

Universidade Federal de Viçosa  
Campus Rio Paranaíba  
Instituto de Ciências Exatas e Tecnológicas

# **SIN 110**

## **Programação**

Sistemas de Informação  
Profa. Rachel Reis  
[rachel.reis@ufv.br](mailto:rachel.reis@ufv.br)

Universidade Federal de Viçosa  
Campus Rio Paranaíba  
Instituto de Ciências Exatas e Tecnológicas

Aula de Hoje

# Linguagens de Programação

Créditos:

**Prof. Guilherme Pena**

# Linguagens de Programação

Para que o computador consiga ler um programa e entender o que fazer, este programa deve ser escrito em uma **linguagem** que o computador entenda

Esta linguagem chama-se **LINGUAGEM DE PROGRAMAÇÃO**



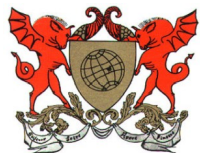
# Etapas de um programa

Etapas de desenvolvimento de um programa:

**Análise:** estuda-se o enunciado do problema para definir os dados de entrada, processamento e dados de saída (O QUE deve ser feito)

**Algoritmo:** utiliza-se ferramentas para descrever COMO resolver o problema identificado

**Codificação:** transforma-se o algoritmo em códigos na linguagem de programação escolhida



# Algoritmo

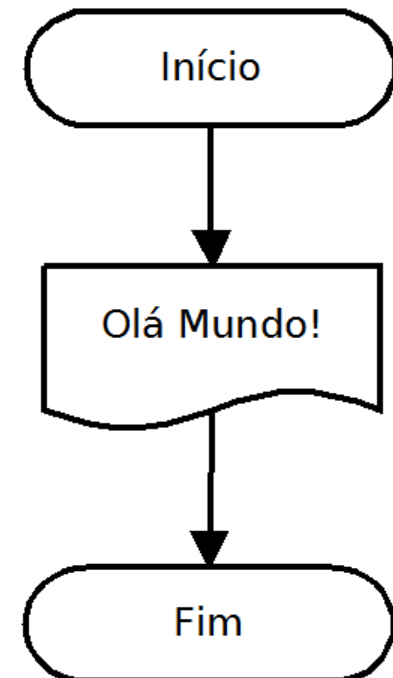
## Descrição narrativa:

1) Mostre a mensagem (“Olá Mundo”)

## Pseudocódigo:

```
algoritmo “Olá Mundo”  
var  
inicio  
    escreva( “Olá Mundo!”)  
fimalgoritmo
```

## Fluxograma:



# Algoritmo

```
#include <stdio.h>
int main(void)
{
    printf("Olá Mundo!");
    return 0;
}
```

→ C

```
begin
    ShowMessage('Olá Mundo!');
end.
```

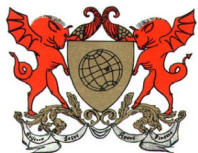
→ Delphi

```
public class AloMundo {
    public static void main(String args[]) {
        System.out.print("Alo Mundo");
    }
}
```

→ Java

PHP

```
<?php
    echo "Olá Mundo!";
?>
```

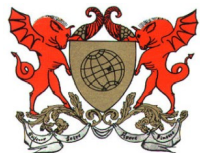


# Linguagens de Programação

São instrumentos para **facilitar a comunicação** entre humanos e computadores a fim de solucionar problemas.

Têm o objetivo de representar alguma informação por meio de uma sequência de símbolos.

É um método padronizado para expressar instruções para um computador.



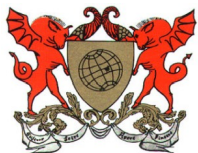
# Linguagens de Programação

Uma linguagem de programação é composta por um **conjunto de regras sintáticas e semânticas** usadas para definir um programa de computador.

- ✓ **Sintaxe:** representação simbólica
- ✓ **Semântica:** o conceito que ela representa

Exemplo: comando “se” na linguagem C

- Sintaxe: `if (<expr>) <instrução>`
- Semântica: se o valor da expressão for verdadeiro, a instrução será executada





# Tipos de Linguagem

## **Linguagens de Máquina ou Baixo Nível:**

- Formada por uma sequência de dígitos binários (0s e 1s)
- Notação facilmente entendida pelo processador mas de difícil compreensão para humanos.

## **Linguagens de Montagem ou *Assembly*:**

- Formada por instruções pré-definidas que são traduções de linguagens de Alto Nível.
- Cada instrução *assembly* gera uma palavra de bits (uma instrução em linguagem de máquina)

## **Linguagens de Alto Nível:**

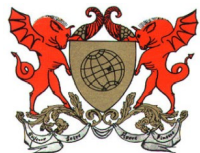
- Aproximam-se das linguagens utilizadas por humanos para expressar problemas e algoritmos



# Linguagem de Máquina

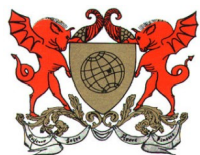
Programação inviável para seres humanos:

```
MZÀ$PÿvèŠÿ]Ē3ÀP, F  
èfF, < uè2ÀëäÀtBª  
ÀuCtà2Àùä-I, "t" < \u€ < "u-IöÄéíYÊ.Žt%~CÛÛô<ì+ërâ<â%.-~Ä%v, vüÿv  
ÿvèÄfÄÿvpÿvüèüêYY<V<FëRÿvpÿvüèWifÄ<â]EU<ìfìHVW<~<F%FpÀu  
'Í!'3Àé•Š~<ØŠ+Īn
```



# Linguagem de Montagem

Endereço	OPcode	Operandos
7C90EBAF	pushfd	
7C90EBB0	sub	esp, 2D0h
7C90EBB6	mov	dword ptr [ebp+FFFFFFDDCh], eax
7C90EBBC	mov	dword ptr [ebp+FFFFFFDD8h], ecx
7C90EBC2	mov	eax, dword ptr [ebp+8]
7C90EBC5	mov	ecx, dword ptr [ebp+4]
7C90EBC8	mov	dword ptr [eax+0Ch], ecx
7C90EBCB	lea	eax, [ebp+FFFFFFD2Ch]
7C90EBD1	mov	dword ptr [eax+000000B8h], ecx



# Linguagem de Alto Nível

```
public class Retangulo
{
    private int base, altura;
    public int calculaArea()
    {
        return (base * altura);
    }
}
```



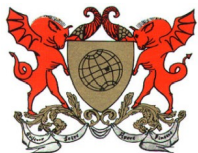
# Domínios de Programação

Os computadores são aplicados em uma infinidade de áreas diferentes.

Linguagens de Programação com objetivos diferentes têm sido desenvolvidas → áreas distintas.

## **Áreas de Aplicação:**

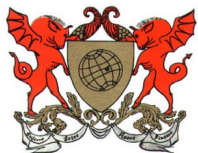
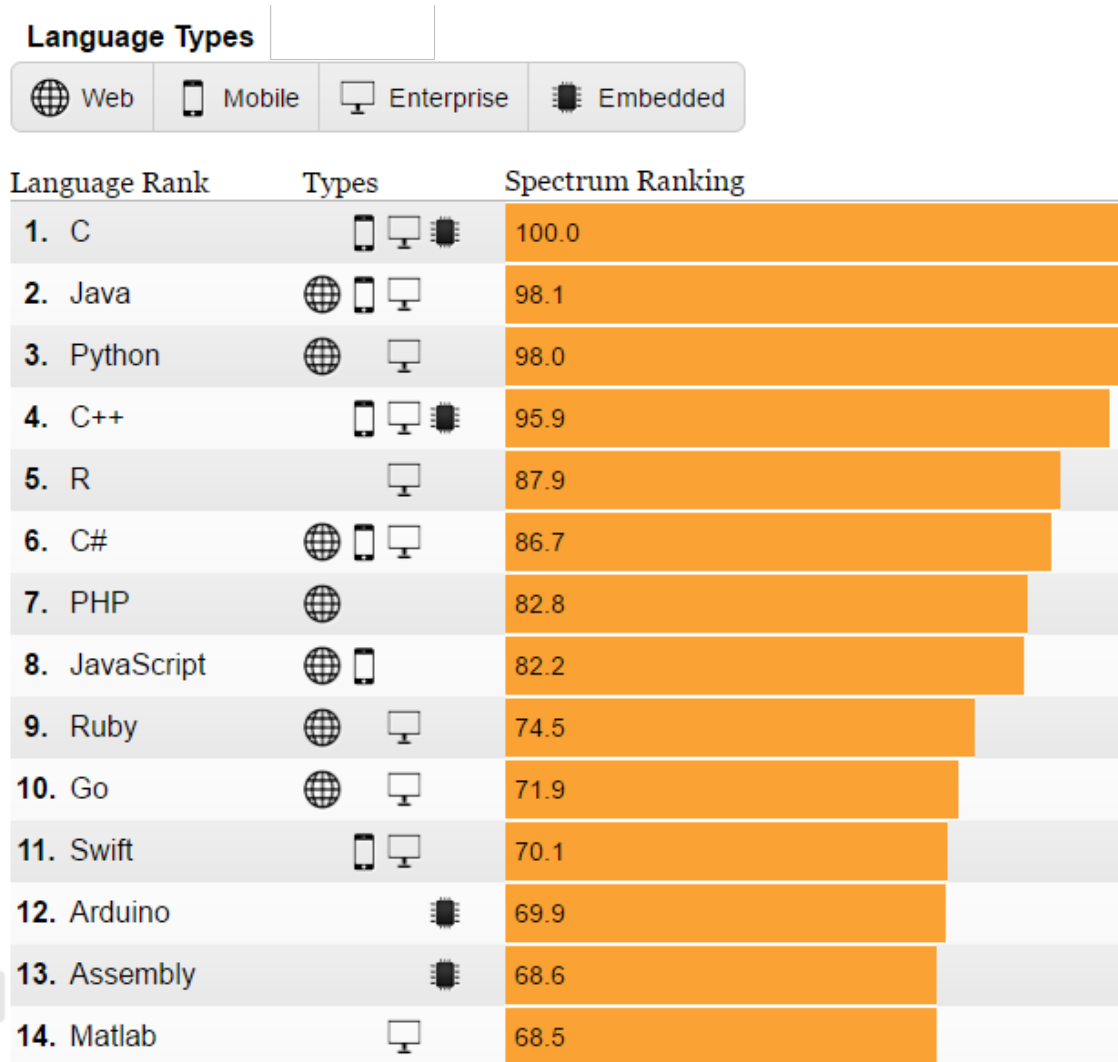
- Aplicações científicas
- Aplicações comerciais
- Inteligência artificial
- Desenvolvimento de software básico
- Desenvolvimento de software para web
- etc.



# Algumas Linguagens

ABC, Ada, Alan, ALF, Algol, Alloy, Amiga E, AMPL, APL, AWK, B, BASIC, BCPL, BETA, Bliss, Blue, Business Rules, C, C++, Charity, CLAIRE, Clean, COBOL, COMAL, cT, DCL, Dialect, Dylan, E, Eiffel, elastiC, Elf, Erlang, Escher, Euphoria, Forth, Fortran, FPL, GNU E, Guile, Gödel, Haskell, Hugo, ICL, Icon, Inform, J, Java, Joy, Juice, K, Lava, LIFE, Limbo, LISP, LOGO, Lua, Matlab, MCPL, Mercury, Miranda, ML, Modula-2, Modula-3, NeoBook, NESL, NetRexx, Oberon, Object Oriented Turing, Objective-C, Obliq, Occam, Octave, Oz, Pascal, Perl, Phantom (Phi), PHP, Pike (LPC), PiXCL, PL/B, PL/I, Pliant, Postscript, Prolog, Python, R, REBOL, Rexx, RPG, RPL/2, Ruby, S, Sather, Scheme, Self, SETL, Simula, Sisal, Smalltalk, SNOBOL, SR, TADS, Tcl, Theta, TOM, V, Visual Basic, Yorick, ZPL

# Ranking de uso das linguagens (2016)



# Algumas Linguagens Malucas

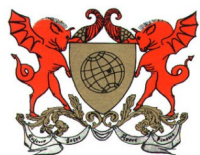
## Brainfuck (Hello World):

```
+++++++ [ >++++ [ >+>++++>+>+><<<<- ] >+>+>->>+ [ < ] <- ] >> .>--- .+++++  
+..+++.>>.<-.<..++  
+.------.------.>>+.>+.
```

## ArnoldC (Hello World):



```
IT'S SHOWTIME  
TALK TO THE HAND "Hello World!"  
YOU HAVE BEEN TERMINATED
```

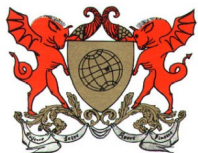




# Algumas Linguagens Malucas

## Ook! (Hello World):

Ook. Ook? Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook.  
Ook. Ook. Ook. Ook. Ook! Ook? Ook? Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook.  
Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook? Ook! Ook! Ook? Ook! Ook? Ook.  
Ook! Ook. Ook. Ook? Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook.  
Ook. Ook. Ook! Ook? Ook? Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook.  
Ook! Ook! Ook? Ook! Ook? Ook. Ook. Ook. Ook! Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook.  
Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook! Ook. Ook! Ook. Ook. Ook. Ook. Ook. Ook.  
Ook. Ook. Ook! Ook. Ook. Ook? Ook. Ook? Ook. Ook? Ook. Ook. Ook. Ook. Ook. Ook.  
Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook! Ook? Ook? Ook. Ook. Ook.  
Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook? Ook! Ook! Ook? Ook! Ook? Ook. Ook! Ook.  
Ook. Ook? Ook. Ook? Ook. Ook? Ook. Ook? Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook.  
Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook.  
Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook. Ook.  
Ook. Ook? Ook! Ook! Ook? Ook! Ook? Ook. Ook! Ook! Ook! Ook! Ook! Ook! Ook! Ook.  
Ook? Ook. Ook? Ook. Ook? Ook. Ook? Ook. Ook? Ook. Ook! Ook. Ook. Ook. Ook. Ook. Ook.  
Ook! Ook. Ook! Ook! Ook! Ook! Ook! Ook! Ook! Ook! Ook! Ook! Ook! Ook! Ook! Ook!  
Ook! Ook! Ook! Ook! Ook! Ook! Ook! Ook! Ook! Ook! Ook! Ook! Ook! Ook! Ook! Ook!  
Ook! Ook. Ook. Ook? Ook. Ook? Ook. Ook. Ook. Ook! Ook.



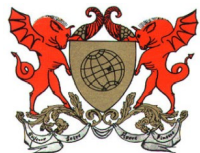
# Algumas Linguagens Malucas



## BIRL



O BIRL (*Bambam's "It's show time" Recursive Language*) é a linguagem de programação mais treze já inventada. Deve ser utilizada apenas por quem realmente constrói fibra e não é água com código. É uma linguagem extremamente simples porém com poder para derrubar todas as árvores do parque Ibirapuera. Programando em BIRL, é verão o ano todo!



UFV - Campus Rio Paranaíba  
Sistemas de Informação

# Paradigma de Programação

Modelo, padrão ou estilo de programação suportado por linguagens que agrupam certas características comuns

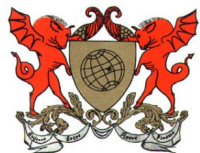
Cada linguagem apresenta uma maneira particular de modelar o que é um programa.

A escolha de um determinado paradigma influencia a forma com que uma aplicação real é modelada do ponto de vista computacional

# Paradigma de Programação

As **Linguagens de Programação** são categorizadas em quatro paradigmas:

- Imperativo
- Orientado a Objetos
- Funcional
- Lógico

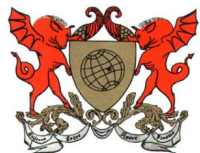


# Paradigma de Programação

## Paradigma Imperativo:

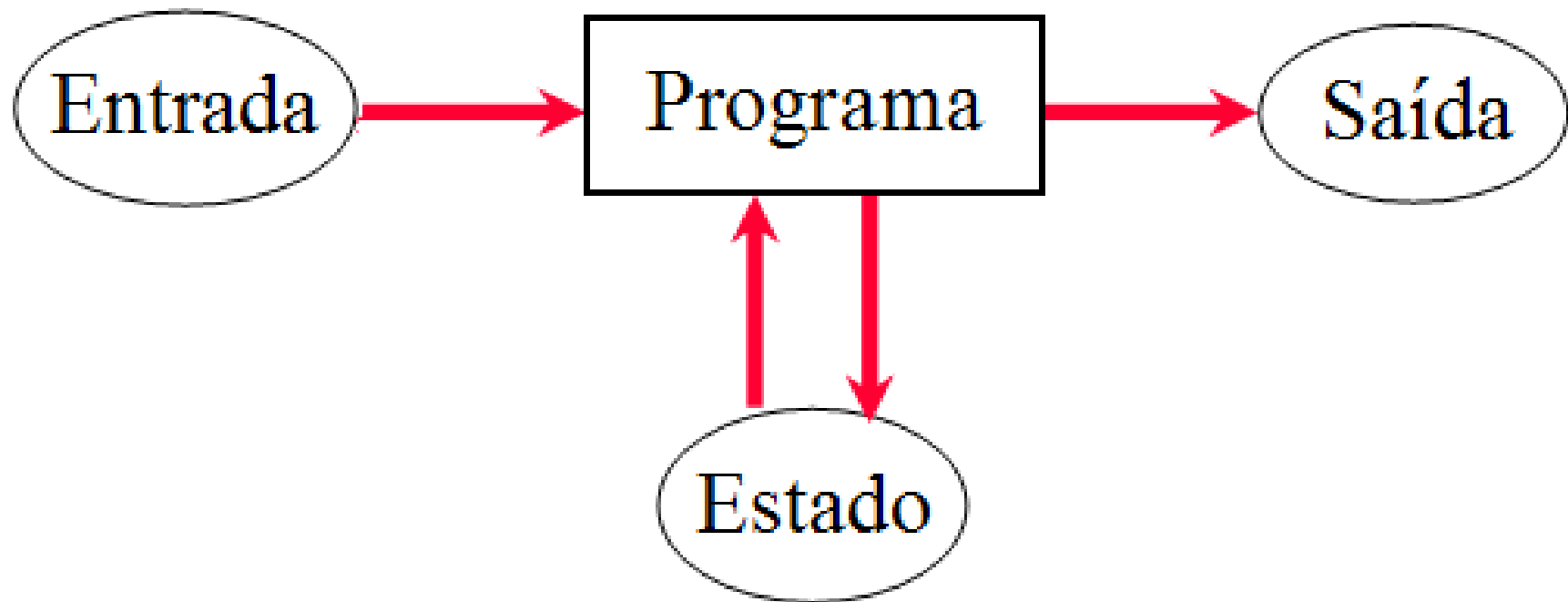
- Fundamenta-se em comandos que manipulam posições de memória através de variáveis (mudam o estado do programa).
- Definem uma sequência de instruções que o computador deve executar.
- Primeiro paradigma a surgir e ainda é o dominante

Exemplos: Fortran, C, Pascal, Cobol



# Paradigma de Programação

## Paradigma Imperativo:



# Paradigma de Programação

## Exemplo:

```
#include <stdio.h>
#include <conio.h>
int main()
{
    int num1, num2, soma;

    printf("Digite dois números: ");
    scanf("%d, %d", &num1, &num2);

    soma = num1 + num2;
    printf("Soma: ", soma);

    getch();
}
```

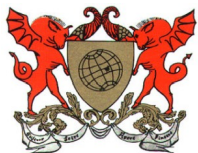


# Paradigma de Programação

## **Paradigma Orientada a Objetos:**

- Muitos autores consideram este paradigma como uma subclassificação do paradigma imperativo
- Idéia : ao invés de programar pensando como a máquina, pode-se programar pensando como humanos
- Sistema : conjunto de objetos representando pequenas partes do problema, que conversam entre si, e que possuem estados (atributos, características) e operações (métodos, comportamentos), possíveis de serem executadas

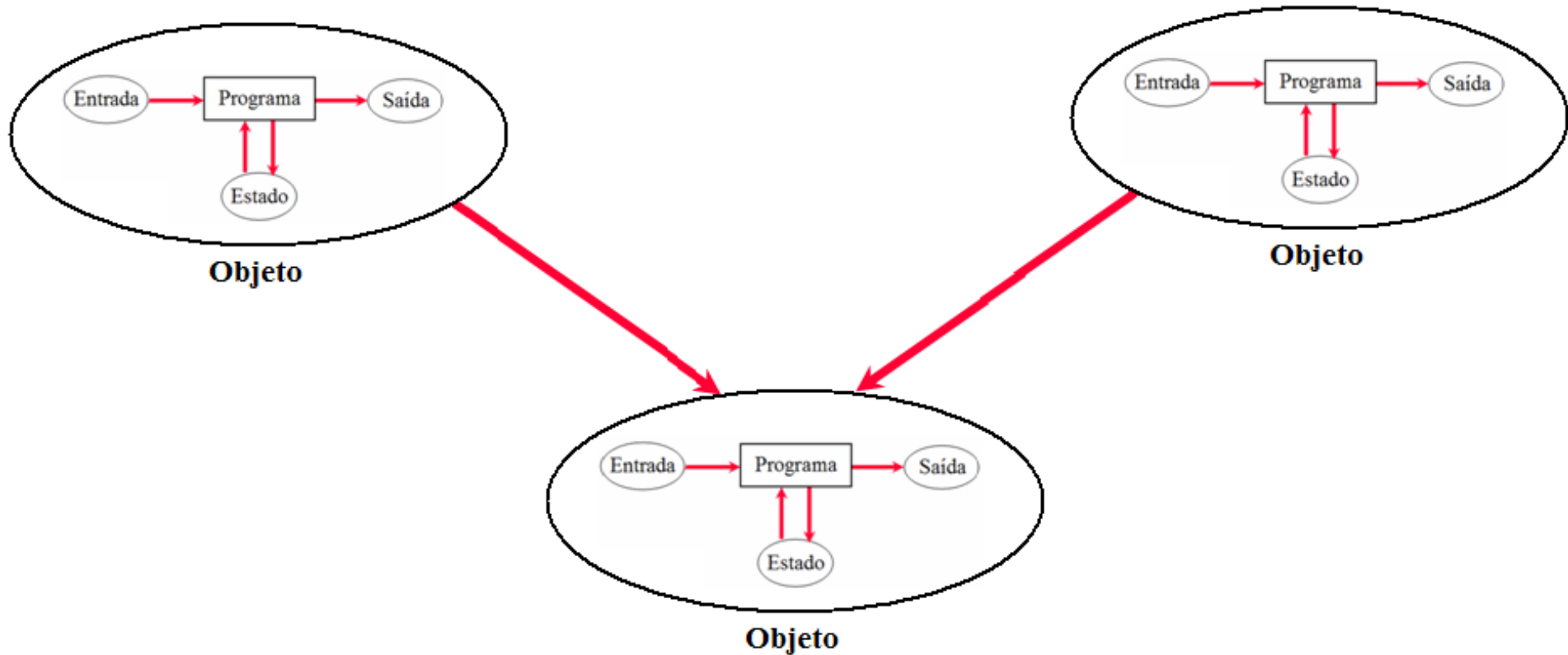
Exemplos: Java, C++, Simula, Smaltalk, C#





# Paradigma de Programação

## Paradigma Orientada a Objetos:



# Paradigma de Programação

## Paradigma Funcional:

- Linguagem em que o programa é construído por funções.
- O relacionamento entre funções é muito simples:
  - Uma função pode chamar outra função, ou
  - O resultado de uma função pode ser usado como um argumento de outra função.
- Execução do programa = avaliar funções/expressões.

Exemplos: LISP, Scheme, Haskell



# Paradigma de Programação

## Paradigma Funcional:

Exemplo:

(plus 3 5) : retorna 8.

(plus 1 2 3 4 5 6 7 8) : retorna 36.

(plus 1 2 3 4 5 6 7 8 -36) : retorna 0

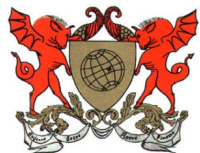


# Paradigma de Programação

## Paradigma Lógico:

- Manipulação de memória é automática
- Usa lógica de predicados como linguagem de programação.
- Um programa em lógica é formado por fatos, regras e consultas sobre o mundo real.

Exemplo: Prolog



# Paradigma de Programação

## Paradigma Lógico:

### Fatos:

gosta(joao, peixe)

gosta(joao, maria)

gosta(maria, livro)

gosta(pedro, livro)

### Perguntas feitas ao Prolog:

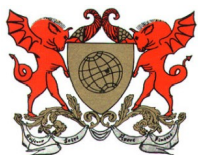
?- gosta(joao, dinheiro).

no

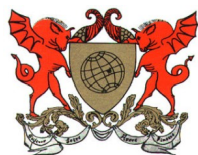
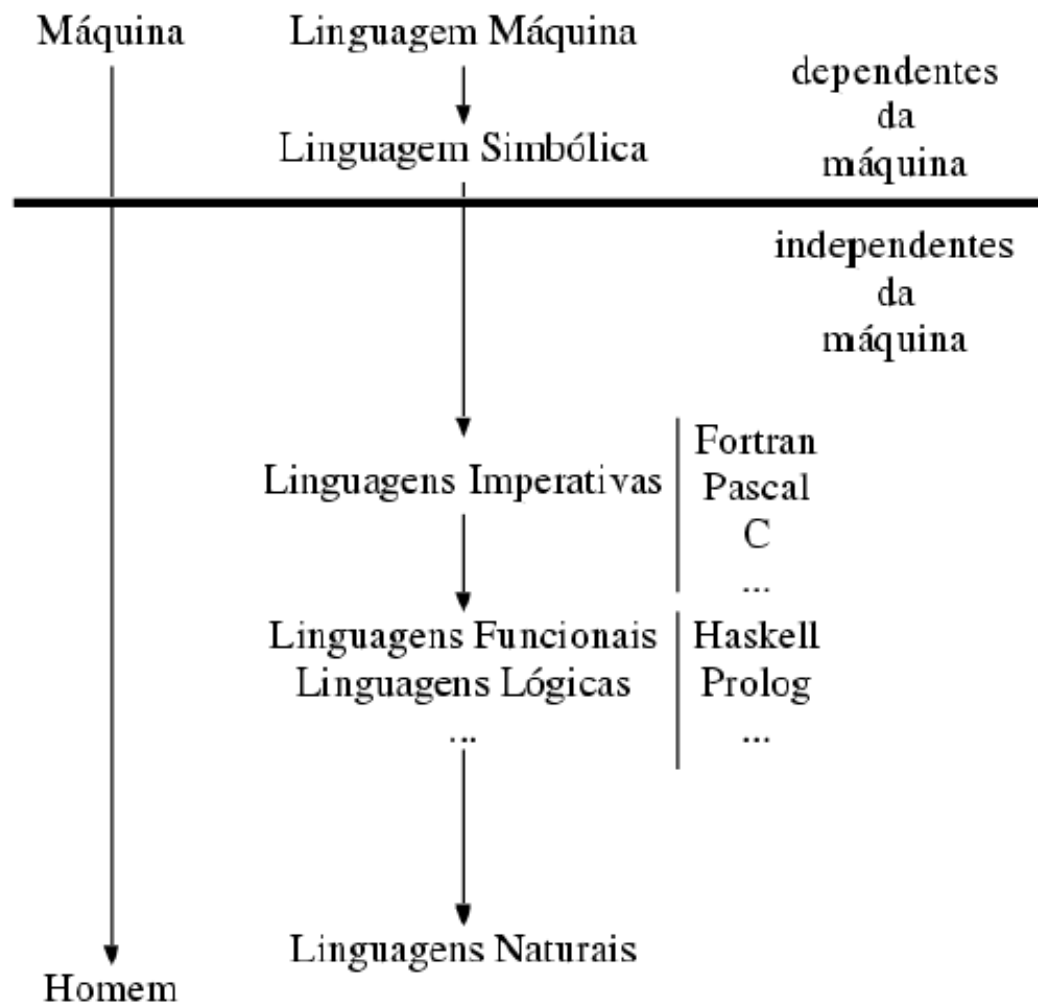
?- gosta(maria, livro).

yes

“no” significa “não foi possível provar”



# Comunicação com o Computador

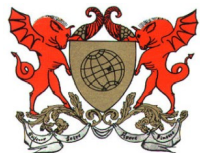


# Análise de Linguagens

Os programas são a forma de se comunicar com um computador (linguagem de máquina)

As linguagens de programação são avaliadas pelo computador de duas formas:

- Compilação (Linguagens Compiladas)
- Interpretação (Linguagens Interpretadas)



# Compilador

Processo:

código fonte → código objeto → executável

Código fonte: é o programa em si, escrito pelo programador, contendo os comandos da linguagem

Código objeto: é a tradução do código fonte para uma forma que o computador possa executar

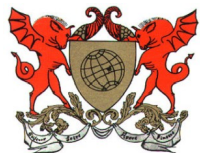


# Compilador

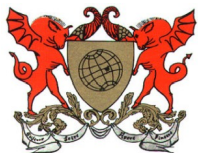
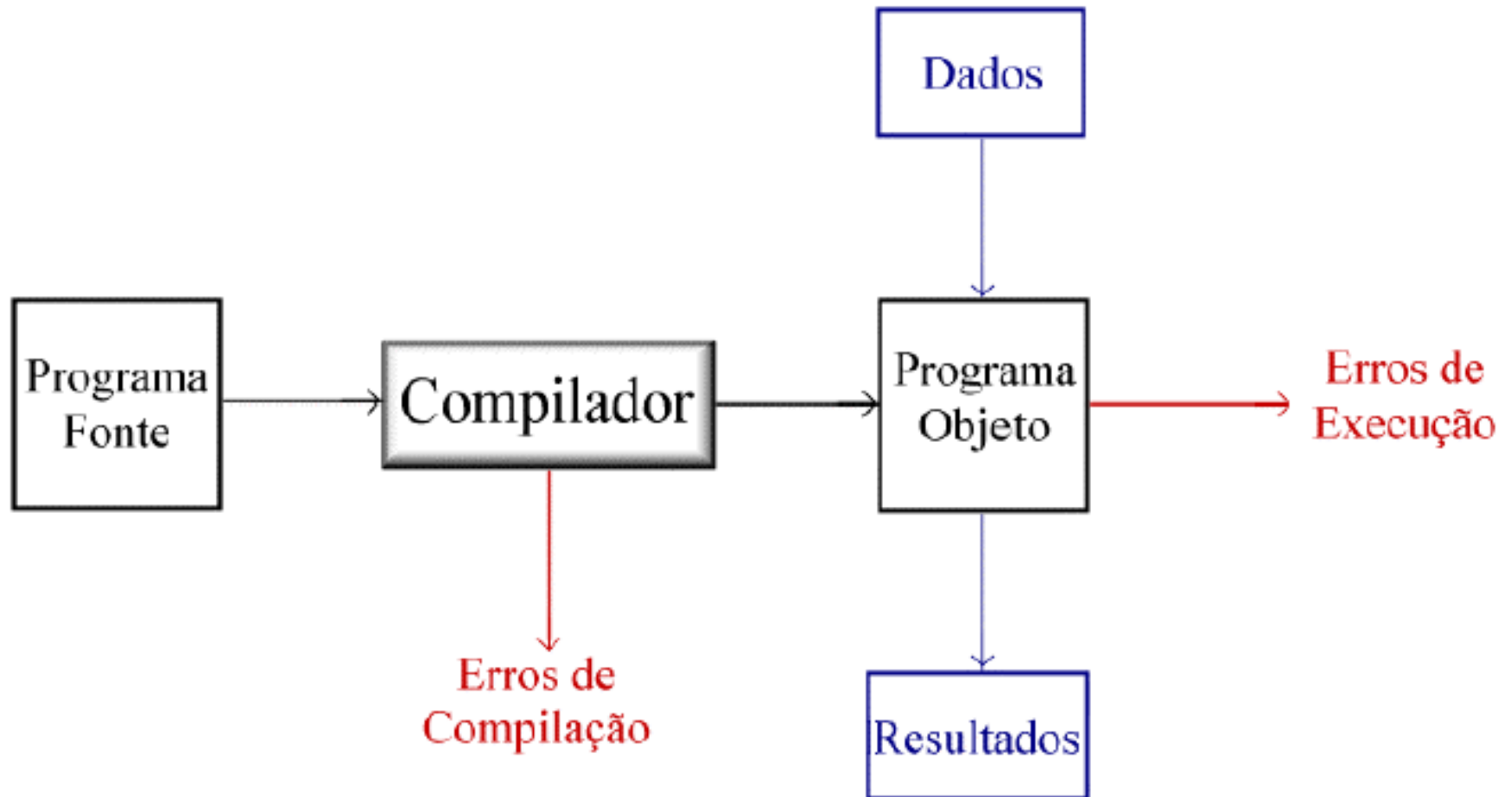
São responsáveis por converter um código escrito em uma linguagem de alto nível (ex.: C++, C, Java, Pascal etc.) em um código binário executável (instruções mais simples, que a CPU entende)

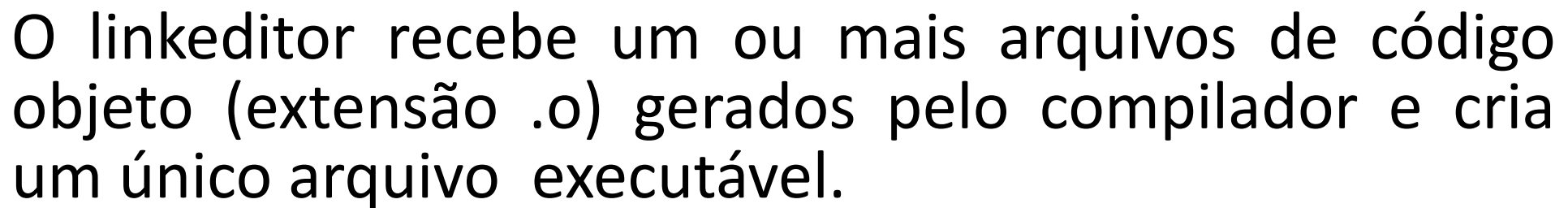
São específicos para a linguagem para o qual foram projetados para compilar e também para o SO e o hardware onde estão sendo executados.

Portanto, para uma mesma linguagem, existem compiladores diferentes, em SOs diferentes.



# Compilador

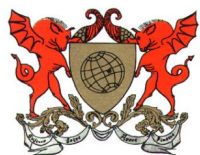




# Interpretador

É um programa que interpreta **diretamente as instruções** do programa fonte, gerando o resultado.

Toda vez que o programa é executado, o interpretador lê e executa uma linha por linha. O interpretador não gera código objeto.



# Interpretador Java

**javac Teste.java**

código fonte

Teste.java

compilador

javac

arquivo byte-code

Teste.class

**java Teste**

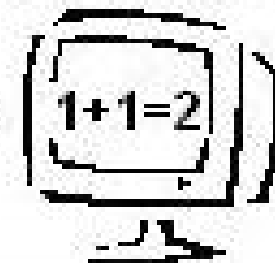
byte-code

Teste.class

interpretador (JVM)

java

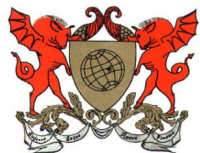
resultado



# Compilação x Interpretação

## Compilação

- O programa fonte não é executado diretamente
- O programa fonte é convertido em programa objeto e depois é executado
- A execução não depende do código fonte, só do executável
- Vantagem: Execução mais rápida



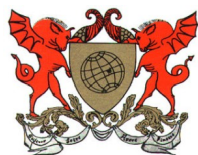
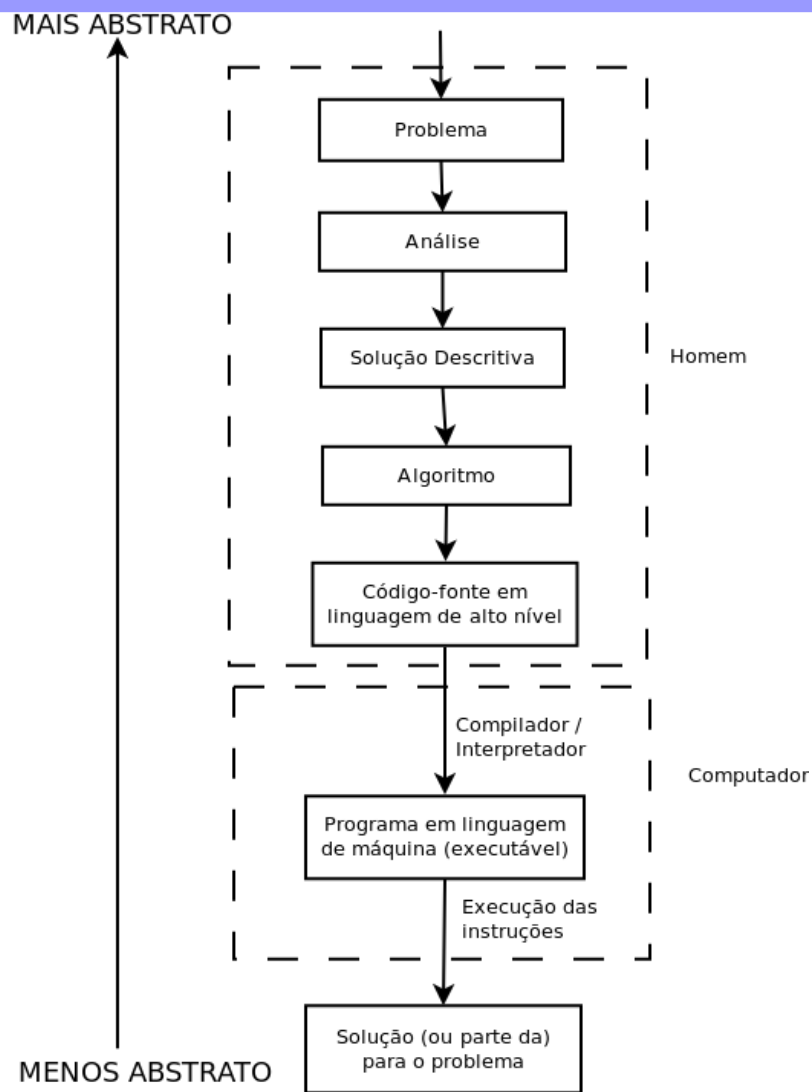
# Compilação x Interpretação

## Interpretação

- Interpreta e executa o programa fonte diretamente
- Não existe a geração de um módulo ou código-objeto
- A execução depende do código fonte presente.
- Vantagem: Programas menores e mais flexíveis



# Fluxo de Resolução de Problemas





# Video (Linguagem de Programação)

O que a maioria das escolas não ensinam aos bilionários que você conhece:

<https://www.youtube.com/watch?v=ZRtPQfOc8jg>

