UNIPAC - Universidade
Presidente Antônio Carlos
Departamento de Sistemas de Informação

Introdução a Programação Funcional

Wendell Pereira da Silva

Objetivos Gerais

Iniciar os estudos sobre o paradigma da programação funcional.

Construir uma base para estudos auto-motivados para disciplinas que exijam programação de computadores.

Objetivos Específicos

Formalizar conceitos primários da disciplina de programação de computadores.

Apresentar o paradigma da Programação Funcional, usando a linguagem *Haskell*, manipulando exemplos de programas elementares usando a aplicação HUGS 98 - um pequeno interpretador da linguagem Haskell.

Tópicos Abordados

Linguagem de Programação - Breve Introdução.

Programação Funcional - Introdução a Linguagem *Haskell* com o HUGS 98.

Linguagens de Programação

Principal Motivação:

A utilização da linguagem natural para comunicação entre seres humanos e computares e atualmente inviável devido a inúmeros fatores, dentre eles:

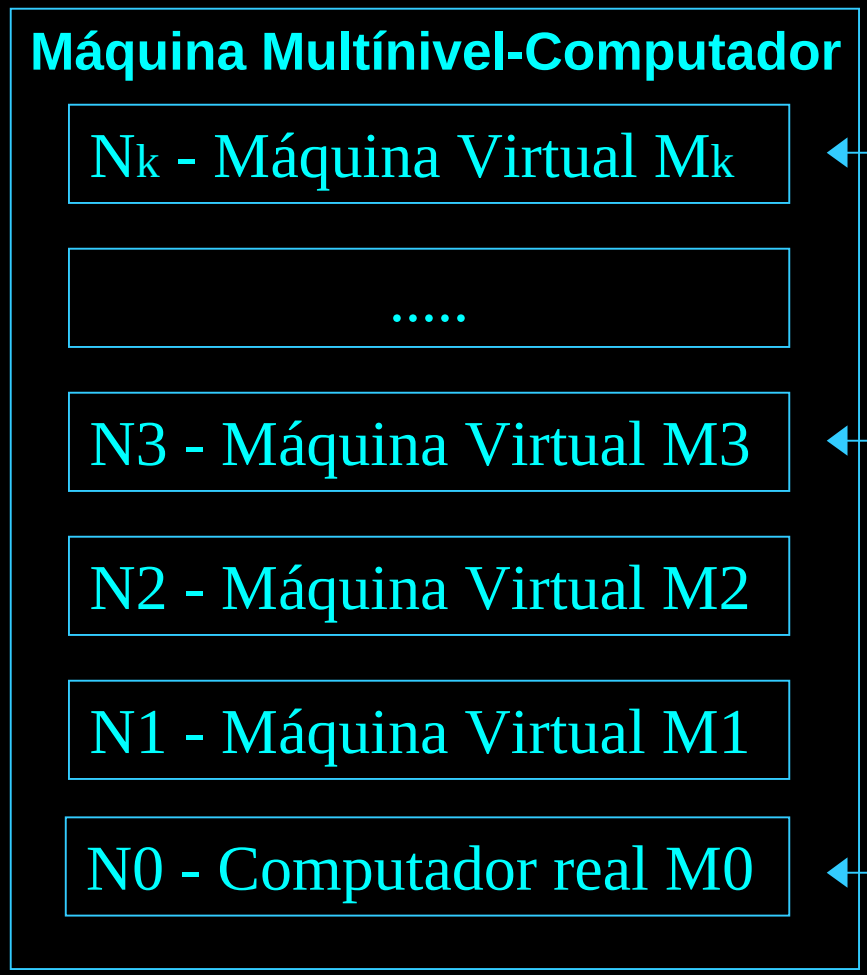
- Ambigüidades da linguagem natural.

- Alta complexidade de análise / processamento / resposta.

Principal Função:

*Oferecer uma **interface de comunicação** entre o **homem** e a **máquina (computador)**, estabelecendo as regras de formação de sentenças e os significados válidos.*

Linguagens de Programação Múltiplos Níveis de Atuação

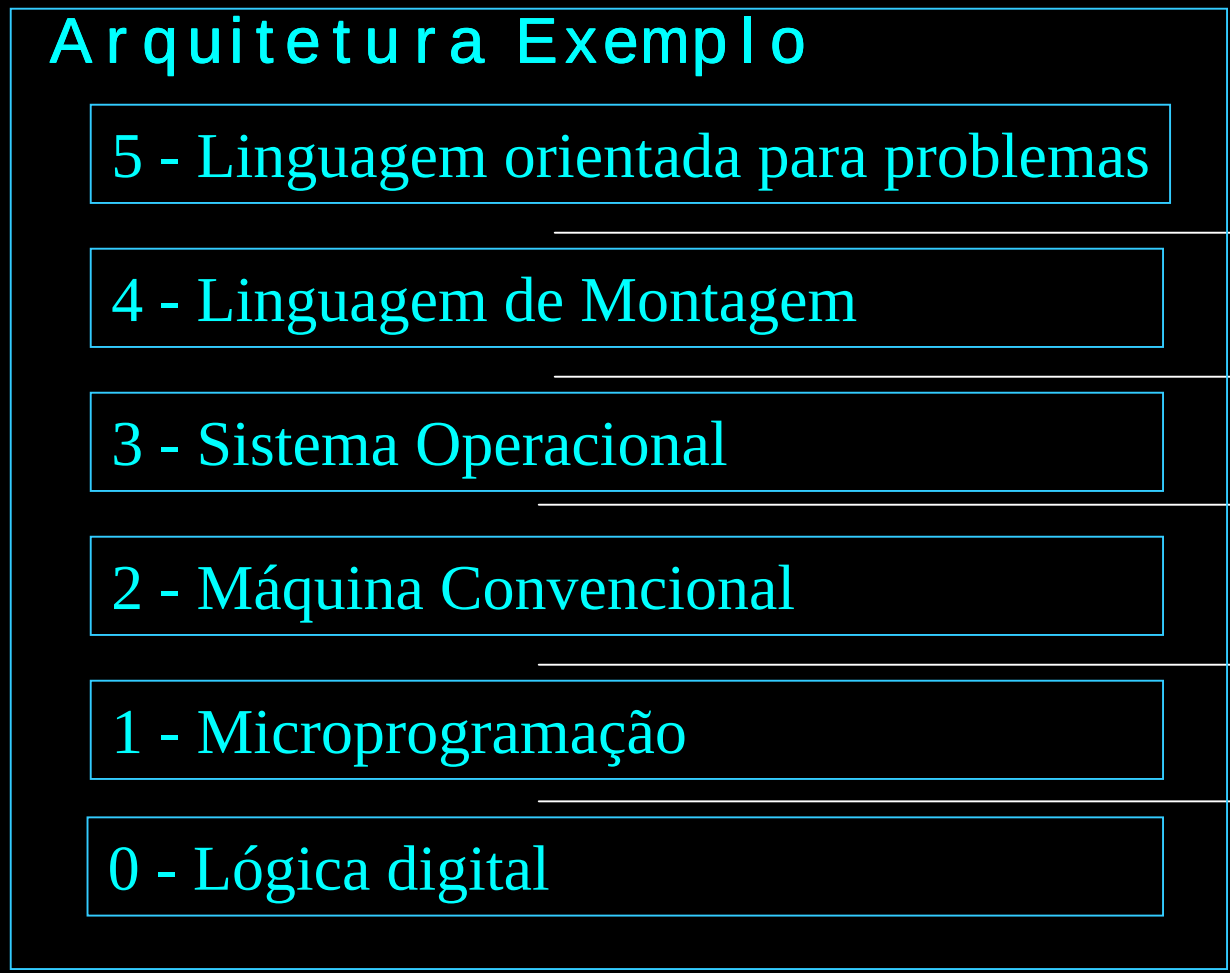


Programas Lk: interpretados por interpretador em Mi (inferior), ou traduzidos para Li de nível inferior.

Programas L3: interpretados por interpretador em M2 ou M1, ou traduzidos para L2 ou L1.

Programas L0 são executados diretamente pelos circuitos eletrônicos

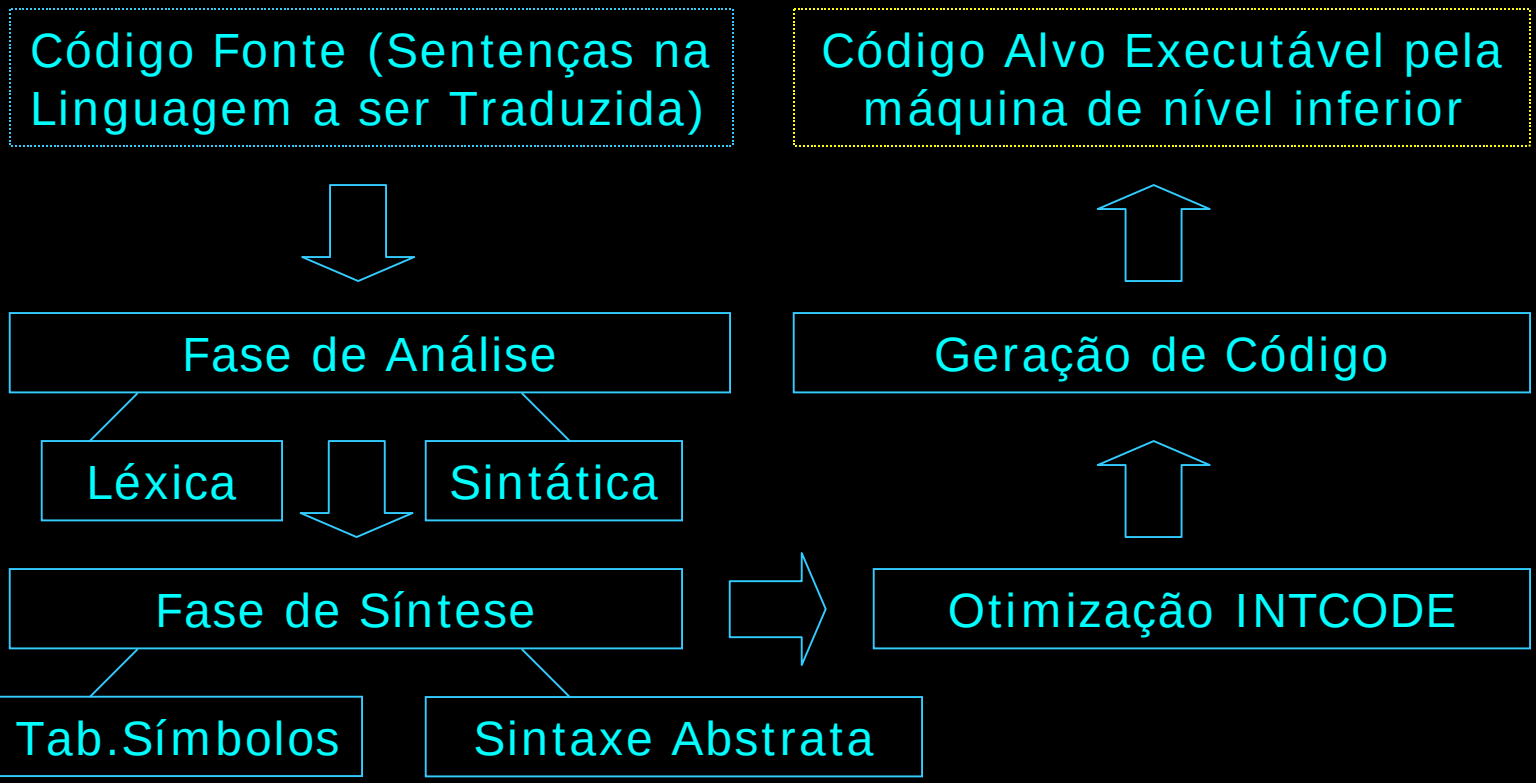
Linguagens de Programação Múltiplos Níveis de Atuação Exemplo



Linguagens de Programação
Compiladores e Interpretadores

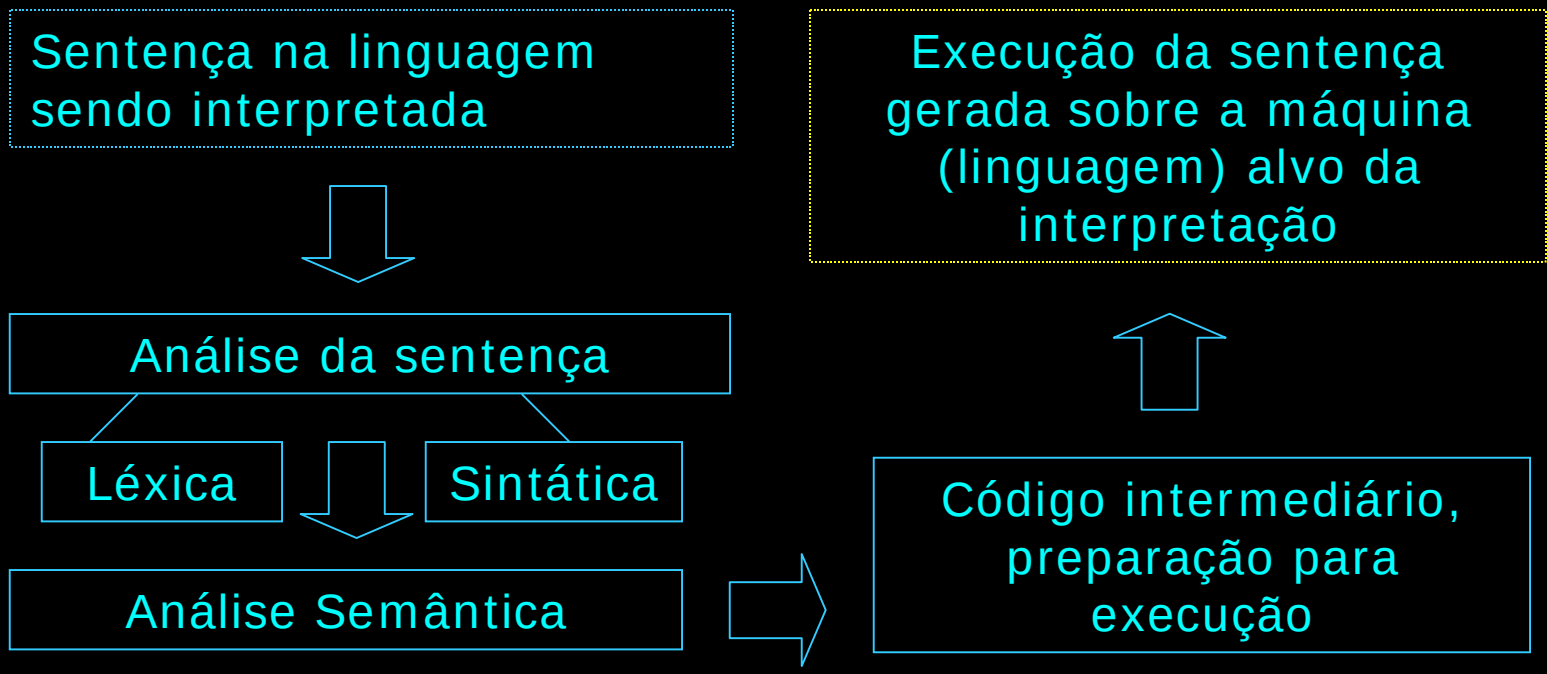
Um *COMPILADOR* é um programa que *TRADUZ* uma linguagem S (origem) para uma linguagem T (alvo).

Tarefas dividida em fases distintas



Linguagens de Programação
Compiladores e Interpretadores

Um *INTERPRETADOR* é um programa que executa *cada sentença* de uma linguagem S (origem) sobre uma máquina virtual M que define uma linguagem *m* (alvo).



Linguagens de Programação
Características

- **Suporte à abstração**
- **Facilidade no Aprendizado**
- **Consistência**
- **Clareza**
- **Auto-documentável**
- **Robustez**
- **Manipulação de Exceções**
- **Modularidade**

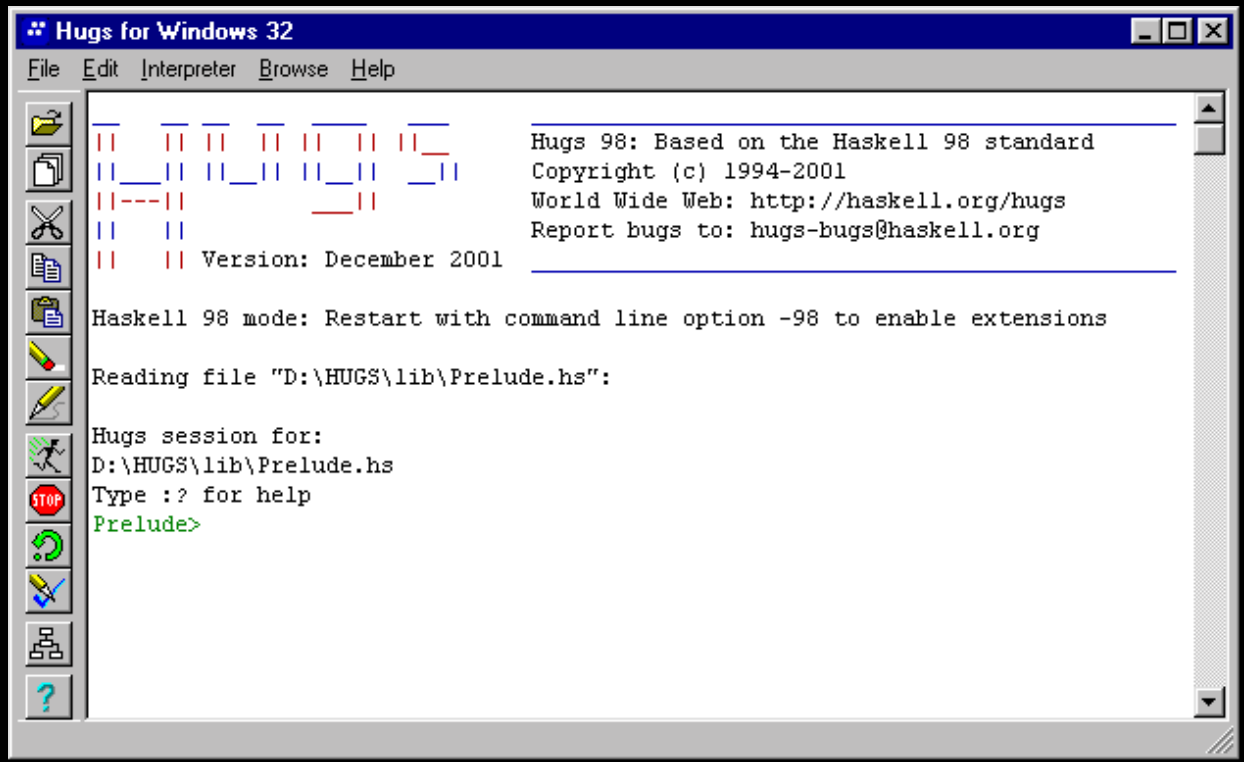
Programação Funcional

- Ênfase no Uso de Funções Matemáticas
- Enfatiza “o que fazer” em vez do “como fazer” (programação declarativa)
- Alto poder de avaliação de expressões e reconhecimento de padrões
- Formalismo nas definições
- Altamente auto-documentável
- Modelagem bastante razoável para problemas complexos
- Facilidade nas definições recursivas

Programação Funcional - Introdução ao HUGS

HUGS (Haskell Users Gofer System) - interpretador da linguagem *Haskell*

Winhugs



RUGS98 - Um Breve Sumário Técnico

- **Implementa um subconjunto de Haskell**
- ***Lazy Evaluation* (Avaliação por demanda)**
- **Inúmeros tipos de dados e funções predefinidas**
- **Sistema manipulador de tipos polimórficos.**
- **Funções de I/O sofisticadas.**
- **Alta portabilidade-Win32, UNIX, LINUX, MAC...**
- **Leve (Pequeno)**
- **Garbage Collector (coleta de lixo automática)**
- **Modular**
- **Funciona em modo *'read-eval-print'***
- **Interface Gráfica**

Testando Algumas Características do HUGS

O primeiro módulo que Hugs carrega é `PRELUDE.HS`.

Nele podemos encontrar as primitivas básicas que esse sistema provê.

O Prompt do HUGS permanece em:

Prelude>

indicando que está pronto para ser usado.

Comandos do Interpretador

Existem vários comandos que o interpretador pode executar.

Certamente, para os nossos propósitos, os comandos

:load, **:**reload, :edit, :quit

serão os mais usados para a carregar os programas que elaborarmos.

Uma lista completa dos comandos é apresentada com

:?

Arquivos do Interpretador

Formato texto - Código Fonte da Linguagem
Extensões *Default* (padrões):

.hs - Haskell Source

.lhs - Literate Haskell Source

Identificadores

Os identificadores são os nomes definidos pelo programador, ou seja, não é definido como uma palavra-reservada e/ou símbolo reservado da linguagem.

Exemplos: nome1, v1, x'', funcaoX, funcaox

Note que funcaox **e** funcaoX **são identificadores distintos em Haskell, o que significa que Haskell é *case sensitive* (sensível ao caso: maiúscula/maiúscula)**

Tipos de dados

Os tipos de dados definem classes de dados.

Exemplo de tipos básicos:

“UNIPAC”	::	String
1234	::	Integer
True	::	Bool
[1,2,3,4,5]	::	[Integer]

Exercícios...

Pesquisar sobre Haskell.

dica de site: http://www.haskell.org

Tarefas

- O que é Haskell?

- Onde Haskell tem sido utilizada?

- Ler o artigo *A Gentle Introduction To Haskell*.

Bibliografia

TANENBAUM, Andrew S. *"Organização Estruturada de Computadores"*, São Paulo 3ª. Edição Prentice/Hall do Brasil – 1995

A Gentle Introdution to Haskell - http://www.haskell.org

HUGS For Beginners - http://www.hugs.org