Linguagens de Programação

Apresentação da Disciplina

Andrei Rimsa Álvares
andrei@cefetmg.br

Introdução

- O que é uma linguagem de programação (LP)?

Diferentes pessoas têm diferentes definições de LPs!

Para você, o que é uma LP?

Algumas Definições

"A tool to aid the programmer" (in *part*)

C. A. R. Hoare

"a convention/method for writing descriptions which can be evaluated"

Chris Reade

"a notational system for describing computation in machine-readable and human-readable form"

Kenneth Louden

O que você acha dessas definições?

Uma Definição Melhor

"a notational system for describing **computation** in **machine-readable** and **human-readable** form"

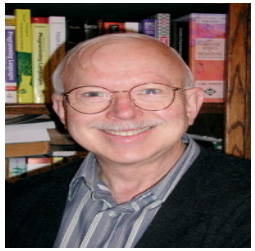
Kenneth Louden

O que é ser computável?

O que é ser legível por máquina?

O que é ser legível por humanos?

Computável



"a notational system for describing **computation** in machine-readable and human-readable form"

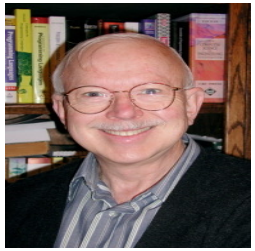
Kenneth Louden

- O que é ser **computável**?
  - Tudo que pode ser executado por um computador (máquina de *Turing*)

A maioria das linguagens de programação são *Turing* completas

Alguém conhece alguma linguagem não-Turing completa?

Legível por Máquinas



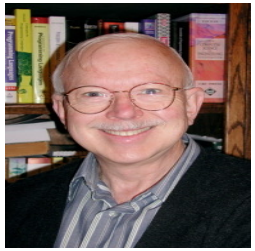
"a notational system for describing computation in **machine-readable** and human-readable form"

Kenneth Louden

- O que é ser **legível por máquinas**?
  - A linguagem deve ter uma **sintaxe** fixa para que o computador possa ler/analisar o programa
  - A linguagem deve ter uma **semântica** fixa para que o computador saiba o significado do programa a ser executado

Qual a diferença entre sintaxe e semântica?

Legível por Humanos



"a notational system for describing computation in machine-readable and **human-readable** form"

Kenneth Louden

- O que é ser **legível por humanos**?
  - A linguagem deve ser de fácil entendimento por humanos
  - Capacidade da linguagem de prover meios para suportar/ impor abstrações

Como gerenciar a complexidade?

Brecha Semântica

- A **brecha semântica** entre humanos e computadores

Humanos
- Interessados em modelar o mundo real
- Mais interessados no que **pode** ser feito do que no **como**

Máquinas
- *Entendem* dados apenas como sequências de bits (0s e 1s)
- *Entendem* apenas instruções de baixo nível

Linguagens de programação de alto nível fazem a **ponte** entre o homem e o computador diminuindo a brecha semântica ao prover notações de alto nível que ainda podem ser executadas por um computador

LINGUAGENS DE PROGRAMAÇÃO

Linguagens de Programação

- Quais linguagens de programação vocês já usaram? Quais linguagens vocês conhecem?

#	LPs

LPs usadas

#	LPs

LPs conhecidas

LPs Populares

- Ranking das linguagens mais populares♤

Sep 2024	Sep 2023	Change	Programming Language	Ratings	Change
1	1		Python	20.17%	+6.01%
2	3	↑	C++	10.75%	+0.09%
3	4	↑	Java	9.45%	-0.04%
4	2	↓	C	8.89%	-2.38%
5	5		C#	6.08%	-1.22%
6	6		JavaScript	3.92%	+0.62%
7	7		Visual Basic	2.70%	+0.48%
8	12	⇈	Go	2.35%	+1.16%
9	10	↑	SQL	1.94%	+0.50%
10	11	↑	Fortran	1.78%	+0.49%

Sep 2024	Sep 2023	Change	Programming Language	Ratings	Change
11	15	⇈	Delphi/Object Pascal	1.77%	+0.75%
12	13	↑	MATLAB	1.47%	+0.28%
13	8	⇊	PHP	1.46%	-0.09%
14	17	↑	Rust	1.32%	+0.35%
15	18	↑	R	1.20%	+0.23%
16	19	↑	Ruby	1.13%	+0.18%
17	14	↓	Scratch	1.11%	+0.03%
18	20	↑	Kotlin	1.10%	+0.20%
19	21	↑	COBOL	1.09%	+0.22%
20	16	⇊	Swift	1.08%	+0.09%

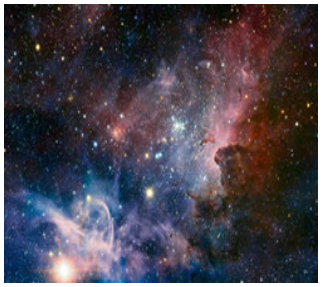
♤: Índice TIOBE (https://tiobe.com/tiobe-index/)

Por que estudar Linguagens de Programação?

1. Aumentar sua habilidade de expressar ideias
2. Aumentar o conhecimento para escolha de uma linguagem mais adequada
3. Melhorar a compreensão das linguagens que você usa
4. Melhorar o entendimento de como linguagens são implementadas
5. Facilitar o aprendizado de outras linguagens
6. Desenvolver sua própria linguagem?

Mais algum motivo?

Requisitos de Linguagens de Programação



Se um problema tem solução com algoritmos, então pode ser resolvido por uma LP

Universal

Implementada em um computador

Implementável

A LP deve prover as operações básicas para um determinado domínio de aplicação

Natural

A LP deve possuir uma implementação eficiente

Eficiente

Critérios de Avaliação de LPs

Facilidade de ler e entender programas

Legibilidade

Conformidade com a especificação

Confiabilidade

Algum outro critério?

Facilidade de usar uma linguagem para criar programas

Redigibilidade

O custo total final

Custo

Critérios de Avaliação: Legibilidade

- Simplicidade
  - Um conjunto gerenciável de funcionalidades e construções
  - Pouca multiplicidade de funcionalidades
- Ortogonalidade
  - Um conjunto relativamente pequeno de construções primitivas pode ser combinado de poucas maneiras
  - Toda possível combinação é legal
- Comandos de controle
  - Presença de comandos de controle bem estabelecidos (`while`)
- Tipos de dados e estruturas
  - Presença de facilidades para definição de estruturas de dados
- Considerações sintáticas
  - Composição de identificadores flexível
  - Palavras reservadas e formas para criar comandos compostos
  - Construções autodescritivas, palavras reservadas significativas

Critérios de Avaliação: Redigibilidade

- Simplicidade e Ortogonalidade
  - Poucas construções, número pequeno de primitivas, pequeno conjunto de regras para combiná-los
- Suporte a abstração
  - Habilidade de definir e usar estruturas ou operações complexas de forma que detalhes possam ser ignorados
- Expressividade
  - Um conjunto de formas relativamente convenientes de especificar operações (Ex.: inclusão do comando `for` em muitas linguagens de programação modernas)

Critérios de Avaliação: Confiabilidade

- Verificação de tipos
  - Verificar programas por erros de tipos
- Tratamento de exceções
  - Interceptar erros em tempo de execução para tomada de medidas corretivas
- Sinônimos (*aliasing*)
  - Presença de dois ou mais métodos distintos de referenciar a mesma área de memória
- Legibilidade e redigibilidade
  - Uma linguagem que não suporta a forma "natural" de expressar um algoritmo irá usar necessariamente abordagens "não-naturais"

Critérios de Avaliação: Custo

- Treinar programadores para usarem a linguagem
- Escrita do programa
- Compilação do programa
- Execução do programa
- Implementação da linguagem (compilador disponível?)
- Confiabilidade
  - Baixa confiabilidade leva a altos custos
- Manutenibilidade

Outros Critérios de Avaliação

- Portabilidade
  - Facilidade que programas podem ser movidos de uma implementação para outra
- Generalidade
  - A aplicabilidade em uma vasta gama de aplicações
- Bem-definidade
  - A completude e precisão da definição oficial de uma linguagem

PARADIGMAS DE LINGUAGENS DE PROGRAMAÇÃO

Linguagens de Programação

Paradigmas de Linguagens de Programação

- A definição de paradigmas de programação auxilia no agrupamento de linguagens similares, que podem ser classificadas em dois tipos:

Assertivas

- Baseada em comandos
- Linguagens de aplicação geral
- Paradigmas: imperativa, orientada a objetos, orientada a aspectos, orientada a eventos, programação concorrente, …

Declarativas

- Baseada no conceito de declarações (o que deve ser feito, não como)
- Ideal para construir relações (muito utilizado em inteligência artificial)
- Paradigmas: lógico e funcional

Assertivas: Programação Imperativa

- Centrado nos conceitos de estado global do sistema (variáveis) e ações (comandos)
- Primeiro paradigma a surgir, ainda dominante

- Vantagens
  - Eficiência (arquiteturas atuais são von Neumann)
  - Modelagem “natural” de aplicações do mundo real
  - Paradigma mais do que estabelecido
- Desvantagens
  - Implementações demasiadamente operacionais: definem “como fazer” e não “o que fazer”

Assertivas: Programação Orientada a Objetos

- Programas estruturados em módulos (classes) que agrupam um estado (atributos) e operações (métodos) sobre este estado
- Classes podem ser estendidas, via herança, e usadas como tipos (cujos elementos são objetos)
- Ênfase em reutilização de código
- Uso mais comum junto com o paradigma imperativo
- Visa corrigir o problema de várias funções acessarem variáveis globais (através de modificadores de tipos)
- Engloba conceitos como encapsulamento, herança, polimorfismo
- Torna a programação mais produtiva, facilita a manutenção
- Torna o código mais robusto, correto, extensível, ...

Assertivas: Programação Orientada a Aspectos

- Na programação orientada a objetos
  - Relacionamento entre grupos de entidades
  - Essencialmente estática
  - Mudança nos requisitos causam grande impacto

- Na programação orientada a aspectos
  - Pode ser vista como uma extensão da OO
  - Permite modificar dinamicamente o modelo para atender novos requisitos

Assertivas: Programação Orientada a Eventos

- Baseada em eventos assíncronos
  - Não interrompe sua execução ao processar entradas e saídas
  - O sistema operacional envia eventos, que são tratados pelo programa

- Não existe isoladamente: é uma extensão a linguagens imperativas e orientadas a objetos

- Exemplos
  - Interfaces gráficas, web, …

Assertivas: Programação Concorrente

- Vários processos executando simultaneamente
- Paradigma cada vez mais usual
- Pode ser utilizado em um único processador ou em vários paralelamente
- Pode ser distribuído

Declarativas: Programação Funcional

- Baseada em funções (como na matemática)
  - Uma função não possui efeitos colaterais
  - Não há o conceito de variáveis como mapeamentos em memória
- Implementa um mapeamento entre entrada e saída
  - Estilo declarativo: sem conceito de estado e comandos de atribuição
- Vantagens
  - Prova de propriedades
  - Concorrência é explorada de forma natural
- Desvantagem
  - Implementações em geral são ineficientes

Declarativas: Programação Funcional

- Exemplo (em SML)

```
fun fact 0 = 1
|   fact n = n * fact (n-1);
```

  - Uso extensivo de recursividade, funções de alta ordem, polimorfismo, entre outros

Declarativas: Programação Lógica

- Definição
  - Programas são relações entre entrada e saída
  - Estilo declarativo

- Implementam relações (Baseadas em cláusulas SE-ENTÃO)
  - Unárias (fatos), binárias, n-árias

- Computação: consultas que retornam verdadeiro ou falso
  - Em caso verdadeiro: o programa obteve sucesso
  - Em caso falso: diz que o programa falhou (mas não significa que a consulta seja falsa, apenas não foi possível inferir como verdadeira)

- Aplicação em inteligência artificial
- Vantagens e desvantagens semelhantes ao paradigma funcional

Declarativas: Programação Lógica

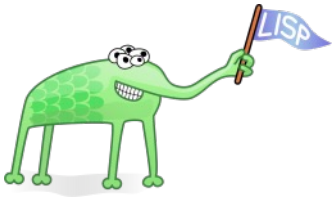
- Exemplo de programas e consultas

```
estrela(sol).

orbita(venus, sol).
orbita(terra, sol).
orbita(marte, sol).
orbita(lua, terra).

planeta(B) :- orbita(B, sol).

satelite(B) :- orbita(B, P),
               planeta(P).
```

```
?- orbita(venus, sol).
true.

?- planeta(mercurio).
false.

?- planeta(X).
X = venus ; X = terra ; X = marte.

?- satelite(X).
X = lua.
```

Paradigmas

- Em qual(is) paradigma(s) as LPs a seguir podem ser classificadas?



ERLANG

Borland Delphi

C++

Objective-C

OCaml

Scala

Java

Ruby

Microsoft C#.net

SWI Prolog

COBOL

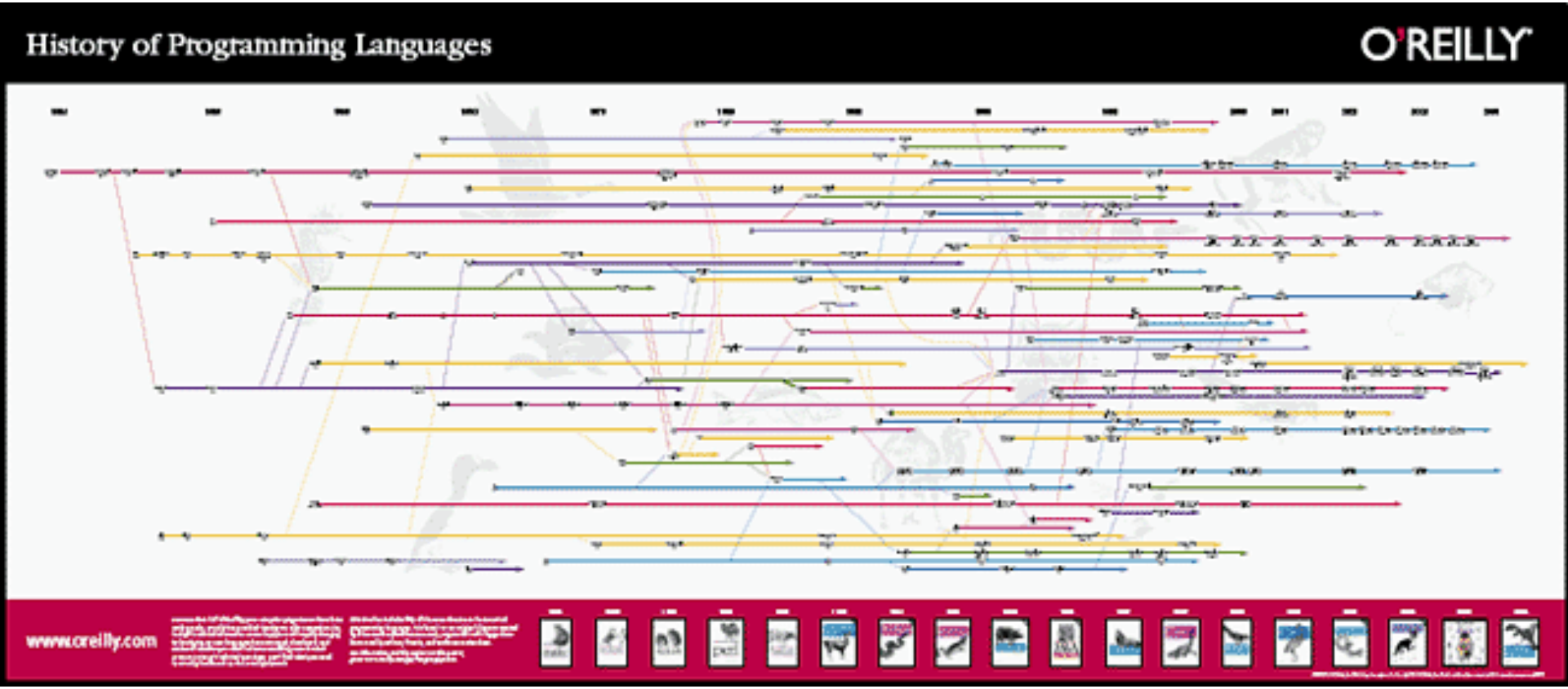
Fortran

C

Microsoft Visual Basic

Histórico

- Linha do tempo de linguagens de programação da O'Reilly♧



♧: https://bit.ly/3jTakbf

CEFET-MG CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS

INFORMAÇÕES, EMENTA, AVALIAÇÕES, REGRAS

Linguagens de Programação

Informações

- Todo o conteúdo da disciplina encontra-se no site: http://rimsa.com.br/page/classes/decom009

Linguagens de Programação

2ECOM.009

Ementa: Evolução das principais linguagens; Noções de sintaxe e semântica; Nomes, vinculações; Verificação de tipos e tipos de dados; Expressões e atribuição; Estruturas de controle; Subprogramas: ambientes de referenciamento, passagem de parâmetros; Tipos abstratos de dados; Orientação a objetos; Tratamento de exceções; Linguagens funcionais e lógicas

Informações

- Dúvidas sobre a disciplina apenas pela lista de discussão:
  https://groups.google.com/d/forum/decom009
  (decom009@googlegroups.com)

Erros nos slides valem 0,25pts extras

Cuidado: somente serão considerados se notificados até antes da segunda prova

Ementa

Primeiro módulo

1) Sintaxe e semântica
2) Sistemas de linguagens
3) Programação funcional: SML básico
4) Tipos
5) Polimorfismo
6) Escopo
7) Programação funcional: SML avançado

Segundo módulo

8) Cálculo λ
9) Localização de variáveis
10) Orientação a objetos
11) Tratamento de exceções
12) Parâmetros
13) Programação lógica
14) Modelos de custo

Avaliações

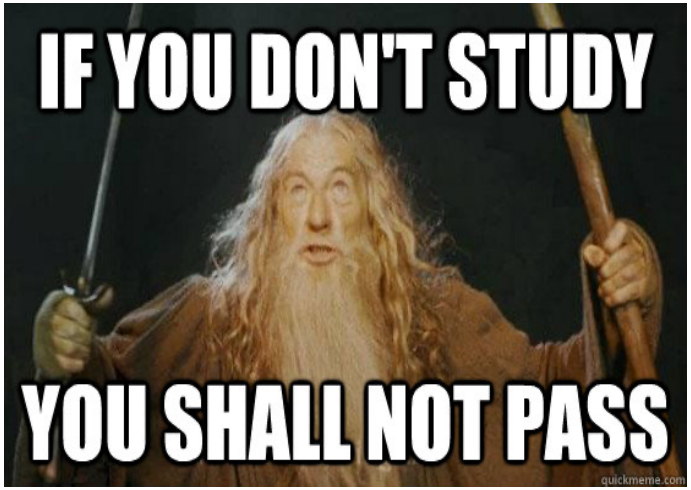
- Prova 1: 40pts
- Prova 2: 40pts
- Prova suplementar: 40pts
- Prova especial: 100pts

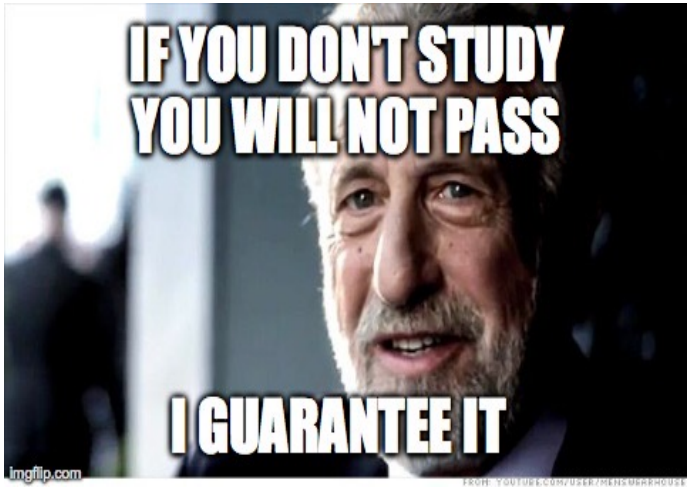
- Listas de exercícios: 14pts (1pt cada)
- Projeto: 8pts

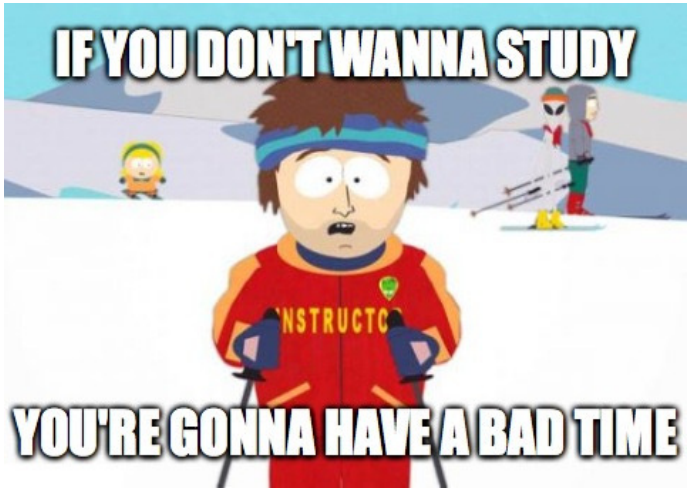
Regras

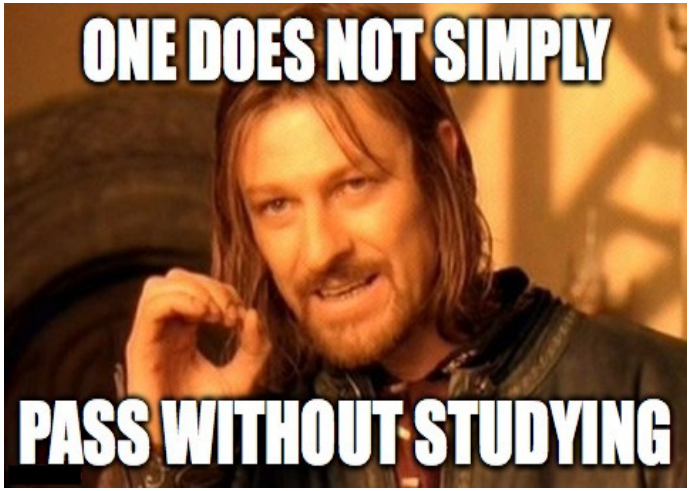
- A presença é obrigatória em 75% das aulas
  - Em todas as aulas haverá chamada
  - Não haverá abono de faltas, salvo nos casos previstos por lei
- A prova é individual e sem consulta
  - Após o início da prova, deve-se esperar no mínimo 30 minutos antes de entregar a prova
  - Colas serão penalizadas com nota zero
- Sobre as listas de exercícios
  - Fazer de forma **individual**
  - Fazer de forma **manuscrita** mais que 50% de cada lista e **digitalizar para PDF** para que ela seja considerada
  - Entregar na semana posterior ao conteúdo dado

Estudem!


IF YOU DON'T STUDY
YOU SHALL NOT PASS
quickmeme.com

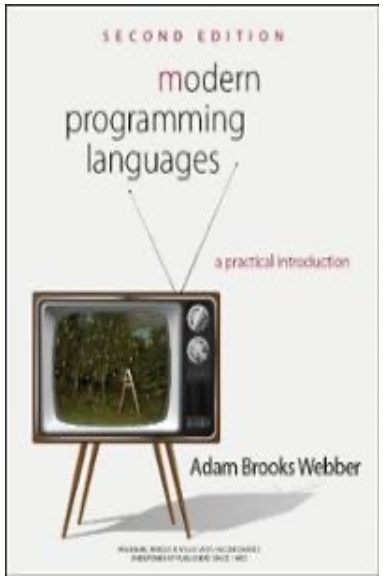

IF YOU DON'T STUDY
YOU WILL NOT PASS
I GUARANTEE IT
imgflip.com


IF YOU DON'T WANNA STUDY
YOU'RE GONNA HAVE A BAD TIME

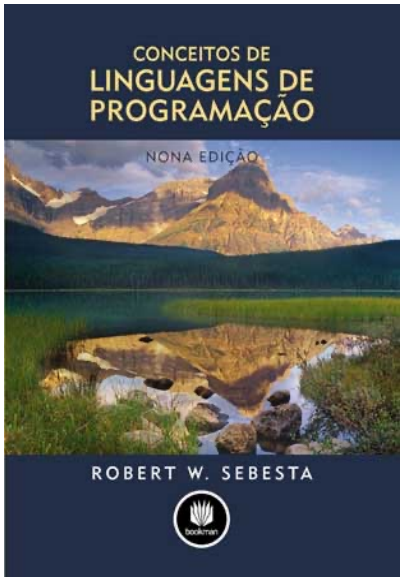

ONE DOES NOT SIMPLY
PASS WITHOUT STUDYING

BIBLIOGRAFIA

Bibliografia



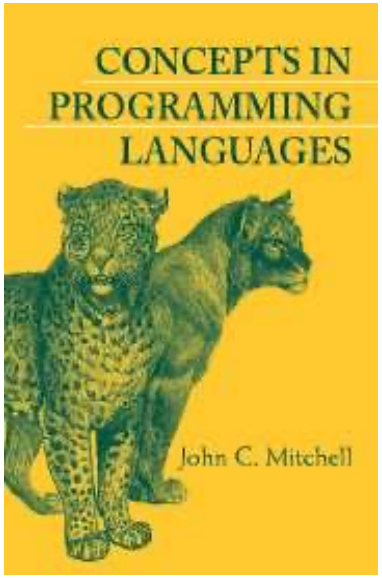
- A. Webber
- Modern Programming Languages: A Practical Introduction
- 2ª edição, 2010



- R. Sebesta
- Conceitos de Linguagens de Programação
- 9ª edição, 2011

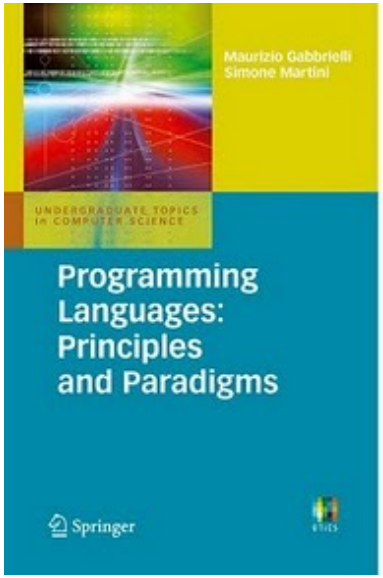
Bibliografia

- F. Varejão
- Linguagens de Programação
- 1ª edição, 2004

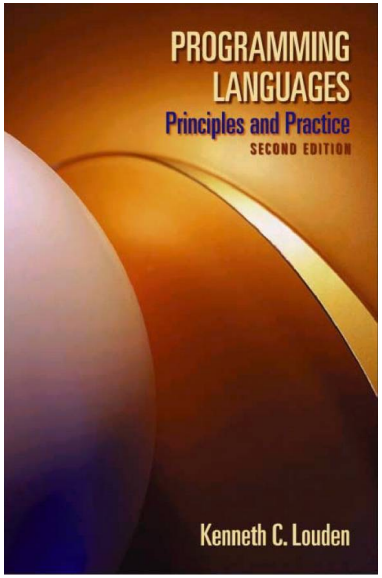


- J. Mitchell
- Concepts in Programming Languages
- 1ª edição, 2002

Bibliografia



- M. Gabbrielli; S. Martini
- Principles of Programming Languages: Principles and Paradigms
- 1ª edição, 2010



- K. Louden
- Programming Languages: Principles and Practices
- 2ª edição, 2002

ISSO É TUDO, PESSOAL!