Modelos de Linguagens de Programação

– Aula 01 –

Tópicos

- **Apresentação da disciplina**
- Plano de ensino, forma e ambiente de trabalho
- Contextualização/conceitos
- Regras de convivência
- Atividades extraclasse

Apresentação da disciplina

- Modelos de Linguagens de Programação – MLP
- Professor: Leandro Krug Wives
- Página e e-mail do professor:
  http://www.inf.ufrgs.br/~wives
  wives@inf.ufrgs.br

Apresentação da disciplina

- MLP procura mostrar e discutir como as linguagens de programação funcionam
  - questões de projeto
  - características
  - mecanismos principais
  - pontos fortes e fracos

Apresentação da disciplina

- **Objetivo da disciplina:**
  - Estudar os princípios de projeto e as características dos principais modelos

- Envolve:
  - conceitos e princípios de linguagens de programação
  - paradigmas de programação (diferentes enfoques na solução de problemas)

Apresentação da disciplina

- NÃO é objetivo da disciplina:
  - apresentar todas as LPs disponíveis
  - aprofundar aspectos de sintaxe
  - discutir ou exercitar em detalhes alguma LP específica
  - apresentar ambientes de execução

Mas vamos estudar alguns desses aspectos...

Aspectos que serão reforçados

- A disciplina é basicamente conceitual!
- Adquirir e usar corretamente a terminologia adequada
- Exercitar a discussão e argumentação sobre o uso de uma determinada LP
- Exercitar a programação em paradigmas e modelos distintos

Apresentação da disciplina

- **Ao final do curso você:**
  - Dominará os princípios de LP
  - Conhecerá mecanismos eficientes para escolher a melhor linguagem para seu problema
  - Terá maior facilidade para aprender novas linguagens
  - Terá maior habilidade para usar eficientemente sua linguagem favorita
  - Conhecerá formas alternativas de expressar algoritmos
  - Dominará os princípios de linguagens de programação

Tópicos

- Apresentação da disciplina
- **Plano de ensino, forma e ambiente de trabalho**
- Contextualização/conceitos
- Regras de convivência
- Atividades extraclasse

Plano de ensino...

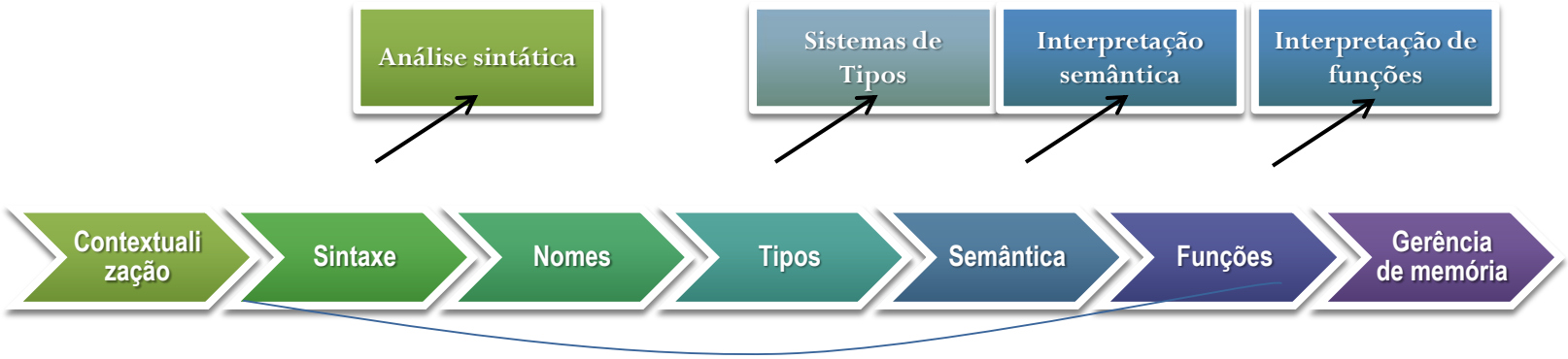
- Súmula e conteúdo programático
- Procedimentos didáticos, laboratórios
- Trabalhos, provas e avaliação

Vamos vê-lo no Moodle: (em conjunto com a lâmina seguinte)

http://moodleinstitucional.ufrgs.br/

- Para se logar: usar código do aluno + senha do portal do aluno
- Toda a comunicação entre o professor e os alunos será feita através dele (envio de trabalhos, etc.)

Conteúdo: visão geral



Adaptado de (Tucker, Noonan 2008)

Aulas em laboratório

- Contêm uma série de exercícios que demonstram conceitos
- O objetivo é permitir ao aluno pensar sobre o assunto de forma mais dinâmica que uma aula teórica
- São normalmente simples, mas nem todos os completam em uma aula (a cargo do aluno sua continuação)
- Todos os exercícios podem ser cobrados nas provas!

Trabalhos práticos

- Normalmente em grupos ou individuais (não muito estimulado)
- Diferentes grupos podem discutir as soluções de maneira conjunta, MAS cada grupo deve fazer seu próprio código
- Podem ser utilizadas bibliotecas, frameworks e componentes, mas devidamente referenciados e com destaque da contribuição do aluno
- A avaliação será individual: certifique-se de que os participantes saibam explicar qualquer parte do código e discuti-lo (domínio no assunto/solução é fundamental)
- Qualquer indício de fraude implica a anulação do trabalho para todos os envolvidos, sem direito a reclamação

Provas

- Fortemente dissertativas (normalmente)
- Serão cobrados:
  - conceitos e seu uso
  - argumentação correta
  - uso correto da terminologia
  - escrita coerente
- São devolvidas no final do semestre, mas podem ser vistas a qualquer momento, basta passar na sala do professor
- Podem exigir extrapolações do que foi dito explicitamente em aula: o objetivo é avaliar se o aluno consegue usar o que foi visto
- Qualquer indício de fraude implica a anulação da prova para todos os envolvidos

Presença

- Regimento da Universidade: 75% obrigatório
- Pedidos de revisão de notas só serão avaliados se o aluno tiver pelo menos 75% de presenças
- Alunos que não tenham atingido o grau mínimo de presença não poderão fazer Recuperação

Tópicos

- Apresentação da disciplina
- Plano de ensino, forma e ambiente de trabalho
- **Contextualização/conceitos**
- Regras de convivência
- Atividades extraclasse

Motivação e contextualização

- Por que estudar princípios e modelos de linguagens de programação?
  - Primeiro devemos refletir sobre:
    1. O que significa programar?
    2. O que é uma linguagem de programação?
    3. O que é um modelo de programação?

O que é programação?

- É a arte de dizer a outro ser humano o que se quer que o computador execute (Donald Knuth)
- Expressão de uma idéia tal que possa ser executada por uma máquina e compreendida por um Ser Humano

- Execução: eficiência, portabilidade, custo, etc.
- Compreensão: simplicidade, poder de expressão, generalidade, etc.

O que é uma LP?

- notação utilizada para especificar ações a serem executadas por um computador
- facilita a expressão e a comunicação de ideias computacionais entre pessoas (compreende conceitos que um programador usa para resolver problemas de programação)
- Visa o hardware (operações que ele executa, seus requisitos e restrições)
- Visa a aplicação (o que é preciso resolver)

O que é uma LP?

Quais linguagens de programação você conhece?

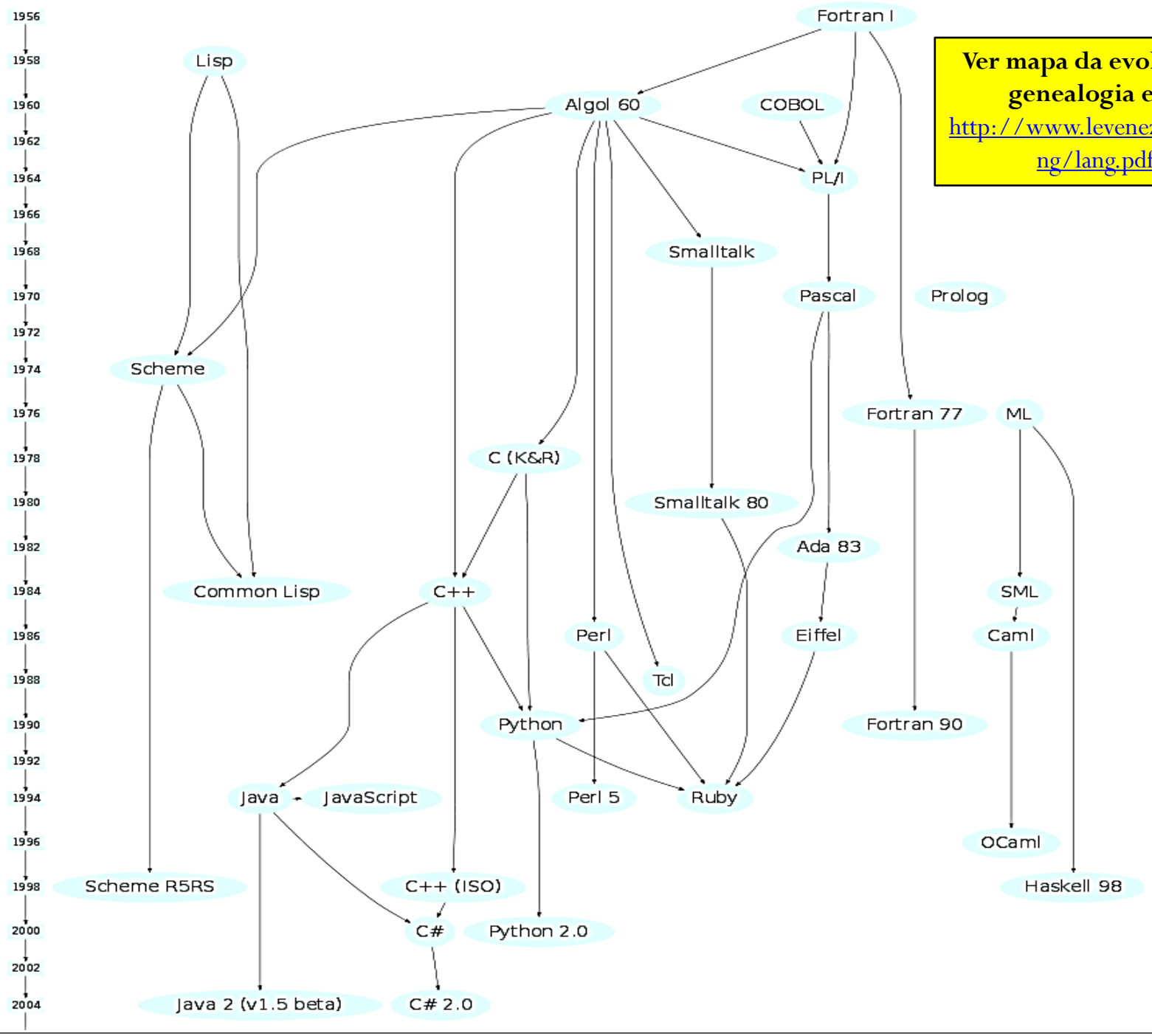
- Pascal, C, C++, Python, PHP, Java, Lisp, Scheme, ML, Fortan, Cobol, AWK, SED, Eiffel, Ruby, JavaScript, BASIC, Logo, Modula-3, Ada, APL, Algol, PERL, C#, .NET, Assembly, Verilog, SystemC, Caml, Ocaml, CLU, Haskell, Matlab, Maple, Oberon, Occam, PL/I, Prolog, Simula, Smalltalk, SNOBOL, SR, TCL, Delphi, Visual Basic, Visual C++, Clipper, FoxPro, SQL, Lua …

- **Único programa implementado em mais de 1300 linguagens diferentes:** http://www.99-bottles-of-beer.net/

Por que há várias linguagens?

- Evoluímos: aprendemos a fazer melhor!
- Há domínios de aplicação com necessidades diferentes
- Existem fatores sócio-econômicos (interesses de fabricantes, instituições...)
- Podem haver necessidades de um Hardware específico

Genealogia das principais LPs



Ver mapa da evolução/ genealogia em:
http://www.levenez.com/lang/lang.pdf

Fonte: http://merd.sourceforge.net/pixel/language-study/diagram.html

Por que há várias linguagens?

- Evoluímos: aprendemos a fazer melhor!

➔ Vamos ver o Histórico…

Primeiros programas

- Programação por chaves ou linguagem de máquina (binária)

```
27bdffd0 afbf0014 0c1002a8 00000000 0c1002a8 afa2001c 8fa4001c
00401825 10820008 0064082a 10200003 00000000 10000002 00832023
00641823 1483fffa 0064082a 0c1002b2 00000000 8fbf0014 27bd0020
03e00008 00001025
```

- Computadores imensos, caros, capacidade de cálculo de uma calculadora de mão
- Tempo do computador valia mais que o tempo do programador
- Problemas?
  - Programação sujeita a erros
  - Longo tempo de programação
  - Depuração e manutenção de código...

Linguagens de montagem

- Uso de mnemônicos:

```
addiu   sp,sp,-32       move    v1,v0           B: subu  a0,a0,a0
sw      ra,20(sp)       beq     a0,v0,D         C: bne   a0,v1,A
jal     getint          slt     at,v1,a0 slt    at,v1,a0
nop                     A: beq  at,zero,B       D: jal   putint
jal     getint          nop                     nop
sw      v0,28(sp)       b       C               lw       ra,20(sp)
lw      a0,28(sp)       subu    a0,a0,v1 addiu  sp,sp,32
                        jr      ra
                        move    v0,zero
```

  - Correspondência direta com a linguagem de máquina (ADD = 1010101)
  - Problemas?
    - LP muito associada à operação do hardware
    - novas máquinas ➔ reprogramação
    - máquinas mais potentes e mais baratas ➔ programas maiores, tempo do programador encarece
  - Necessidade: LP independente de máquina e computação numérica

Linguagens de alto nível

- Fortran (meados de 1950)
- ALGOL, LISP mais ou menos em paralelo
- Problema: descrição em linguagem de alto nível, mas execução em linguagem de máquina
- Surgimento de compiladores:
  - Não há mais uma correspondência direta entre o programa descrito em alto nível e o programa executado pelo HW
  - Diferentes escolhas na implementação do compilador afetam a definição de novas linguagens

Engenharia de Software

- Engenharia de Software (ES) (década de 70)
  - Abstração de dados: definição de tipos
  - Abstração de controle: comandos, procedimentos
  - Preocupação com programação em larga escala: módulos e programação estruturada

  Exemplos de linguagens populares:
  - Uso acadêmico: Algol (algoritmos), Pascal (tipos de dados)
  - Uso comercial: Cobol (arquivos), PL/I (uso amplo)

➔ Objetivo era facilitar a manutenção e o desenvolvimento sistêmico do software

Estruturação e modularização

- Década de 80:
  - Ênfase em mecanismos de LP e abstrações
  - Correção de programas: verificação de tipos, exceções
  - Programação concorrente, distribuída e de tempo real
  - Programação baseada em estruturas (TADs)
  - Princípios de O.O. (herança)

- Exemplos de linguagens:
  - Uso acadêmico: Pascal / Modula
  - Programação de tempo real: Ada 83
  - Orientada a objetos: Smalltalk

Estruturação, modularização, programação visual

- Década de 90:
  - Estruturação de dados: encapsulamento
  - Estruturação da computação: classe
  - Estruturação do programa: classes e objetos
  - Programação para Internet: plataforma neutra

  - Exemplos de linguagens:
    - Pascal / Delphi
    - C / C++
    - Ada83 / Ada95
    - Java
    - Scripts

Por que há várias linguagens?

- Evoluímos: aprendemos a fazer melhor!
- Há domínios de aplicação com necessidades diferentes
- Existem fatores sócio-econômicos (interesses de fabricantes, instituições...)
- Podem haver necessidades de um Hardware específico

Domínios de aplicação

- Uma LP é capaz de expressar QUALQUER programa de computador?
- Existem linguagens adequadas a classes de problemas específicos: isso simplifica a linguagem, otimiza sua implementação
- Exemplos:
  - Matlab: manipulação de matrizes e vetores
  - C: programação de baixo nível
  - Java: programação internet, O.O.
  - Fortran: cálculos, programação científica

Por que há várias linguagens?

- Evoluímos: aprendemos a fazer melhor!
- Há domínios de aplicação com necessidades diferentes
- Existem fatores sócio-econômicos (interesses de fabricantes, instituições...)
- Podem haver necessidades de um Hardware específico

Algumas considerações

- É possível escrever qualquer programa em qualquer linguagem de programação: todas as LPs são igualmente poderosas
- Porém, a facilidade para se escrever o programa é diferente entre as LPs
- Como saber:
  - Qual LP usar?
  - Se precisamos de uma nova LP?
  - Se devemos usar uma nova LP?

Linguagem X Resolução de Problemas

- Hipótese de Sapir-Whorf (lingüística):

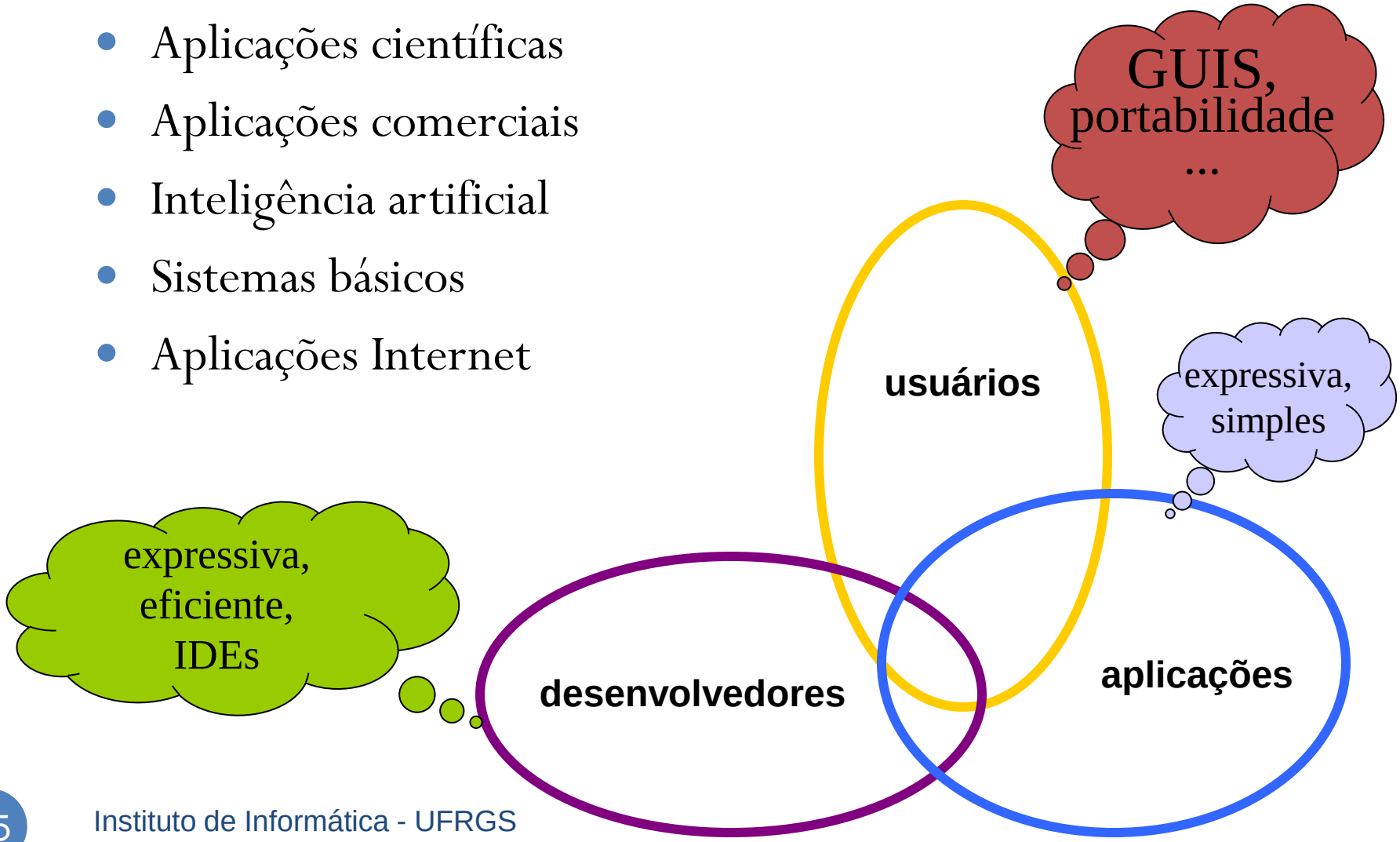
  "A estrutura da linguagem define os limites do pensamento"

  - Uma dada linguagem pode facilitar ou dificultar certos modos de pensar (aumentar ou diminuir o nível de abstração)
  - Pessoas podem ter dificuldades em se expressar, em abstrair por não conhecerem direito uma língua (sua ou outra)
  - Embora nenhuma linguagem possa nos impedir de encontrar uma solução, uma dada linguagem pode nos guiar para uma classe de soluções e para uma forma de abordar o problema

  → impacto na qualidade dos programas

Domínios vs interesses

- Aplicações científicas
- Aplicações comerciais
- Inteligência artificial
- Sistemas básicos
- Aplicações Internet



Sobre LPs...

- Quais são as principais diferenças e similaridades?
- O que torna uma LP boa?
- O que torna uma LP mais adequada para uma aplicação em particular?
- O que torna uma LP famosa ou muito usada?
- Boa e famosa significam a mesma coisa?

Modelos!

Principais modelos/paradigmas

- Imperativo
  - Estruturado (Von Neumann)
  - Orientado a objetos
- Declarativo
  - Funcional
  - Lógico (restrições)

- Sequencial versus concorrente

OBS: há outras visões! Há linguagens multiparadigma!

Tópicos

- Apresentação da disciplina
- Plano de ensino, forma e ambiente de trabalho
- Contextualização/conceitos
- **Regras de convivência**
- Atividades extraclasse

Regras de convivência

- **Horário da aula:**
  - Pequenos atrasos (até 15 min.) são tolerados, MAS indesejados
  - Nas provas, não chegue após esse intervalo de tolerância, pois não poderá entrar na aula

- **Chegou atrasado?**
  - Seja invisível
  - Não bata a porta
  - Não inicie conversas

- **Prime pela educação e respeito aos que dividem esta atividade com você**

Regras de convivência

- **Durante a aula:**
  - Sua participação é importante! Não hesite em contribuir com comentários/questões que possam enriquecer a discussão!
  - Não saia com dúvidas, significa mais tempo estudando mais tarde
  - Celulares e similares: Desligados!
- **Assuntos NÃO relacionados com a aula:**
  - Resolva-os no intervalo!
  - Conversas CURTAS entre colegas são toleradas, MAS indesejadas.
- **Precisa discutir com o colega?**
  - Seja BREVE e fale em voz baixa.
  - Retire-se da sala usando o princípio da INVISIBILIDADE…

Regras de convivência

- Alguma coisa não vai bem?
  - FALE LOGO para termos a chance de consertar!

- Não lembra de algo (avaliação, regras, material, cronograma):
  - Fora do horário de aula: consulte o Moodle ou mande e-mail para o professor
  - Em horário de aula: consulte o professor

Tópicos

- Apresentação da disciplina
- Plano de ensino, forma e ambiente de trabalho
- Contextualização/conceitos
- Regras de convivência
- **Atividades extraclasse**

Atividade para a semana...

- Enquetes no Moodle:
  - Perfil global dos alunos
  - Conhecimento prévio e expectativas
- Leitura do capítulo 1 do livro:
  - Sebesta. Robert W. **Conceitos de Linguagens de Programação**. Bookman: Porto Alegre, 2000.