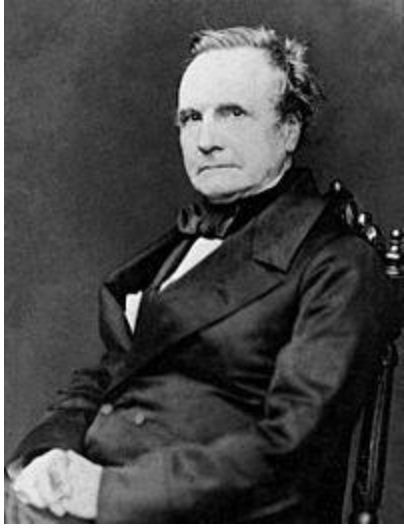
Apresentação criada pelo prof. Gabriel Alves
https://sites.google.com/site/gabrielalvesbsi/

A História da Programação

Antes dos Primórdios

- Cartões perfurados eram usados na indústria têxtil desde a década de 1800
  - Joseph-Marie Jacquard, mecânico, desenvolveu uma máquina têxtil que usava cartões perfurados para alterar a saída em 1801-1804
- **Semen Nikolaevich Korsakov**, em 1830, utilizou os cartões perfurados para armazenar e recuperar dados, a partir de máquinas que inventou

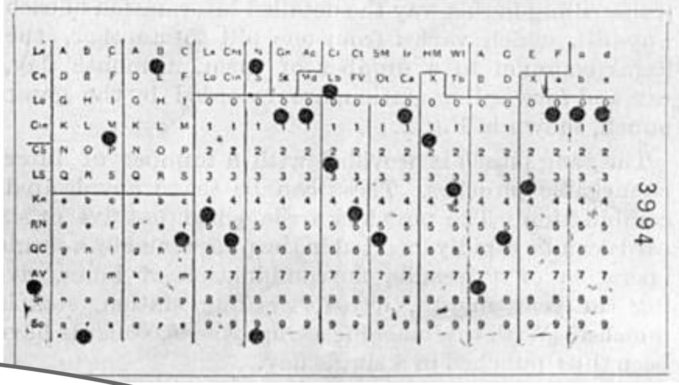
Antes dos Primórdios



- Charles Babbage
  - Projetou uma máquina analítica em 1833
    - Máquina mecânica (grande calculadora)
    - Capaz de realizar operações de soma, subtração...
    - Ministrou uma palestra que foi publicada em 1842 em francês
  - Construída uma pequena parte em 1871
    - Nunca foi finalizada
  - Seu trabalho não era de conhecimento dos inventores do computador moderno

Antes dos Primórdios

- Ada Lovelace
  - Conheceu Charles Babbage
  - Babbage pediu que traduzisse seu artigo
  - Ada levou quase 1 ano para traduzir, mas incluiu "algumas" anotações no artigo
    - Anotações maiores que o artigo original
- Mais de 100 anos depois a máquina de Babbage foi reconhecida como o primeiro modelo de computador e as notas de Ada foram reconhecidas como o primeiro programa



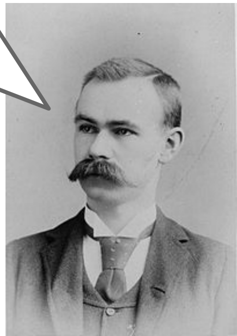
Os Primórdios

- Na décad
  Ho

Holerite

s.m. Documento que certifica o pagamento do salário de um funcionário, tanto pode ser depositado em sua conta bancária, como retirado pelo trabalhador no departamento da empresa destinado a esse propósito; **contracheque** ou demonstrativo de pagamento. (Etm. do inglês: hollerith)

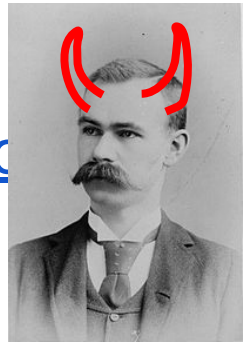
  - 191 … ndo a *Computing Tab* … TR)
  - 1924: A CTR foi renomeada para *Internatio… Business Machines* (IBM)



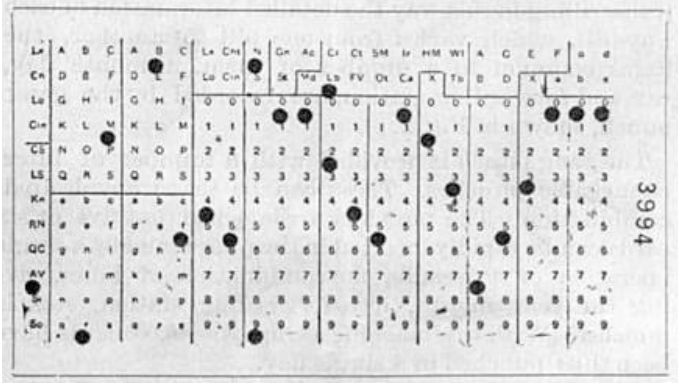
Os Primórdios

- Na década de 1930 a 1945, a subsidiária alemã da IBM, a *Deutsche Hollerith Maschinen* [illegible] (Dehomag), [illegible]ta
  - As máquin[illegible] população ([illegible] especificamente [illegible]
  - ***IBM and the Holocaust: The Strateg[illegible]iance Between Nazi Germany and America's M[illegible]t Powerful Corporation:*** http://www.amazon.com/IBM-Holocaust-Strategic-Alliance-Corporation-Expanded/dp/0914153277

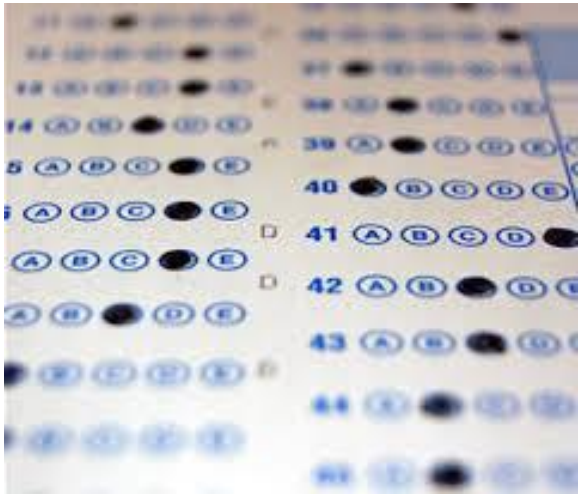
Mas, eu também tenho discípulos: http://gizmodo.uol.com.br/as-5-coisas-mais-malignas-que-aconteceram-gracas-a-ajuda-de-empresas-de-tecnologia/



Os Primórdios



- Os cartões perfurados foram cedendo espaço para a fita magnética nos anos de 1960
- São utilizados até hoje

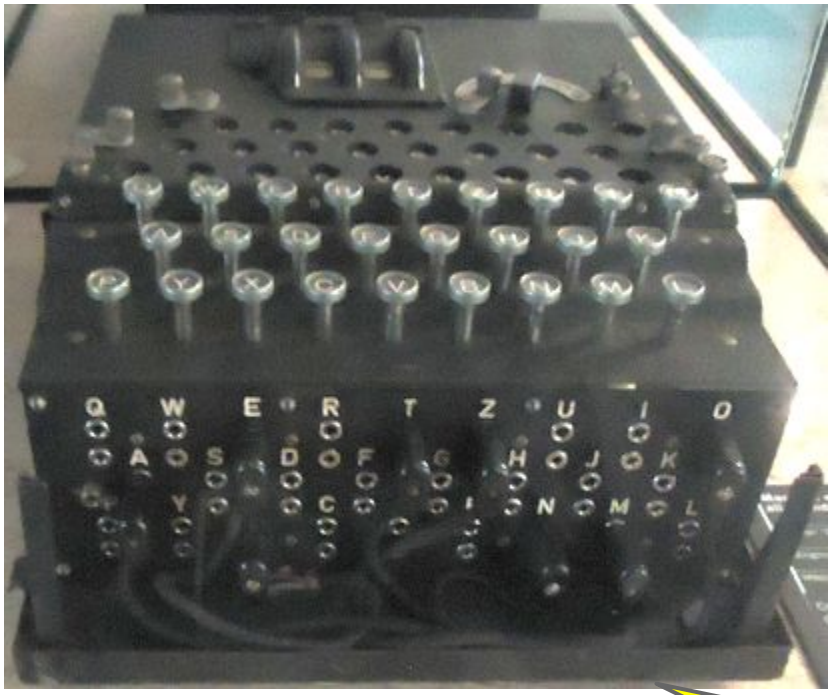


Os Primórdios

- Alan Turing
  - Trabalhou durante a II Guerra Mundial na inteligência britânica para quebrar códigos alemães
  - Participou da criação de máquinas físicas quebra destes códigos
    - ***Alan Turing - The Enigma:*** *http://www.amazon.com.br/Alan-Turing-Enigma-Inspired-Imitation-ebook/dp/B00M032W92/*
    - ***The Imitation Game:*** *http://www.imdb.com/title/tt2084970/*
    - ENIGMA...

Os Primórdios

- Alan Turing



Enigma (Musée de l'Armée)!

Os Primórdios

- Alan Turing
  - Projetou o modelo **lógico** para os computadores modernos em 1936 (data da publicação)
  - Morreu por envenenamento "acidental" por cianeto em 1954
    - Julgado e culpado por "vícios impróprios" (1952) após o fim da gerra. Humilhado publicamente.
    - O governo britânico se desculpou em 2009 ao já considerado "pai da informática" (1975)

Os Primórdios

- John von Neumann
  - Um dos mais importantes matemáticos do século XX
  - Projetou um *computador de programa armazenado* em 1946
    - Publicado em 1945
    - Baseado na máquina de Turing
    - Programas seriam armazenados na sua memória
      - Antes os programas ficavam em cartões perfurados
      - Melhor desempenho do computador

As Primeiras Linguagens

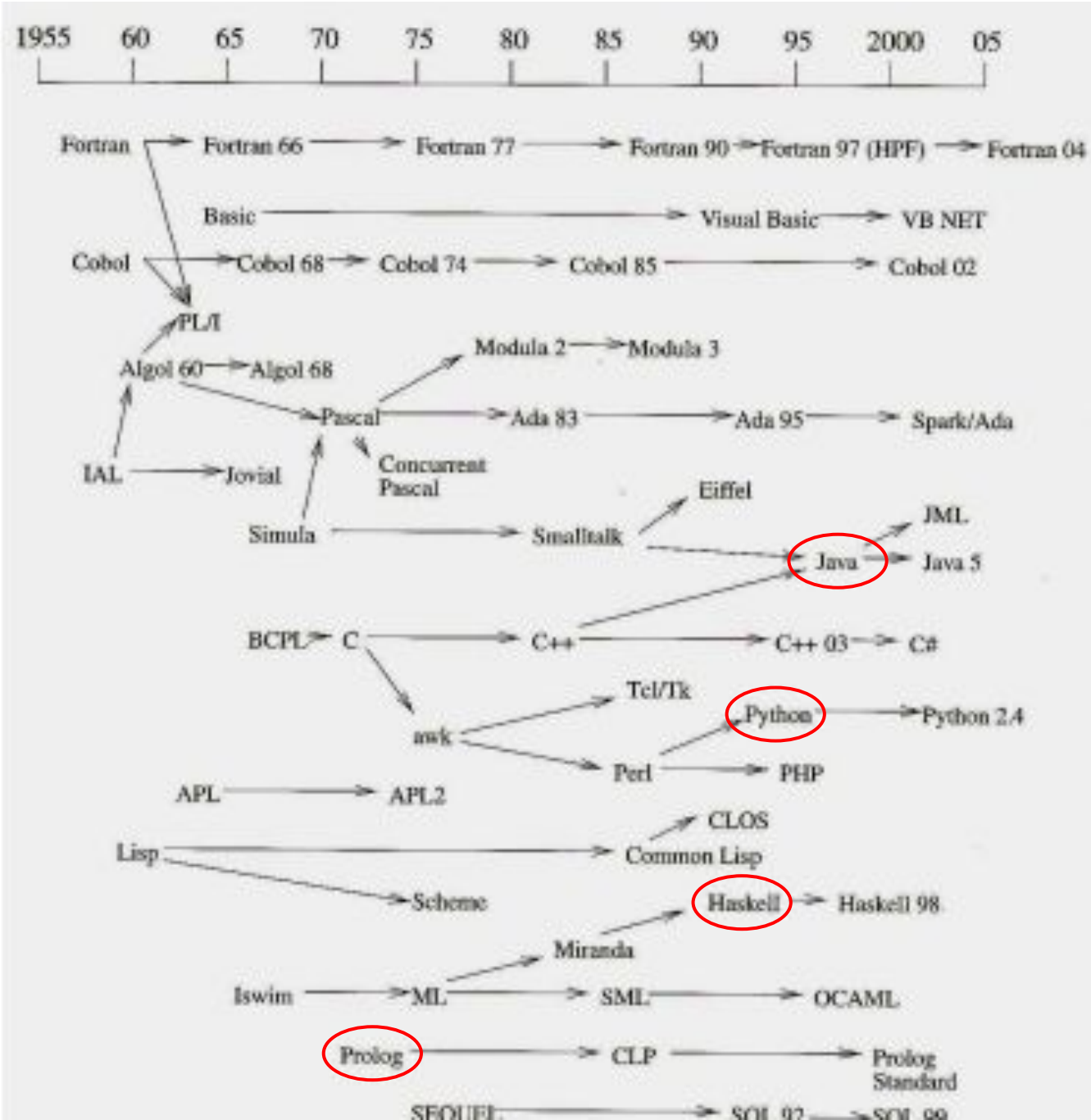
- A partir do trabalho de von Neumann surgiram as primeiras linguagens de programação
- Linguagens de máquina e *assembly* na década de 1940
  - 1948 - Primeiro *computador de programa armazenado* **Manchester Small-Scale Experimental Machine (SSEM)**, "**Baby**" *University of Manchester*
  - 1954 - Primeiro computador de produção de massa (**IBM 704)** com suporte a pontos flutuantes

As Primeiras Linguagens

- Na década de 1950 surgiram as linguagens de alta ordem (HOL, do inglês *high order language*)
  - Independe da arquitetura da máquina
  - **Fortran** (IBM), Cobol, Algol e Lisp
  - Projetos influenciaram linguagens futuras

Timeline

- Timeline Resumido



Timeline

- Timeline das Linguagens de Programação
  - http://imgs.obviousmag.org/archives/uploads/2007/2007090800_linguagens_programacaoprog_lang_poster.pdf

Paradigmas de Programação

EXERCÍCIO

Pesquise e escreva com suas palavras semelhanças e diferenças das linguagens de programação Haskell, Python e Java.

Conceitos Gerais

bla bla bla whiskas sache

Para que estudar diferentes linguagens de programação?

- Melhorar o entendimento das linguagens
- Maximizar a utilização das linguagens
  - Tirar todo o suco da laranja
- Senso crítico quanto às linguagens
- Seleção das linguagens "ideais" para a solução de problemas
- Facilitar o aprendizado de novas linguagens
  - VHDL, Outsystems, Delphi, Access, COBOL
- Criar uma nova linguagem
  - Nunca vou criar uma linguagem.
  - Nunca vou criar um framework? E uma API? E uma integração com outra linguagem ou framework?

Para que estudar diferentes linguagens de programação?

- Resumindo
  - **Utilização** eficaz e eficiente do que conhece
  - **Escolha** eficaz do que conhece
  - **Aprendizado** eficaz e eficiente do que não conhece

O que é uma linguagem de programação?

- É uma linguagem usada por uma pessoa para expressar um processo através do qual um computador pode resolver um problema
  - Computador: executa o processo expresso pela linguagem
  - Pessoa: programador que traduz o problema da linguagem natural para a de programação
  - Processo: atividade descrita pela linguagem de programação
  - Problema: o problema a ser resolvido

O que uma linguagem de programação deve prover?

- Deve dar suporte ao projeto, implementação, teste, verificação e manutenção do software
- Objetivos:
  - **Simplicidade**
  - **Clareza nas ligações**
  - **Confiabilidade**
  - **Suporte**
  - **Abstração**
  - **Ortogonalidade**
  - **Implementação eficiente**

O que uma linguagem de programação deve prover?

- **Simplicidade**: clareza e simplicidade sintática e semântica. Os programas devem ser fáceis de escrever e de ler.
- **Abstração**: capacidade de representar estruturas de dados, bibliotecas etc. Facilidade no reaproveitamento do código.

O que uma linguagem de programação deve prover?

- **Clareza nas ligações**: momento em que ocorre a ligação entre os elementos. O código abaixo é avaliado em tempo de compilação ou tempo de execução? O que ocorre em cada caso?

```
...
function inverso(a){ return 1/a; }
main() { print(inverso(x/0)); }
...
```

O que uma linguagem de programação deve prover?

- **Confiabilidade**: mecanismos de manipulação de exceção confiáveis. Determinismo na execução do programa independente da plataforma e da entrada. Tratamento de *memory leak*.
- **Suporte**: facilidade para instalação, aprendizado, suporte da comunidade, IDEs, *frameworks*, documentação (livros, tutoriais etc.)

O que uma linguagem de programação deve prover?

- **Ortogonalidade**: co
  construídos sobre u
  independente de op
  operações diferente
  coisa). Programas e
  tendem a ser mais s
  - A não-ortogonalidade
    eficiência

A linguagem de programação C, por exemplo, peca na ortogonalidade, ao permitir que o tipo “Void” seja utilizado em um “Struct”, mas não em elementos de um Array ou na declaração de uma variável.

O que uma linguagem de programação deve prover?

- **Implementação eficiente**: os recursos devem permitir a implementação eficiente em plataformas modernas.
  - As linguagens devem dar suporte à escrita dos programas considerando os "hardwares" modernos
    - Como programar em Java para a nuvem? E em Python? E em Assembly?
  - Os programas precisam ser eficazes mas também **eficientes**
    - As primeiras versões de Java eram criticadas por serem lentas
    - Ada foi projetada para ser usada em sistemas embarcados em aeronaves
      - Existe "tempo real" e "tempo Ada"

Projeto de Linguagens de Programação

- Existem três categorias que formam os **princípios** das linguagens de programação:, **Sintaxe**, **Semântica** e **Nomes e Tipos**.
- **Sintaxe**: Descreve a estrutura da linguagem. Define conjunto de operadores e símbolos que os programadores podem utilizar.

Projeto de Linguagens de Programação

- **Semântica**: Define o significado de um programa. Efeito dos comandos sobre os valores das variáveis. Trata de interpretadores, independência de plataforma, pilha de execução etc.

Projeto de Linguagens de Programação

- **Nomes e Tipos**: regras para nomear os elementos da linguagem (variáveis, funções, parâmetros etc.). Também trata do escopo, visibilidade e ligação. Impacta tanto a sintaxe quanto a semântica da linguagem.

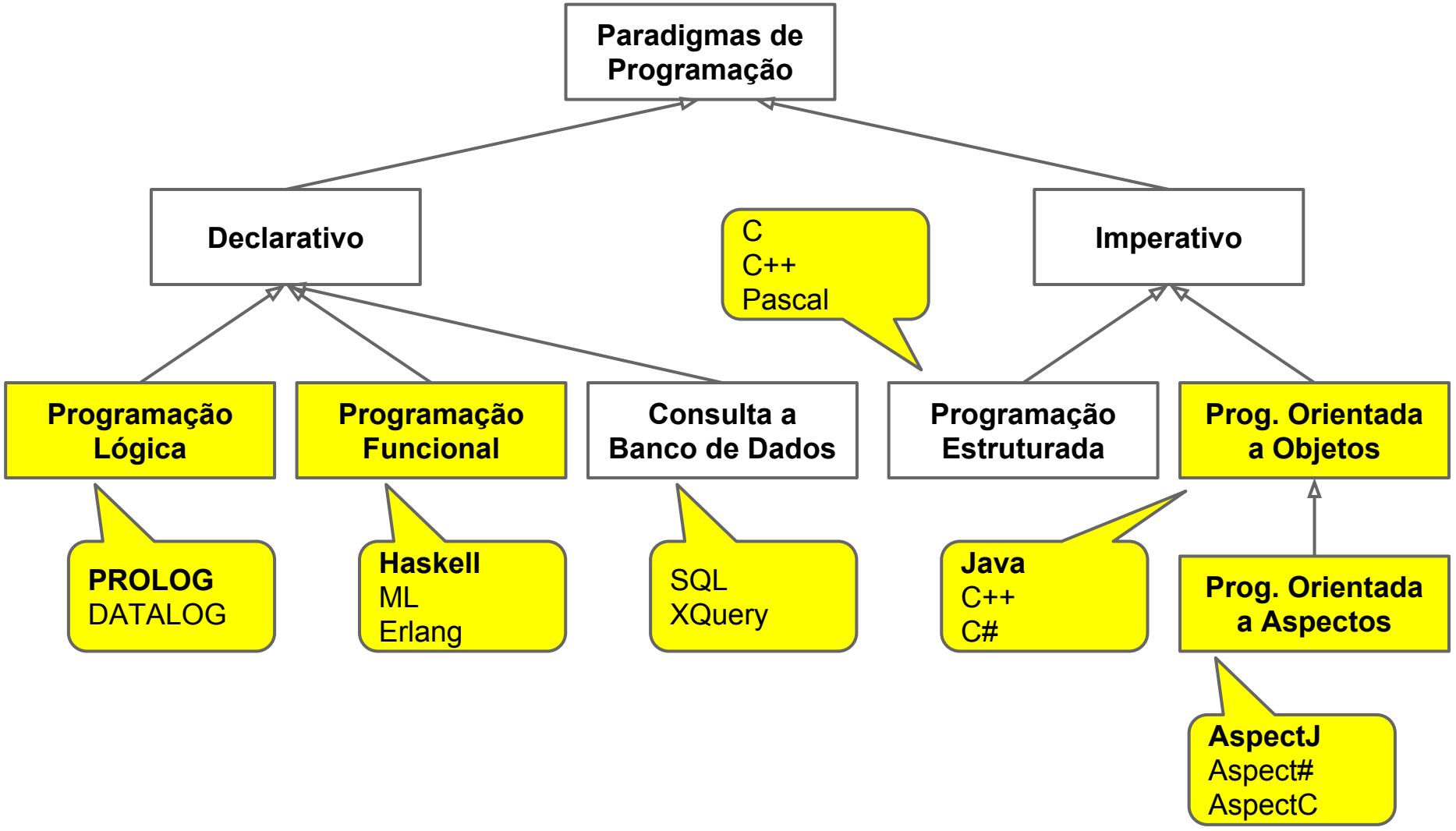
Paradigmas de Programação

- ...é um padrão de resolução de problemas que se relaciona a um determinado gênero de programas e linguagens - **Tucker, *Linguagens de Programação - Princípios e Paradigmas***
- É um estilo de programação.

Paradigmas de Programação

- Existem 4 principais paradigmas: imperativo, orientado a objetos, funcional e lógico **(Tucker)**
  - **Visão Paradigmas:**
    - https://docs.google.com/file/d/0BzcArSpeWDdsNmRtV1RaNkdZa2c/edit?usp=sharing
    - http://pt.wikipedia.org/wiki/Paradigma_de_programa%C3%A7%C3%A3o

Paradigmas de Programação



Paradigmas de Programação

- **Programação imperativa**: a computação é realizada através de comandos que alteram o estado do programa.
- **Programação procedimental**: também conhecida como **programação estruturada**. Programação imperativa baseada em procedimentos (constituído por um ou mais subprogramas, blocos, laços e outros elementos).
- **Programação estruturada**: também conhecida como programação procedimental, na qual as mudanças de estado são localizadas em subrotinas, parâmetros e retornos de funções.

Muitas vezes estes termos são "erroneamente" utilizados como sinônimos.

As linguagens orientadas a objetos também são imperativas.

Paradigmas de Programação

- O **paradigma imperativo** se baseia na **máquina de Turing**
  - Comandos
  - Estados
- No **paradigma orientado a objetos** o programa é organizado como uma coleção de objetos, cujo estado é alterado através das mensagens trocadas entre eles.

Paradigmas de Programação

- Em contrapartida ao paradigma imperativo está o **paradigma funcional**
  - Este paradigma se baseia no **cálculo lambda**
  - Não utiliza o conceito de estados
  - Modelagem através de funções matemáticas e chamadas recursivas
  - Tudo é função
- O **paradigma lógico** foca em **o que** deve ser obtido e não em **como** deve ser obtido
  - Podem ser chamadas de **baseadas em regras**
  - Expressa o não-determinismo de forma natural
    - Útil para problemas de especificação incompleta

Paradigmas de Programação

- Algumas linguagens suportam múltiplos paradigmas de programação
  - **Python, F#** e **Scala**: orientada a objetos, procedural e funcional
  - **C++, C#** e **Java**: orientada a objetos, genérica e procedura

Compilação e Máquinas Virtuais

- Os programas escritos em umas LP devem ser **analisados** e em seguida **traduzidas** para a linguagem da máquina que irá executá-lo
  - Esta máquina pode ser um computador ou um *software* (ex.: máquina virtual)
- No caso da máquina ser um computador, diz-se que a LP é **compilada**.
- No caso da máquina ser um interpretador, diz-se que a LP é **interpretada**.

Compilação e Máquinas Virtuais

Maior eficiência
Menor portabilidade.

- **LP compilada**
  - O compilador traduz o código fonte para a linguagem da máquina específica em que ele será executado.
  - O código fonte pode ser diferente de acordo com a máquina destino.

Menor eficiência
Maior portabilidade.

- **LP interpretada**
  - O código fonte é executado diretamente por um **interpretador** que executa o código fonte, tranduzindo-o para a máquina específica.
  - É possível compilar o código fonte para uma **máquina virtual** que é responsável por executar o código compilado, tranduzindo-o
    - Compilad

Maior eficiência (que interpretação).
Maior portabilidade (que compilação).

Paradigmas de Programação

EXERCÍCIO

1) Crie um programa estruturado que receba algo que represente um carro ou uma moto e imprima a quantidade de rodas. Faça o mesmo utilizando a orientação a objetos.

2) Reflita sobre os códigos criados: você geralmente cria programas orientados a objetos ou estruturados?

3) "Não existe agora, e nem existirá, uma linguagem de programação na qual seja difícil escrever programas ruins." - Larry Flon 1975.
Discorra sobre a afirmação de Larry Flon.