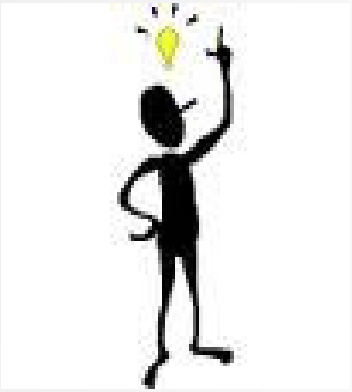
Linguagem de Máquina (Código Executável)

```
class HelloWorld
{
    public static void main (String[]args)
    {
        System.out.println("ICT");
    }
}
```

Execução

UNIFESP
UNIVERSIDADE FEDERAL DE SÃO PAULO
1933

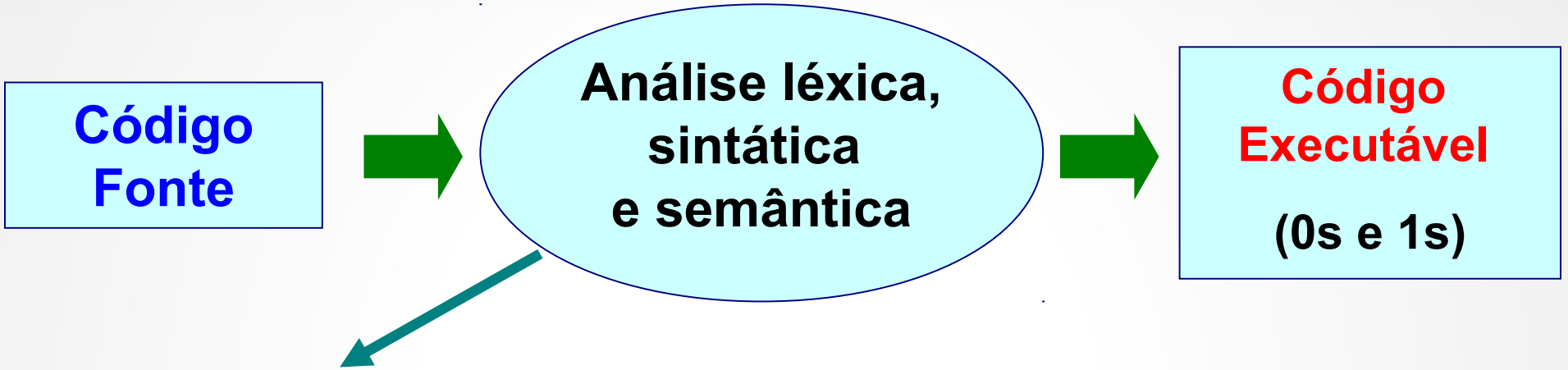
Ling. Programação: Tradução



Compilação e **interpretação** são os processos pelos quais o código fonte é traduzido em código de máquina

Há ainda a possibilidade de uma mescla das duas (em um sistema híbrido), como é o caso da linguagem Java

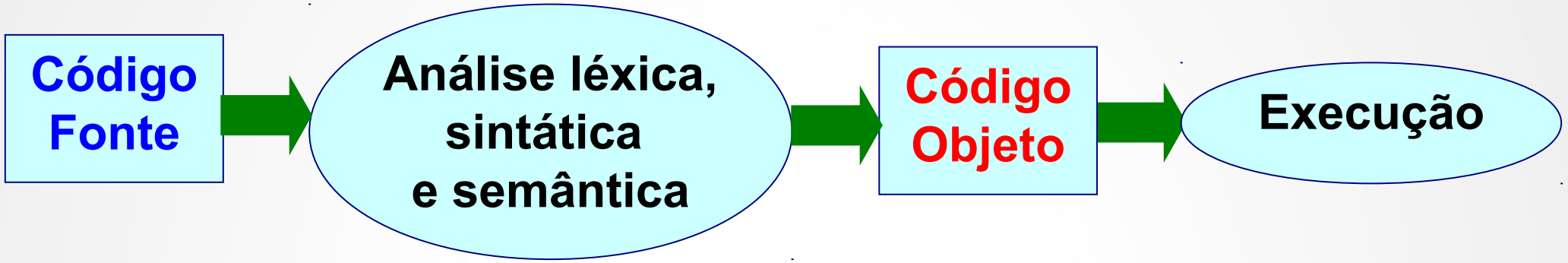
Compilação



Estas análises são feitas em todo o código fonte, para depois gerar o código executável

```
class HelloWorld
{
   public static void main (String[]args)
   {
      System.out.println("ICT");
   }
}
```

Compilação



- O arquivo fonte é escrito pelo programador, em uma linguagem de programação
- O arquivo produzido pelo compilador é normalmente identificado como código objeto
- O código objeto consiste nas instruções de baixo nível que podem ser executadas pelo computador

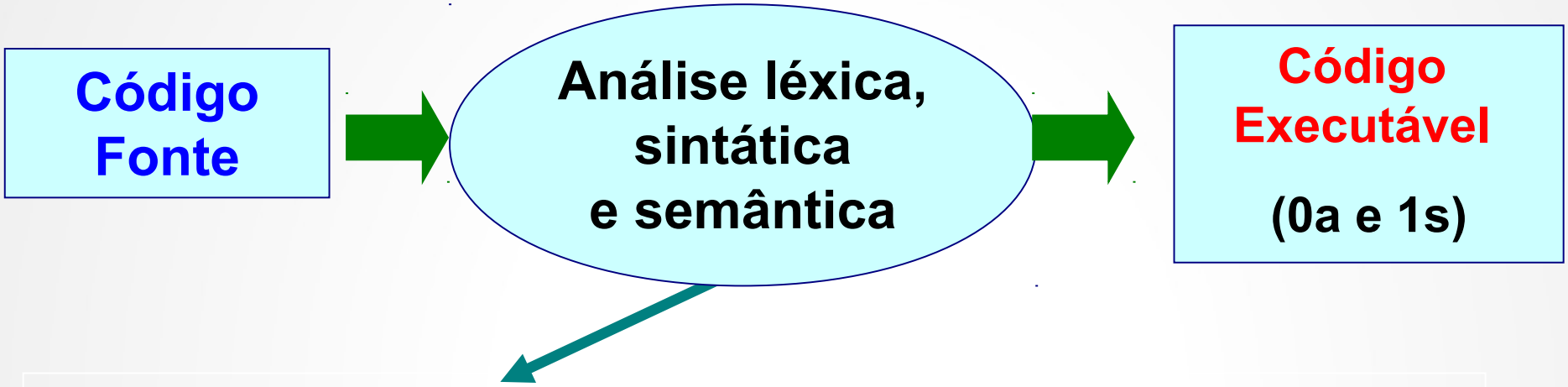
Compilação

- O código executável produzido não é portável
  - Diferentes compiladores são construídos para as diferentes arquiteturas de processadores
- Exemplos de linguagens compiladas:
  - C, Pascal, Fortran, C++, Cobol

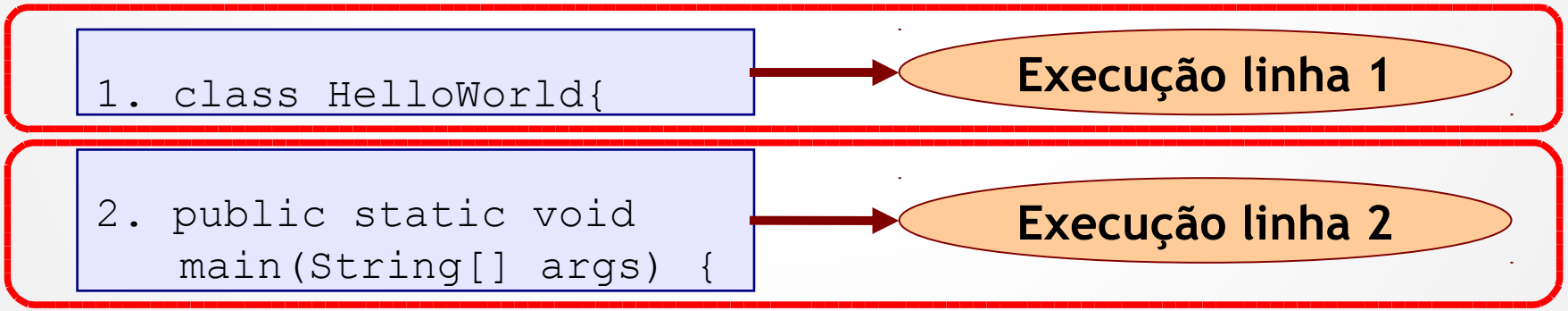
Compilação

Vantagens	Desvantagens
Gera código executável	
Não é necessário recompilar novamente para executar o programa	É dependente de plataforma (código de máquina gerado é preparado para rodar em uma plataforma específica)
A primeira vez que for executar um programa será relativamente demorado, pois será necessário compilá-lo. Mas nas próximas execuções, será mais rápido	
É preciso ter um compilador na máquina onde se desenvolve o programa, mas não onde ele é executado	

Interpretação



Estas análises são feitas em cada linha por vez



Interpretação

- Cada instrução do programa fonte é lida e transformada em linguagem de máquina do computador hospedeiro, e **executada imediatamente**

- Quem faz este processo é um programa denominado **interpretador**
  - Ele está preparado para usar a linguagem de máquina do computador hospedeiro

Interpretação

- Um interpretador é um programa que executa repetidamente a seguinte sequência:
  1. Obter o próximo comando do programa
  2. Determinar que ações devem ser executadas
  3. Executar estas ações

Caso haja alguma linha de código mal codificada
(não respeitando o que foi definido na linguagem),
o programa termina sua execução abruptamente em erro

Interpretação

- Exemplos de linguagens interpretadas:
  - HTML
  - Haskell
  - Lisp

Interpretação

Vantagens	Desvantagens
Não gera um arquivo de código executável	Toda a vez que for executar o programa, terá que ler o código fonte
É independente de plataforma	Quando comparada com a compilação, a primeira vez da execução de uma interpretação é mais rápida. Mas as demais execuções serão mais lentas
É necessário ter o interpretador na máquina onde o código fonte foi desenvolvido, bem como onde o código fonte será executado	

Bibliografia

Capítulo 1 LOUDEN, Kenneth C; SILVA, Flávio S.C. Compiladores: princípios e práticas. São Paulo: Thomson, 2004. 569 p.

Slides:

- Professores Maria Graças B. Marietto, Marcelo Zanchetta, UFABC