INE5622 – INTRODUÇÃO A COMPILADORES

- **PLANO DE ENSINO**
  - **Objetivo geral**
    - **Conhecer o processo de especificação e implementação de linguagens de programação, a partir do estudo dos conceitos, modelos, técnicas e ferramentas que compõem a Teoria das Linguagens Formais e a Teoria de Compiladores.**
  - **Objetivos específicos**
    - **Adquirir uma visão geral sobre o Processo de Compilação sob o ponto de vista de implementação.**
    - **Adquirir noções básicas sobre a Teoria das Linguagens Formais.**
    - **Saber especificar aspectos léxicos e sintáticos de linguagens através de autômatos e gramáticas.**
    - **Conhecer critérios e características usados no projeto/avaliação de Linguagens de Programação.**
    - **Conhecer as principais técnicas e ferramentas de apoio usadas na construção de compiladores, sabendo usá-las na especificação e implementação de linguagens de programação.**
    - **Obter Subsídios que permitam um melhor entendimento, utilização e avaliação das Linguagens de Programação.**
  - **Programa**
  - **Avaliação**
  - **Bibliografia**

- **MOTIVAÇÃO**
  - **Linguagens Formais e Autômatos**
    - **Necessidade de uma visão geral dos Fundamentos Teóricos da Computação**
      - **Contato com Modelos e Técnicas Formais usadas na especificação/implementação de Linguagens**
    - **Capacidade para correlacionar aspectos teóricos e práticos da Computação**
      - **Base para a melhoria no entendimento, no uso e na produção de software (básico e aplicativo)**
  - **Linguagens de Programação e Compiladores**
    - **Melhorar o entendimento de L.P.**
      - **aspectos conceituais x implementações**
      - **escolha e uso mais racional/eficiente**
      - **facilitar o aprendizado de novas linguagens**
    - **Capacidade para Especificar/Implementar Linguagens**
      - **Uso geral**
        - **novas, extensões, atualizações**
      - **Uso específico**
        - **Tempo Real, Robótica, Descriçao de Hardware**
        - **S. O., B.D., Protocolos, Redes, Ling. Naturais**
    - **Uso dos modelos/ técnicas em outros sistemas**
      - **Proc. de textos, Reconh. de padrões, SI em geral**
    - **Ponto de partida para estudos avançados**

I.1 - Introdução a Compiladores

I.1.1 - Definições preliminares

Tradutor

- É um programa que traduz um programa fonte escrito em uma linguagem qualquer (denominada linguagem fonte) para um **programa objeto equivalente** escrito em outra linguagem (denominada linguagem objeto)

Compilador

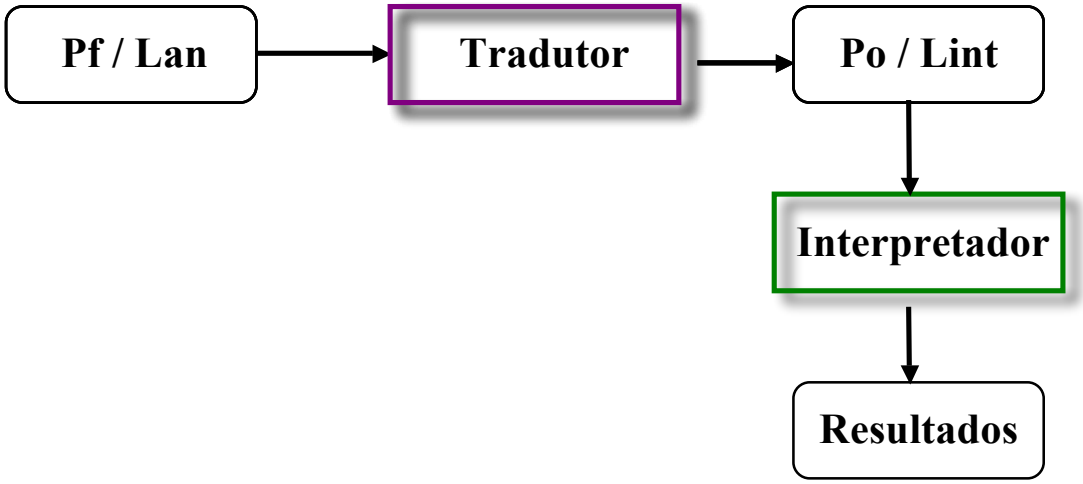
- É um Tradutor em que a linguagem fonte é uma linguagem de alto nível e a linguagem objeto é uma linguagem de baixo nível (assembly ou máquina)

Interpretador

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

Tradutor / Interpretador

- Esquema híbrido para implementação de linguagens de programação



Montador

- É um Tradutor em que o programa fonte está escrito em linguagem assembly e o programa objeto resultante está em linguagem de máquina

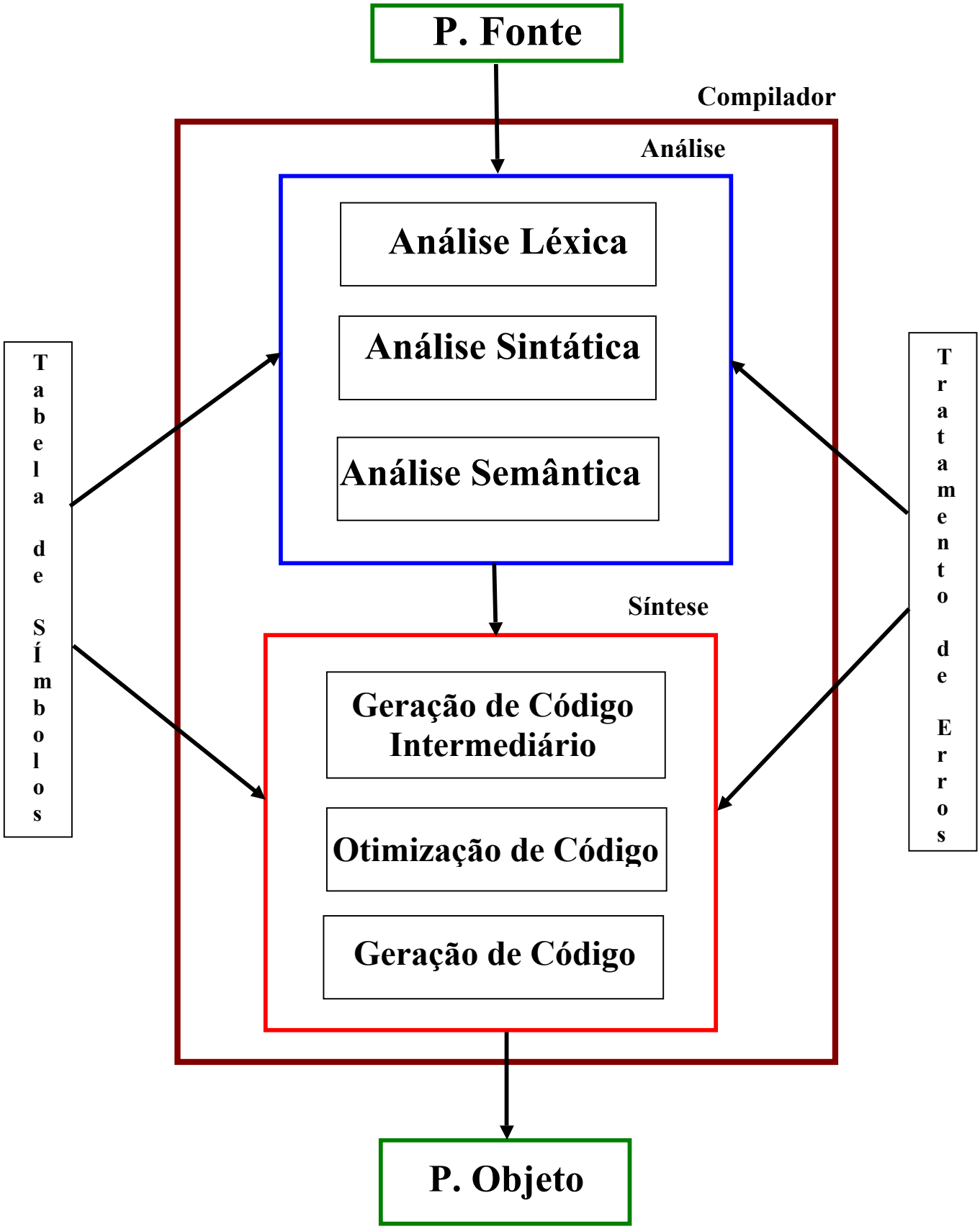
Pré-processador

- É um Tradutor em que tanto o programa fonte quanto o programa objeto estão escritos em linguagens de alto nível

Cross - Compiler

- Compilador que gera código para uma máquina diferente da utilizada na compilação.

I.1.2 - Estrutura geral de um Compilador (Modelo Análise - Síntese)



Modelo Front-end – Back-end

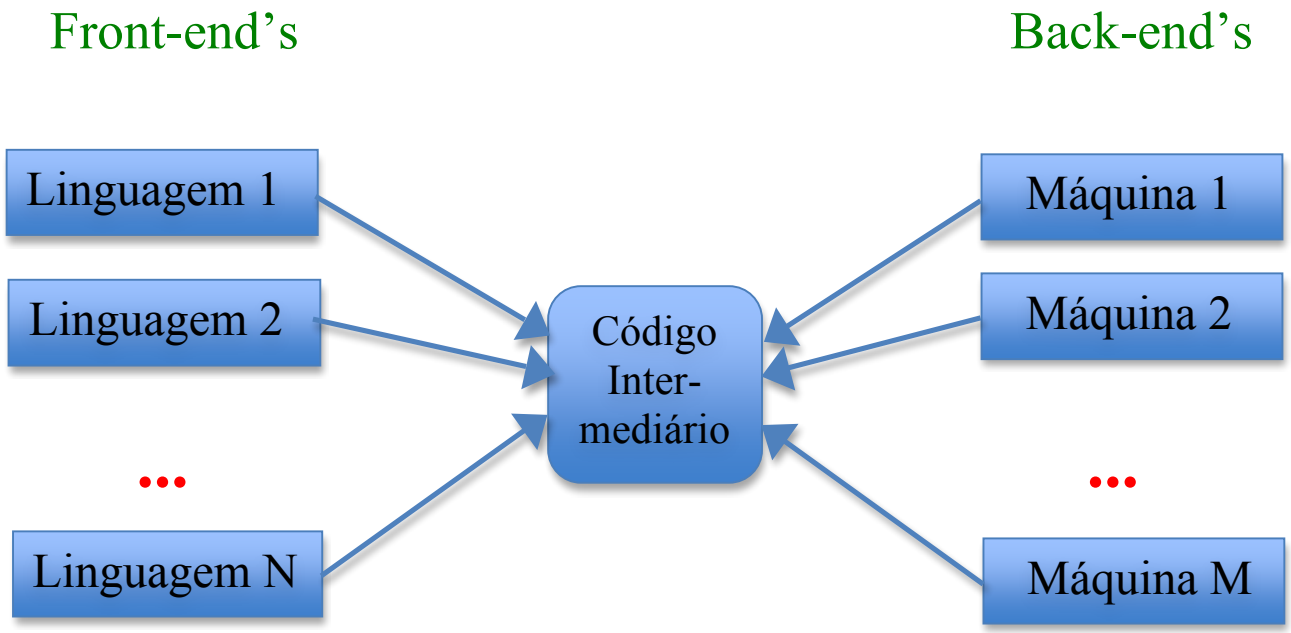
Front-end

- Independente de máquina
- Compreende:
  - Análise (Léxica, Sintática e Semântica)
  - Geração de Código Intermediário

Back-end

- Independente de linguagem
- Compreende:
  - Otimização de Código
  - Geração de Código

Infraestrutura de desenvolvimento de Compiladores



I.1.3 - Formas de Implementação de Compiladores

Fase - Procedimento que realiza uma função bem definida no processo de compilação.

Passo - Passagem completa do programa compilador sobre o programa fonte que está sendo compilado.

Formas de Implementação

- **Compiladores de 1 passo**
  - **Todas as funcionalidades (fases) são executadas simultaneamente**
- **Compiladores de vários passos**
  - **Diferentes composições (agrupamento de fases)**
  - **Exemplos :**
- **Critérios para escolha**
  - Memória disponível
  - Tempo de Compilação
  - Tempo de execução
  - Características da Linguagem
    - **Referências futuras**
  - Características das Aplicações
    - **Necessidades de Otimização**
  - Tamanho / Experiência da Equipe
  - Disponibilidade de Ferramentas de Apoio
  - Prazo para desenvolvimento
- **Vantagens X Desvantagens**

I.1.4 - Fases de um Compilador

I.1.4.1 - Analisador Léxico

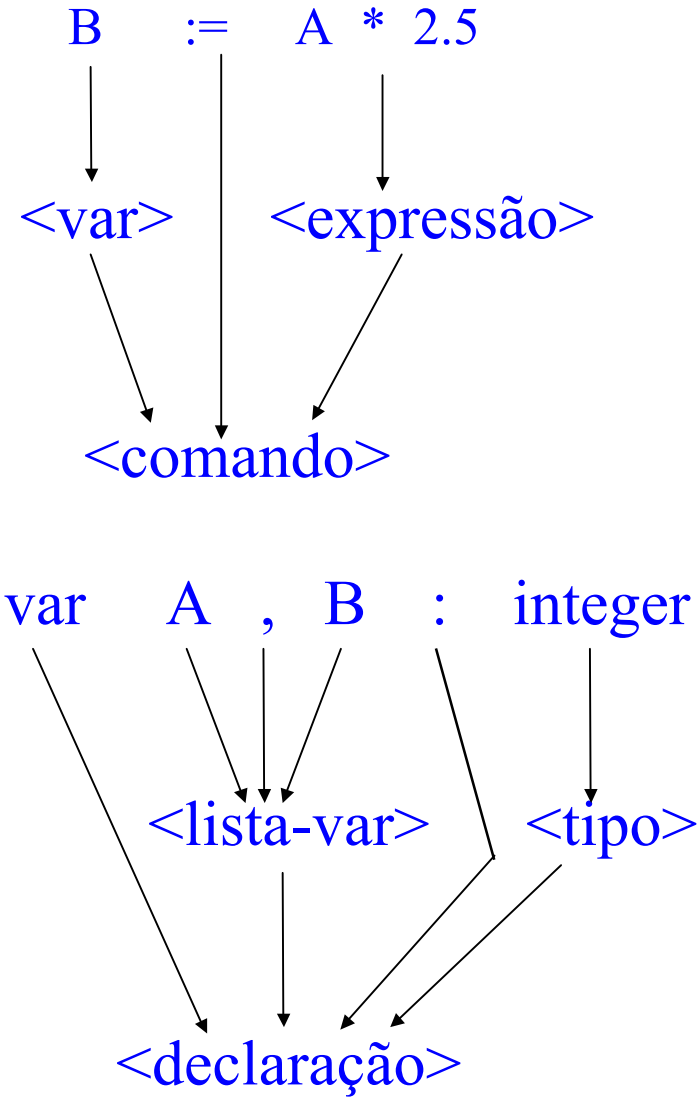
- Interface entre o programa fonte e o compilador
- Funções básicas:
  - Ler o programa fonte
  - Agrupar caracteres em itens léxicos (tokens)
    - Identificadores
    - Palavras Reservadas
    - Constantes (numéricas e literais)
    - Símbolos especiais (simples, duplos, ...)
  - Ignorar elementos sem valor sintático
    - Espaços em brancos, comentários e caracteres de controle
  - Detectar e diagnosticar erros léxicos
    - Símbolos inválidos, elementos mal formados
- Exemplo:

Programa Fonte	Tokens Reconhecidos	
program exemplo; var A, B : integer; begin (* Inicio do programa *) read (A); B := A + 2.5; . . . end.	program exemplo ; var A , ... end .	1 - PR 2 - ID 3 - SE 4 - PR 2 - ID 5 - SE ... 37 - PR 38 - SE

I.1.4.2 - Analisador Sintático

- Funções básicas
  - Agrupar TOKENS em estruturas sintáticas (expressões, comandos, declarações, etc. ...)
  - Verificar se a sintaxe da linguagem na qual o programa foi escrito está sendo respeitada
  - Detectar/Diagnosticar erros sintáticos

- Exemplos:



I.1.4.3 - Analisador Semântico

SEMÂNTICA ≅ COERÊNCIA ≅ SIGNIFICADO ≅ SENTIDO LÓGICO

- Funções básicas:
  - Verificar se as construções utilizadas no P.F. estão semanticamente corretas
  - Detectar e diagnosticar erros semânticos
  - Extrair informações do programa fonte que permitam a geração de código

- Verificações Semânticas Usuais
  - Análise de escopo
    - Declaração de variáveis
      - Obrigatória?
      - Múltiplas declarações permitidas?
  - Compatibilidade de tipos
  - Coerência entre declaração e uso de identificadores
  - Correlação entre parâmetros formais e atuais
  - Referências não resolvidas
    - Procedimentos e desvios

Tabela de Símbolos :

Definição - Estrutura onde são guardadas as informações (os atributos) essenciais sobre cada identificador utilizado no programa fonte.

Atributos mais comuns

- nome
- endereço relativo (nível e deslocamento)
- categoria
  - variável
    - simples - tipo
    - array - dimensões, tipo dos elementos
    - record - campos (quant. e apontadores)
    - ...
  - constante
    - tipo e valor
  - procedimentos
    - procedure ou função
    - número de parâmetros
    - ponteiro para parâmetros
    - se função, tipo do resultado
  - parâmetro
    - tipo
    - forma de passagem (valor , referência)
  - campo de record
    - tipo, deslocamento dentro do Record

Tratamento ou Recuperação de ERROS:

- **Funções**
  - Diagnosticar erros léxicos, sintáticos e semânticos encontrados na etapa de análise
  - Tratar os erros encontrados, permitindo que o compilador **recupere-se** da situação de erro e possa concluir a análise.

I.1.4.4 - Gerador de Código Intermediário

- Função
  - Consiste na geração de um conjunto de instruções (equivalentes ao programa fonte de entrada) para uma máquina hipotética (virtual)
- Exemplo

E := (A + B) * (C + D)

Quadrupla	Máquina de acumulador
(+ , A , B , T1) (+ , C , D , T2) (+ , T1 , T2 , E)	carregue A some B armazene T1 carregue C some D armazene T2 carregue T1 multiplique T2 armazene E

I.1.4.5 - Otimizador de código

- Função
  - Melhorar o código, de forma que a execução seja mais eficiente quanto ao tempo e/ou espaço ocupado

- Otimizações mais comuns
  - Agrupamento de sub-expressões comuns

    ex. c := (a + b) * (a + b)
  - Eliminação de desvios para a próxima instrução
  - Retirada de comandos invariantes ao LOOP
  - Eliminação de código inalcançável
  - Redução em força
  - Transformação/avaliação parcial
  - Alocação ótima de registradores

I.1.4.6 - Gerador de Código

- Função :
  - Converter o programa fonte (diretamente ou a partir de sua representação na forma de código intermediário) para uma sequência de instruções (assembler ou máquina) de uma máquina real.

I.1.5 - Planejamento da Construção de um Compilador

- **Por que é necessário construir um novo compilador ?**
  - **Criação de uma nova Linguagem**
  - **Extensão de uma linguagem existente**
  - **Surgimento de uma nova máquina**
  - **Desempenho do compilador existente**

- **Definição Preliminar da Ling. fonte**
  - **Objetivos**
    - **Propósito geral ou específico**
    - **Comercial ou experimental**
  - **Filosofia de Programação**
    - **Imperativa (Estruturada/Objetos)**
    - **Funcional, Lógica**
    - **Mista (Multi - Paradigma)**
  - **Potencialidade(s) Básica(s)**
    - **Sistema de Tipos**
    - **Concorrência (Multi-Thread)**
    - **Distribuição, Facilidades de B. D.**
    - **Abstração Funcional**

- **Definição preliminar da ling. objeto**
  - **Nível**
    - **Alto nível, Intermediária**
    - **Assembly, Máquina**
  - **Máquina Alvo**
    - **Real ou Hipotética (Virtual)**

- **Definição do Tipo de Tradutor**
  - **Compilador**
  - **Interpretador**
  - **Tradutor/Interpretador**
  - **Pré-Processador**
  - **Montador**

- **Estrutura do Tradutor**
  - **Número de fases**
  - **Número de passos**
  - **Forma de Integração**
  - **Linguagens Intermediárias**

- **Definição do Ambiente Operacional**
  - **Hardware e sistema Operacional**
  - **Linguagem de Implementação**
  - **Disponibilidade de Ferramentas**

- **Especificação da Linguagem Fonte**
  - **Completa e Detalhada**
    - **Aspectos Léxicos**
    - **Aspectos Sintáticos**
    - **Aspectos Semânticos**

- **Especificação da Linguagem Objeto**
  - **Completa e Detalhada**
    - **Arquitetura da Máquina Alvo**
    - **Repertório de Instruções**
    - **Procedimentos de Tradução**

I.2–Introdução à Teoria das Linguagens Formais

Teoria da Computação

O que é ?

- Fundamento da Ciência da Computação
- Tratamento Matemático da Ciência da Computação
- Estudo Matemático da Transformação da Informação
- Guia : "Que problemas podem ser efetivamente computáveis, como e com que complexidade"

Qual sua importância?

Classificação dos problemas:

- **Não-Computáveis**
- **Computáveis**
  - **Indecidíveis**
  - **Decidíveis**
    - **Intratáveis**
    - **Tratáveis**

Exemplos de problemas:

- Computabilidade / Decidibilidade
- Equivalência entre Programas
- Garantia de parada de um programa
- Complexidade de Algoritmos
- Significado e Correção de Programas

Teoria da Computação
X
Teoria das Linguagens Formais

Definição de Teoria da Computação sob a ótica da Teoria das Linguagens Formais:

Conjunto de Modelos Formais (autômatos e gramáticas, p. ex.), que juntamente com suas propriedades (decidibilidade, equivalência e complexidade), fundamentam a Ciência da Computação.

Conceitos e Propósitos Fundamentais da Teoria da Computação

PROCEDURE X ALGORITMO

Procedure: **Sequência finita de passos, executáveis mecanicamente de forma discreta.**

Algoritmo : **É uma procedure que, independentemente das entradas, possui parada garantida.**

Exemplos:

- Determinar se I ≥ 1 é um número primo
- Determinar se existe um número perfeito > I
- Determinar se um programa está sintaticamente correto
- Determinar se um programa qualquer entrará em loop para uma entrada qualquer (Halting Problem)

Conj. Recursivos e Recursivamente Enumeráveis

X

Algoritmos e Procedures

X

Problemas Decidíveis e Problemas Indecidíveis

Propósitos da Teoria da Computação

(ou : Modelos que formalizam a noção do que é do que não é efetivamente computável)

Máquinas de Turing (Turing, 1936)

→ Tese de CHURCH – Todo processo efetivo pode ser realizado por uma máquina de turing)

Gramáticas (Chomsky, 1959)

→ Tipos 0, 1, 2 e 3

Algoritmos de Markov (Markov, 1951)

→ Sistemas de regras de produção

→ Processamento de strings

λ-Calculus (Church, 1936)

→ Método para especificação de funções

→ Influenciou programação funcional

Sistemas de POST (Emil Post, 1936)

→ Formalização de sistemas de re-escrita

→ Lógica formal e sistemas especialistas

Funções Recursivas (Kleene, 1936)

→ Método para definição de funções a partir de um conjunto de equações

exemplo : $X^Y = 1$, se $y = 0$

$= X \cdot X^{Y-1}$, se $y > 0$

Teoria das Linguagens Formais

- **O que é LINGUAGEM FORMAL ?**
- **O que é LINGUAGEM ?**
  - Forma de comunicação
  - Conjunto de símbolos + conjunto de regras
  - Exemplos: L. Máquina, PASCAL, Português, ...

Conceitos Básicos

- Alfabeto :
  - Conjunto finito e não vazio de símbolos
- Sentença:
  - Sequência de símbolos de um alfabeto
- Tamanho de uma sentença:
  - Quantidade de símbolos de uma sentença
- Sentença vazia:
  - denotada por ε, é uma sentença de tamanho 0
- Potência de uma sentença:
  - exemplo: $a^3 = aaa$
- Fechamento de um alfabeto:
  - Reflexivo : V^*
  - Transitivo (ou Positivo) : V^+

Linguagens e suas Representações

- **Linguagem :**

$$L \subseteq V^*$$

- **Formas de Representação:**
  - Enumeração
  - Sistemas Geradores
  - Sistemas Reconhecedores
- **Linguagens Formais**
  - Dispositivos / Modelos matemáticos
- **Linguagens Recursivas:**
  - Algoritmos
- **Linguagens Recursivamente Enumeráveis**
  - Procedures
- **Teoria das Linguagens Formais:**
  - Estudo dos modelos matemáticos que possibilitam a especificação, o reconhecimento, a classificação, as propriedades e o interrelacionamento entre linguagens.

- **Importância da Teoria das Linguagens:**
  - Apóia aspectos básicos da Teoria da Computação:
    - Decidibilidade, Computabilidade e complex.
  - Fundamenta Aplicações Computacionais:
    - Processamento de Linguagens (esp. / impl.),
    - Rec.de Padrões, Modelagem de Sistemas, ...