

Linguagens de Programação

Paradigmas de Linguagens de Programação

Samuel da Silva Feitosa

Aula 3

A stack of colorful blocks representing various programming languages. The blocks are arranged in a pyramid-like structure. The languages visible on the blocks include JavaScript (green), Ruby (red), C# (blue), Java (yellow), T-SQL (orange), PL-SQL (light blue), C (purple), Swift (pink), and Python (dark red). The blocks are stacked on a surface that appears to be a keyboard.

Classes de Linguagens de Programação

Classificação das LPs

- Quanto ao grau de abstração
 - Baixo ou alto nível.
- Quanto ao paradigma
 - Imperativo, Orientado a objetos, Funcional e Lógico.
- Quanto a estrutura de tipos
 - Fortemente ou fracamente tipada.
 - Dinâmica ou estaticamente tipada.



Grau de Abstração

ULLAMCO LABORIS NISI UT
ALIBUI EX EA COMMOD
CONSEQUAT. DUIS AUT

A4 SIZE

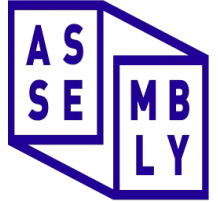
Grau de abstração

- **Baixo nível**

- Relacionada com a proximidade do hardware e seu código de máquina.
- Projetada pensando mais no acesso e controle ao hardware do que na facilidade de uso para o usuário.

- **Alto nível**

- Projetada com foco no usuário.
- Foco na facilidade de escrita, legibilidade do código, capacidade de abstração e recursos.



LP Compilada ou Interpretada

- **Linguagem compilada**

- Programas são traduzidos para código de máquina e depois executados diretamente pelo computador.

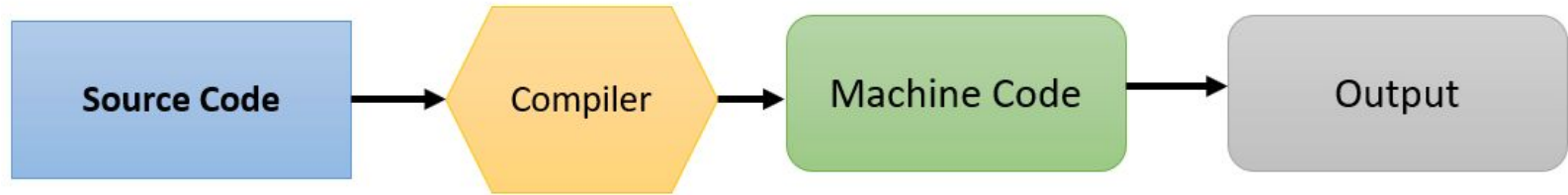
- **Linguagem interpretada**

- Os programas são interpretados diretamente por outro software, chamado de interpretador.

- **Linguagem híbrida**

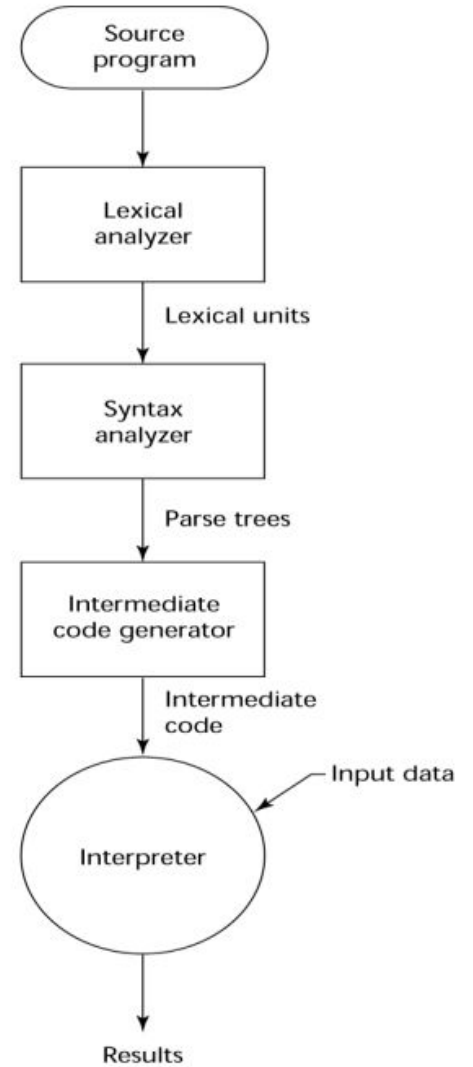
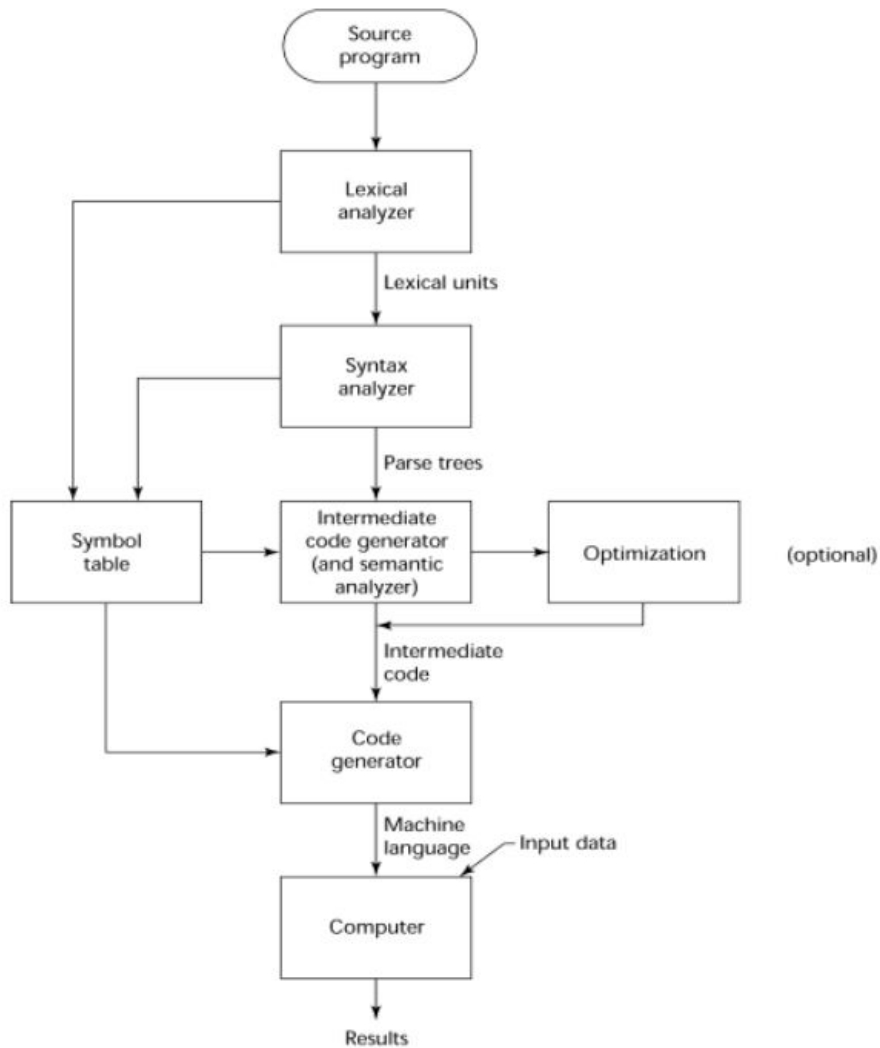
- Traduz os programas em linguagem de alto nível para uma linguagem intermediária projetada para facilitar a interpretação.

How Compiler Works



How Interpreter Works





A hand is shown interacting with a futuristic, glowing digital interface. The interface consists of concentric circles, glowing lines, and various data points. The background is dark blue with some hexagonal patterns and numbers like 60, 40, 80, and 2. The text "Diferentes Paradigmas" is overlaid on a red rectangular box in the upper right quadrant.

Diferentes Paradigmas

Definição de LP

Uma LP é uma linguagem destinada a ser usada por uma **pessoa** para expressar um **processo** através do qual um **computador** pode resolver um **problema**.

- Os quatro modelos (paradigmas) de LP correspondem aos pontos de vista dos quatro componentes citados.

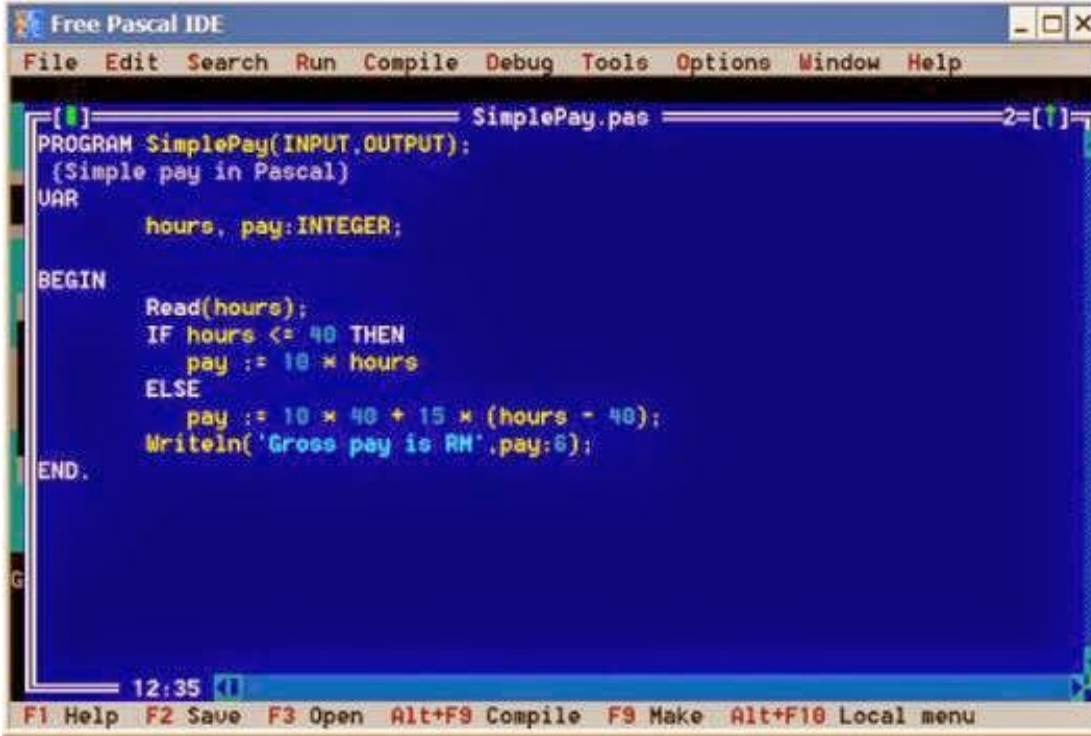
O que é “paradigma”?

- **Definição do dicionário:**
 - Algo que serve de exemplo geral ou de modelo;
 - Modelo que serve de padrão;
 - Norma já estabelecida;
- **Paradigma de programação** é um meio de se classificar as **linguagens de programação** baseado em suas funcionalidades.

Linguagens Imperativas

- Principais características
 - Variáveis, atribuição e iteração.
 - Exemplos: C, Pascal.
- As linguagens imperativas são orientadas a **ações**, onde a computação é vista como uma sequência de **instruções** que manipulam valores de **variáveis**.
- Programas centrados no conceito de um **estado** (variáveis) e **ações** (comandos) que manipulam o estado.

Exemplo de código imperativo

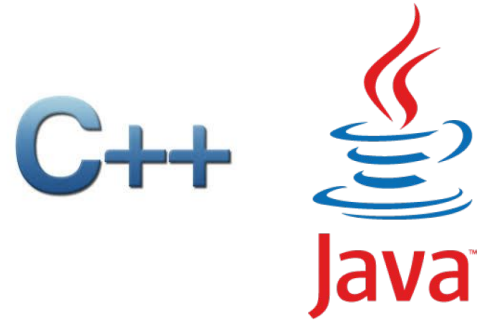


The image shows a screenshot of the Free Pascal IDE window. The title bar reads "Free Pascal IDE". The menu bar includes "File", "Edit", "Search", "Run", "Compile", "Debug", "Tools", "Options", "Window", and "Help". The main editor area displays a Pascal program named "SimplePay.pas". The code is as follows:

```
[1] SimplePay.pas 2=[1]
PROGRAM SimplePay(INPUT,OUTPUT);
{Simple pay in Pascal}
VAR
    hours, pay:INTEGER;
BEGIN
    Read(hours);
    IF hours <= 40 THEN
        pay := 10 * hours
    ELSE
        pay := 10 * 40 + 15 * (hours - 40);
    WriteIn('Gross pay is RM',pay:8);
END.
```

The status bar at the bottom shows the time "12:35" and a series of function key shortcuts: "F1 Help", "F2 Save", "F3 Open", "Alt+F9 Compile", "F9 Make", and "Alt+F10 Local menu".

Linguagens Orientadas a Objetos



- Principais características
 - Abstração de dados, objeto, mensagem, herança.
 - Exemplos: Java, C++, Perl, Python.
- Tratam os elementos e conceitos associados ao problema como objetos.
- Objetos são entidades abstratas que embutem dentro de suas fronteiras as características e operações relacionadas com a entidade real.
- Uma aplicação é estruturada em classes que agrupam estados e operações.

Exemplo de código OO

C++

```
1 #include "conio.h"
2 #include "iostream"
3 #include "string"
4 using namespace std;
5
6 class Person
7 {
8     public:
9         string name;
10        int age;
11        void speak(); //member function
12 };
13 void Person::speak() //member function
14 {
15     cout<<"Name= "<<name<<endl;
16     cout<<"Age= "<<age<<endl;
17 }
18 }
```

Java™

```
/**
 *
 * @author Mike
 */
public class Person
{
    private String name;

    /** Creates a new instance of Person */
    public Person(String n)
    {
        name = n;
    }

    //displays the name
    public void displayName()
    {
        System.out.println(name);
    }
}
```

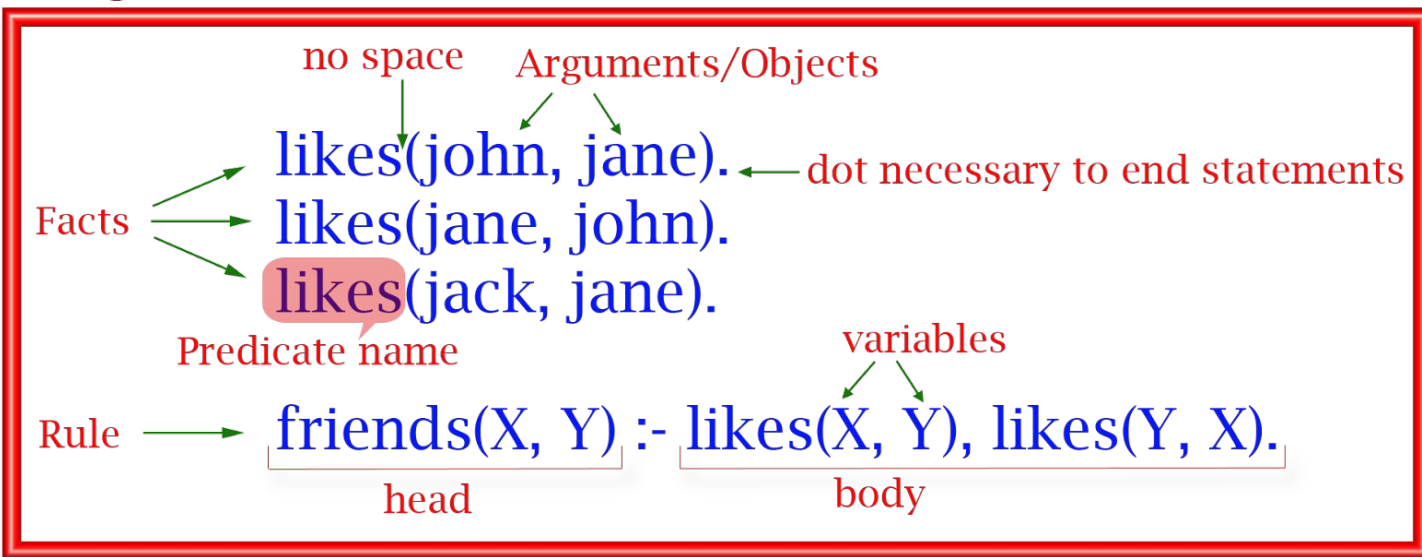
Linguagens Lógicas



- Principais características
 - Relação e dedução, baseado em regras.
 - Exemplo: Prolog
- Programação é baseada em fatos, que podem ser relações (associações) entre coisas, e regras, que produzem fatos deduzidos a partir de outros.
- Foi muito usado em sistemas especialistas (IA) e também em bancos de dados.

Exemplo de código Prolog

Program Window



Linguagens Funcionais



- Principais características
 - Função, aplicação e avaliação.
 - Exemplos: Lisp, Scheme, Haskell.
- Sistemas são construídos através da **definição, composição e aplicação** de funções.
- **Estilo declarativo:** não há o conceito de estado nem comandos como atribuição.
- Conceitos sofisticados como polimorfismo, funções de alta ordem e avaliação sob demanda.

Exemplo de código funcional

LISP

```
(define (reduce f a x y b fx fy)
  (cond ((close-enough? a b) x)
        ((> fx fy)
         (let ((new (x-point a y)))
           (reduce f a new x y (f new) fx)))
        (else
         (let ((new (y-point x b)))
           (reduce f x y new b fy (f new))))))
```

Haskell

```
quicksort1 :: (Ord a) => [a] -> [a]
quicksort1 [] = []
quicksort1 (x:xs) =
  let smallerSorted = quicksort1 [a | a <- xs, a <= x]
      biggerSorted = quicksort1 [a | a <- xs, a > x]
  in  smallerSorted ++ [x] ++ biggerSorted
```

The background of the slide is a dark, textured surface covered with various green, handwritten-style characters and symbols, including letters, numbers, and mathematical notations. A purple pen is positioned diagonally across the lower half of the image, pointing towards the bottom right. A solid dark red rectangular box is located in the upper right quadrant, containing the title text.

Estrutura de tipos

Estrutura de tipos

- Fracamente tipada
 - Tipo da variável muda dinamicamente conforme a situação. Exemplos: PHP, Smalltalk.
- Fortemente tipada
 - Uma vez atribuído o tipo, se mantém o mesmo até ser descartado. Exemplos: Java, Ruby, Python.
- Dinamicamente tipada
 - Tipo da variável definido em tempo de execução. Exemplos: Perl, Python, Javascript.
- Estaticamente tipada
 - Tipo da variável definido em tempo de compilação. Exemplos: Java, C, C++.

Comentários Finais

- Definição de paradigma e paradigmas de programação.
- Classificações das linguagens de programação
 - Quanto ao grau de abstração, paradigma, e estruturas de tipos.
- Processo de compilação e interpretação
- Visão geral de linguagens em diferentes paradigmas