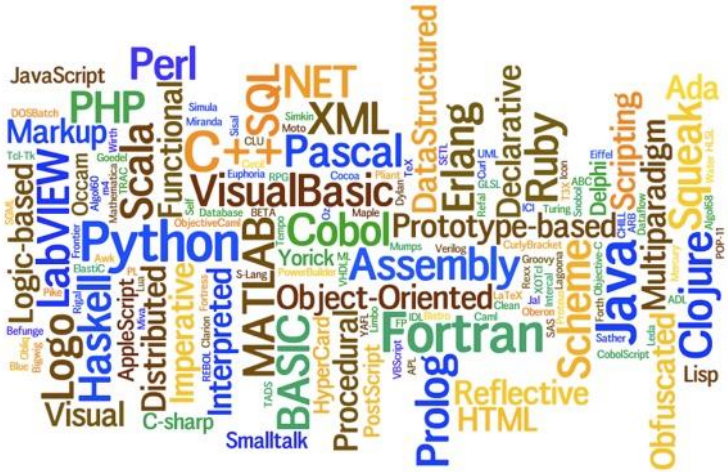
CIÊNCIA DA COMPUTAÇÃO
UENF

Por que estudar Linguagens de Programação



Prof. Ausberto S. Castro V.
ascv@uenf.br

Bibliografia Básica

Cap. 1

Porque estudar Conceitos de LP?

1. Aumento da capacidade para expressar ideias

2. Embasamento para escolher linguagens adequadas

3. Aumento da habilidade para aprender novas linguagens

4. Melhor entendimento da importância da implementação

5. Melhor uso das linguagens já conhecidas

6. Avanço geral da computação

6

Porque estudar Conceitos de LP?

- **Aumento da capacidade para expressar ideias**
  - Compreensão limitada da linguagem natural = limitados na complexidade de expressar seus pensamentos.
  - Conhecimento de muitas LP reduz as limitações no desenvolvimento de software

- **Embasamento para escolher linguagens adequadas**
  - Programadores "profissionais":
    - treinamento in-house
    - Pouca educação formal
    - Linguagens associadas ao trabalho da organização
  - Novos projetos: Uso de linguagens com as quais estão familiarizados, mesmo ultrapassadas
    - Ultrapassada: pobre, limitada em recursos

Porque estudar Conceitos de LP?

- **Aumento da habilidade para aprender novas linguagens**
  - Contínua evolução das linguagens e das metodologias, significa aprendizagem contínua
  - Maior conhecimento de vocabulário e conceitos, significa maior facilidade para aprender uma nova linguagem
    - Objetos e classes: Java, C++, Smalltalk,

- **Melhor entendimento da importância da implementação**
  - Porque a linguagem foi projetada daquela maneira?
  - Implementações para determinada aplicação
    - SQL para Métodos Numéricos?
    - Ruby para homepages?
    - MATLAB para Banco de Dados?
  - Encontrar e corrigir "bugs" do programa (só por especialistas)
  - Visualizar como um computador executa as instruções
    - Algoritmo recursivo mais lento que um iterativo
    - Subprogramas com chamadas frequentes: altamente ineficiente

Porque estudar Conceitos de LP?

- **Melhor uso das linguagens já conhecidas**
  - Novas necessidades ou necessidades específicas
  - Projeto de interfaces para usuários: projetos complexos
- **Avanço geral da computação**
  - Porque uma linguagem se tornou popular?
    - Facilidade vs. eficiência
    - Marketing
    - Exemplo: Algol 60 vs FORTRAN (1962)
  - Nem sempre as linguagens mais populares são as melhores
  - Novas linguagens associadas a novas arquiteturas, novos paradigmas, novas pesquisas.

Domínios da Programação

Domínios da Programação

- **Aplicações científicas**
  - Computações aritméticas com ponto-flutuante
- **Aplicações comerciais**
  - Facilidades para produzir relatórios
- **Inteligência Artificial**
  - Computações simbólicas (não numéricas)
- **Programação de Sistemas**
  - Eficiência na execução – sistemas operacionais
- **Linguagens de Scripting**
  - Listas de comandos, shells
- **Linguagens para Propósitos Especiais**
  - Relatórios comerciais, máq. Programáveis, simulação
- **Linguagens para a web**
  - Menus, cliente-servidor,

Qual é a melhor linguagem de programação?


C
Language


C#


C++


OBJECTIVE-C


.f90


Microsoft®
Visual Basic®
for Applications


python™


perl


JavaScript

Critérios de Avaliação de LP

Critérios de Avaliação de LP

- **Definição**
  - Conceitos fundamentais das várias construções e capacidades da linguagem
  - Impacto sobre o processo de desenvolvimento e manutenção de software
- **Critérios:**
  - Legibilidade
    - Facilidade com que os programas são lidos e entendidos
  - Capacidade de Escrita
    - Quão facilmente pode ser usada para criar programas
  - Confiabilidade
    - Comportamento de programas sob todas as condições
  - Custo
    - Treinamento de programadores, escrita, compilação, execução, implementação
  - Portabilidade

Características vs. Critérios

Característica	Critérios		
	Legibilidade	Capacidade de escrita	Confiabilidade
Simplicidade/ortogonalidade	●	●	●
Estruturas de Controle	●	●	●
Tipos de dados e estruturas	●	●	●
Projeto de Sintaxe	●	●	●
Suporte para abstração		●	●
Expressividade		●	●
Verificação de Tipos			●
Manipulação de Exceções			●
Aliasing restrito			●

Legibilidade

- **Facilidade com que os programas são lidos e entendidos**
- **Características:**
  - Simplicidade global
    - Linguagem com um grande número de componentes básicos é mais difícil de ser aprendida (C vs Java)
    - Multiplicidade de recursos

```
count = count + 1
count += 1
count++
++count
```

    - Sobrecarga (overloading) de operadores: um símbolo tem mais de um significado. +: Soma, concatenação, inteiros e matrizes
  - Ortogonalidade
    - Conjunto relativamente pequeno de construções primitivas pode ser combinado em um número relativamente pequeno de maneiras de construir as estruturas de controle e de dados

      Tipos de dados (int, real) vs. Operadores (+,-,*,/)

Legibilidade

- Instruções de controle
  - Decadas 50 e 60: má legibilidade causada pelas limitadas instruções de controle
  - Uso da instrução GO TO vs. Uso do WHILE, FOR, REPEAT
  - FORTRAN IV, BASIC

```
! this program calculates Bending Moments, and Shear
! at different distances along a uniform load
! the distance will be denoted "x",
! it is always less than the span "s".

      program BEAM

      integer span, l, v
      real load, r1, M, F

      write(*,30) 'BEAM prog to calculate shear & Moment along a beam'
 30   format (//,3x,A,//)
! getting input values
      print *,'Enter the value of span: '
      read  *, span
      print *,'Enter the value of uniform load q: '
      read  *, load

! calculate the reaction
      r1 = load*span/2.0

! calculating segment of distance
      v = span/40

      do 120 i= 1,40
      l = i * v

! Bending Moment & Shear
      M = r1*l - load*l**2/2.0
      F = r1 - load * l

      write (*,100) 'Bending Moment at x(',l,') is: ', M
 100  format (/,A,I2,A,5F8.2)

      write (*,110) 'Shear at x(',l,') is ', F
 110  format (/,A,I2,A,5F8.2)

 120  continue
!BYE
      stop
      end
```

```
10 'This will draw 5 spheres
20 GOTO 160
50 IF VERT GOTO 100
60 CIRCLE (X,Y),R,C,,,.07
70 FOR I = 1 TO 5
80 CIRCLE (X,Y),R,C,,,I*.2:NEXT I
90 IF VERT THEN RETURN
100 CIRCLE (X,Y),R,C,,,1.3
110 CIRCLE (X,Y),R,C,,,1.9
120 CIRCLE (X,Y),R,C,,,3.6
130 CIRCLE (X,Y),R,C,,,9.8
140 IF VERT GOTO 60
150 RETURN
160 CLS:SCREEN 1:COLOR 0,1:KEY OFF:VERT=0
170 X=160:Y=100:C=1:R=50:GOSUB 50
180 X=30:Y=30:C=2:R=30:GOSUB 50
190 X=30:Y=169:GOSUB 50
200 X=289:Y=30:GOSUB 50
210 X=289:Y=169:GOSUB 50
220 LINE (30,30)-(289,169),1
230 LINE (30,169)-(289,30),1
240 LINE (30,169)-(289,30),1,B
250 Z$=INKEY$: IF Z$="" THEN 250
RUN
```

Legibilidade

- Tipos de Dados e Estruturas
  - Facilidade para definir Tipos de dados e Estruturas de dados

```
          Fteste = 1
   Final_do_teste = 1
Final_do_teste = true
```

```
CHARACTER (LEN=30) NOME
```

```
char nome[30];
```

Legibilidade

- Considerações sobre a sintaxe
  - Formas identificadoras
    - Restringir os identificadores a tamanhos muito pequenos
    - FORTRAN 77: max 6 caracteres
    - BASIC 78: uma única letra ou 1 letra + 1 dígito
  - Palavras Especiais
    - Pares: **begin-end**, { … }, **repeat…until**
    - Significado: REPEAT = repetir, FOR…DO = para… fazer
    - Palavras-chave: não podem ser usadas como nome de variáveis
  - Forma e significado
    - Aparência indica finalidade: public, private, static, common
    - Problemas com o **shell** do UNIX: `find / -name –print | grep`

Capacidade de Escrita

- Medida de quão facilmente uma linguagem pode ser usada para criar programas para um domínio de problema escolhido

```
for i=1 to m                                    PASCAL
   for j = 1 to n
        begin
          soma = 0
          for k=1 to p
              soma = soma +  A (i,k) * B(k,i)
          C(i,j) = soma
        end
```

MATLAB:

```
C = A * B
```

Capacidade de Escrita

- **Simplicidade e Ortogonalidade**
  - Grande número de construções: uso abusivo de alguns, uso inadequado e desuso de outros
  - Erros ao escrever programas podem não ser detectados, uma vez que quase todas as combinações de primitivas são legais

- **Expressividade**
  - Operadores poderosos
    - Em C: count++ count = count + 1
    - Em Pascal: Laços de contagem: FOR melhor que WHILE

Capacidade de Escrita

- **Suporte para Abstração**
  - Abstração: capacidade de *definir*
  - Abstração de Processo: uso de *sub-programas*
  - Abstração de Dados: uso de tipos diferentes (árvores para arrays, classes, objetos, etc.)

$$x = \frac{-b \pm \sqrt{b^2 - 4.a.c}}{2.a}$$

```
X = bhaskara(A,B,C)
```

```
V1 = b*b - 4*a*c
V2 = sqrt(V1)
V3 = -b + V2
V4 = -b - V2

Vfinal = V3 /(2*a)
Wfinal = V4/ (2*a)
```

Confiabilidade

- **Um programa é confiável se ele se comportar de acordo com suas especificações sob todas as condições**
- **Recursos que indicam confiabilidade:**
  - Verificação de Tipos
    - Verificar se existem erros de tipo: na compilação ou na execução
  - Manipulação de Exceções
    - Capacidade de interceptar erros em tempo de execução
  - Aliasing
    - Ter dois ou mais métodos, ou nomes, para fazer referência à mesma célula da memória. É um recurso perigoso. Ex. Ponteiros
  - Legibilidade e capacidade de Escrita
    - Programas de difícil leitura complicam também sua escrita e sua modificação.

Custo

- **Custo de treinamento de programadores**
  - Simplicidade e ortogonalidade
  - Experiência dos programadores
- **Custo para escrever programas**
  - Reduzidos em um Ambiente de Programação
- **Custo para compilar programas**
  - Tipo de compilador
- **Custo para executar programas**
  - Verificação de tipos: mais lento
  - Compilação-execução: otimização de métodos
    - Laboratório de estudantes: pouca ou nenhuma otimização
    - Execução de programas completos: otimizar o código

Custo

- **Custo do sistema de implementação**
  - Compiladores/interpretadores disponíveis
  - Sistema de implementação (software) caro
  - Sistema de hardware caro
- **Custo da má confiabilidade**
  - Software para sistemas críticos: usina de energia nuclear, raios X, aeroportos, eletricidade, etc.
  - Linguagens Formais
- **Custo de manutenção**
  - Crise de software
  - Software de vida longa - manutenção: 2 a 4 vezes os custos de desenvolvimento

Portabilidade

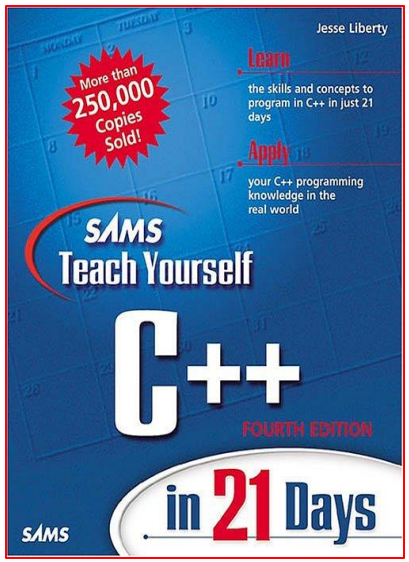
- **Definição:**
  - Facilidade com que os programas podem ser mudados de uma implementação para outra: DOS, Windows, UNIX, System X, VMS
- **Influenciado pelo grau de padronização**
  - BASIC: não padronizadas
  - FORTRAN: 77, 90, 95
  - C++: ainda não padronizada (Comissão trabalhando)
- **Linguagens portáveis**
  - LaTeX
  - HTML
  - Linguagens Formais

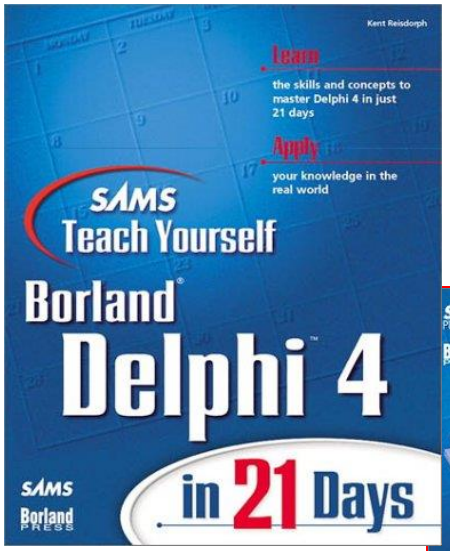
Qualidades de uma LP

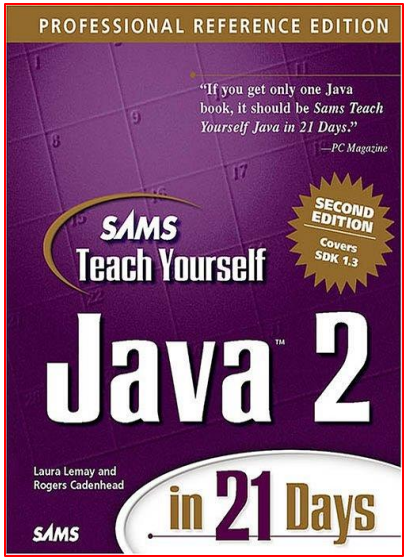
- Funções definíveis
- Tamanho do código objeto compilado
- Eficiência na execução
- Transparência para:
  - Código de máquina
  - Eficiência
  - Definições matemáticas
- Tipos de dados definíveis
- Documentabilidade própria, legibilidade
- Código fonte conciso
- Extensibilidade
- Primitivas
- Modificabilidade de programas
- Manutenibilidade de programas
- Reusabilidade de programas
- Facilidades de controle de versões
- Suporte para programação em equipe
- Ferramentas de depuração
- Aspectos de Modularidade
- Interfaces para outras linguagens e sistemas
- Habilidades interativas
- Habilidades para Input/Output
- Estruturas de controle
- Sintaxe
- Tipografia-Display
- Editores especializados para programação

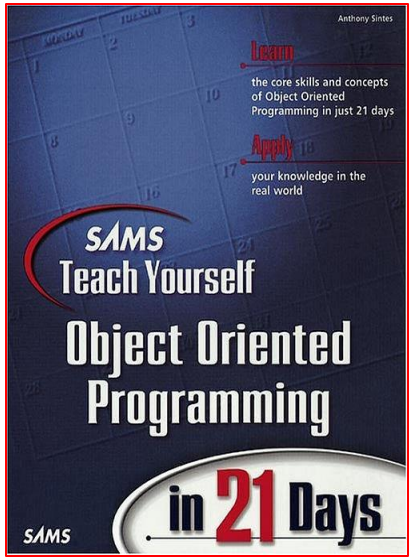
O valor de uma Linguagem de Programação

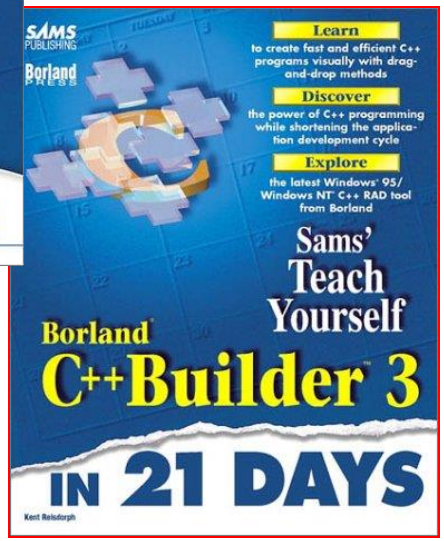
“Aprenda em 21 dias”



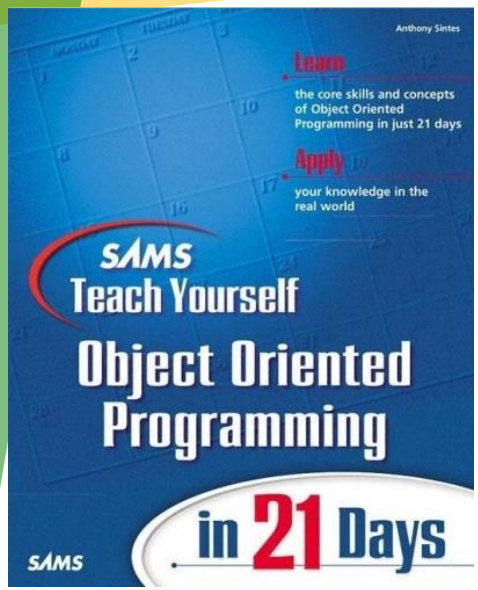


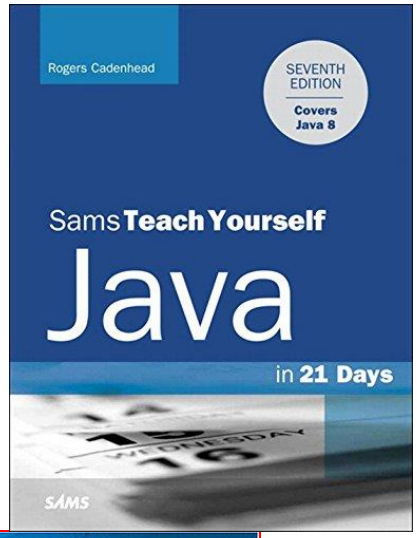


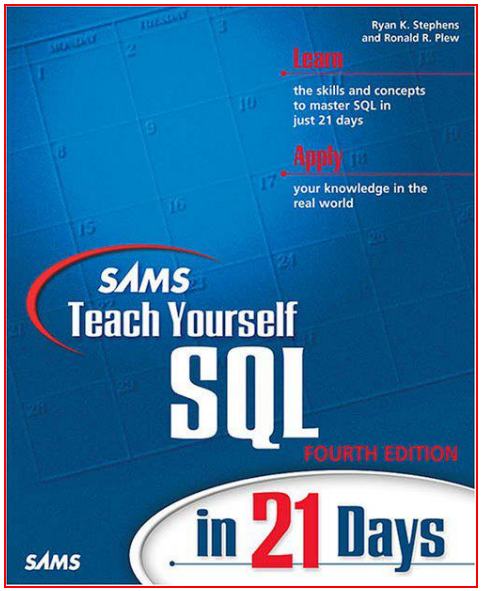


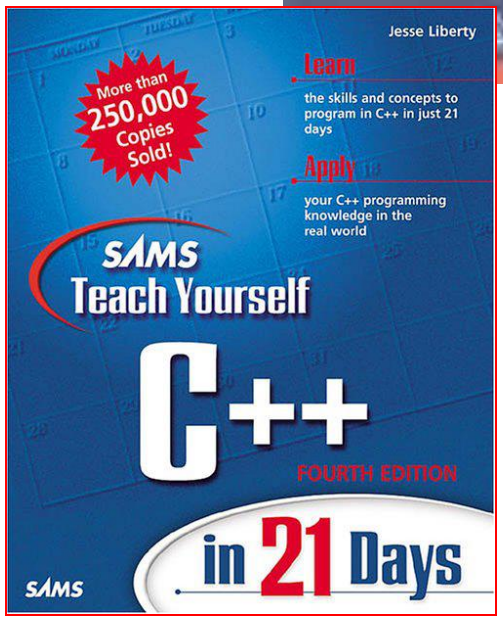


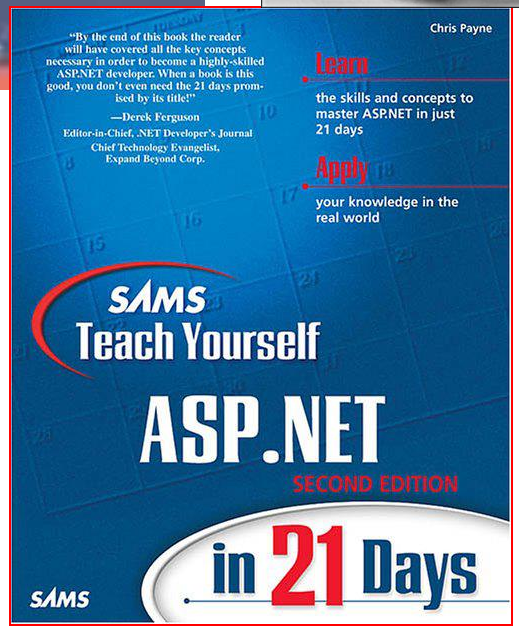
“Aprenda em 21 dias”



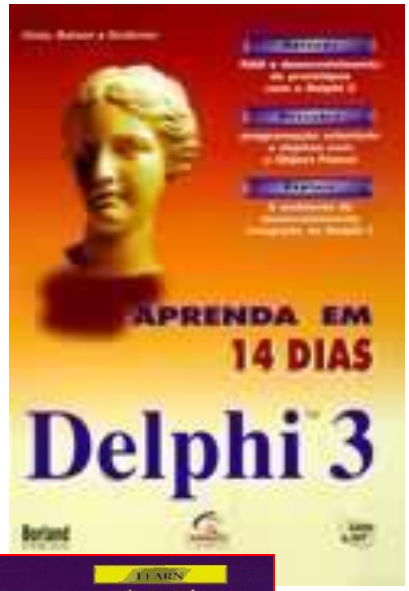


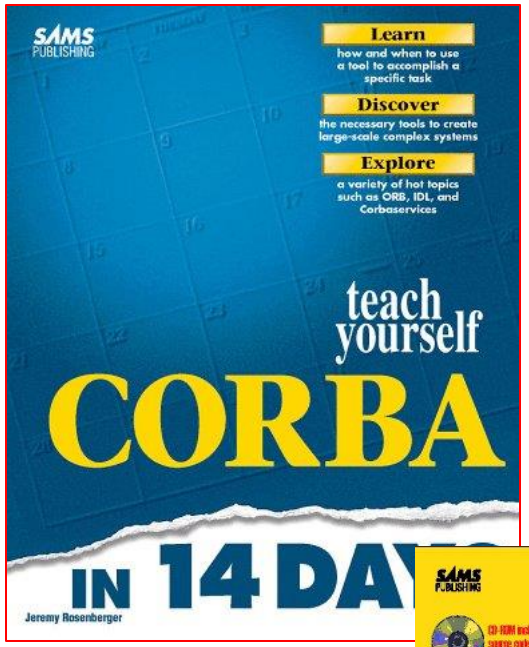


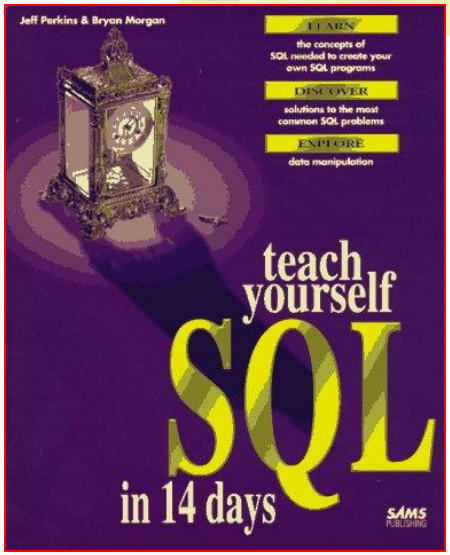


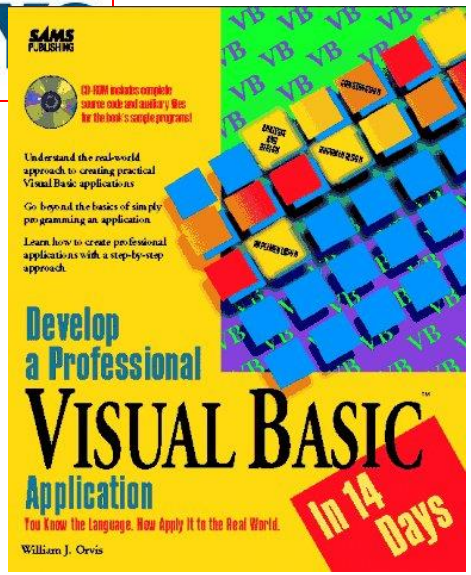


“Aprenda em 14 dias”

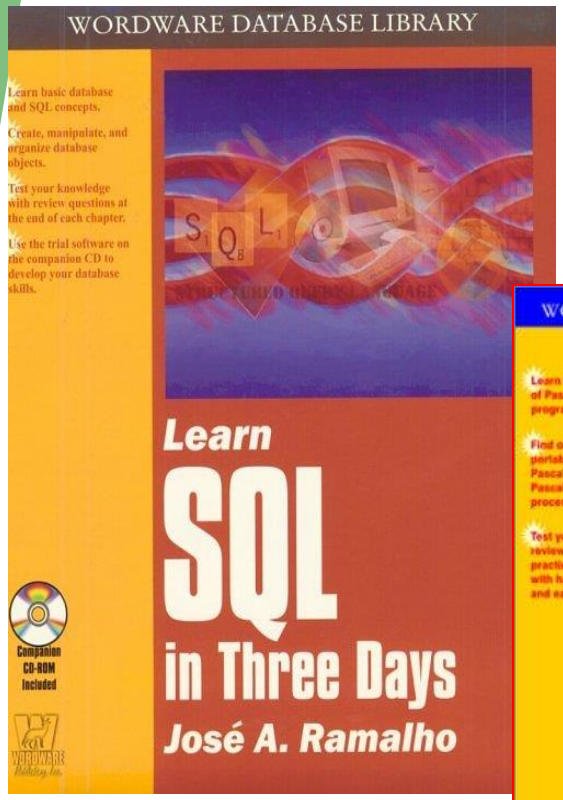


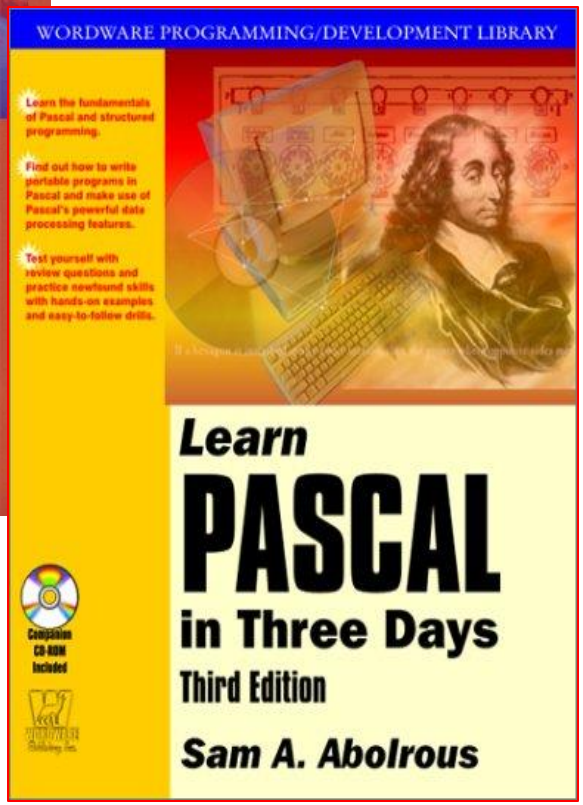


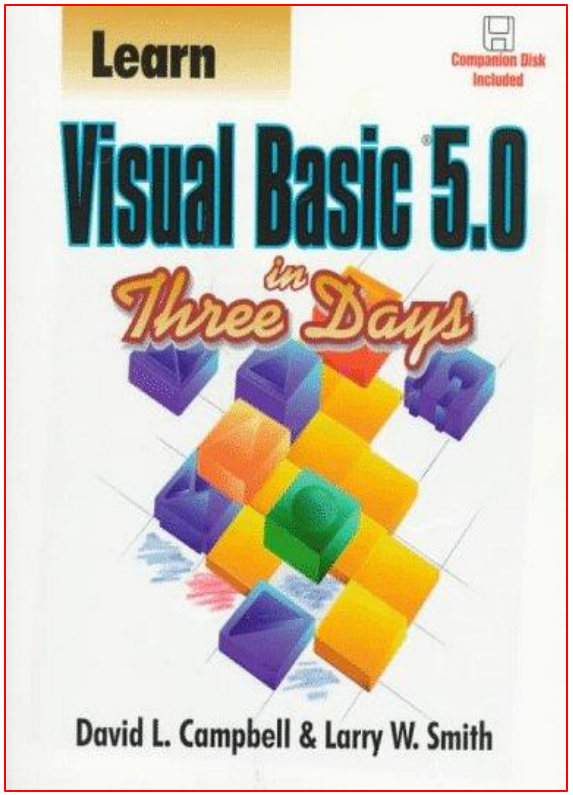




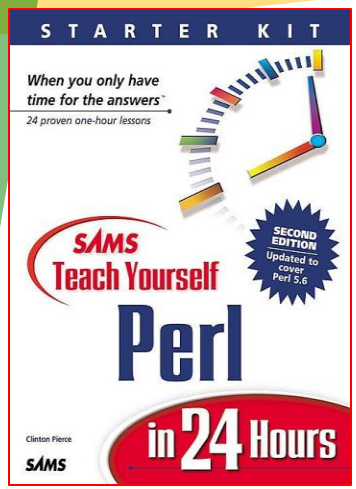
“Aprenda em 3 dias”

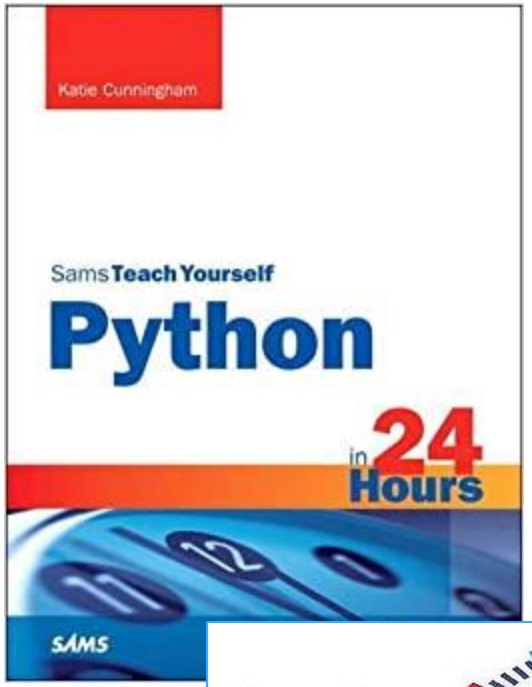


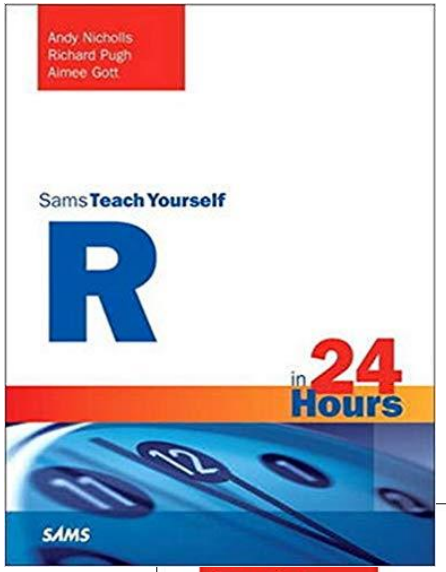


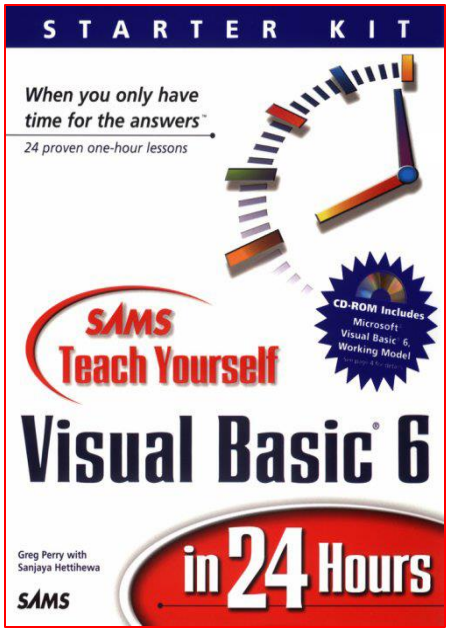


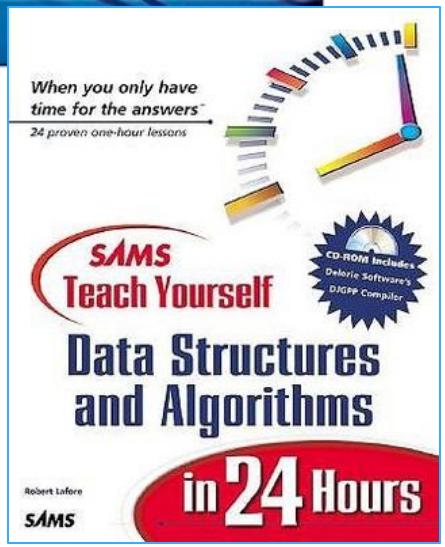
“Aprenda em 24 hrs”

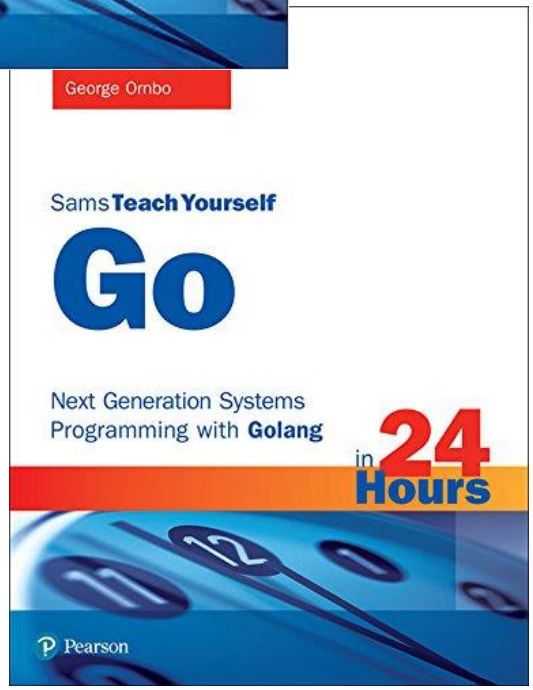




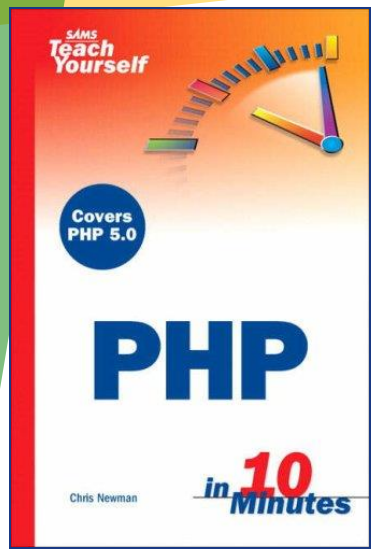


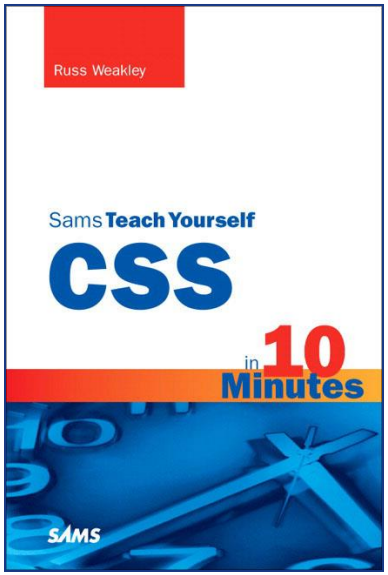


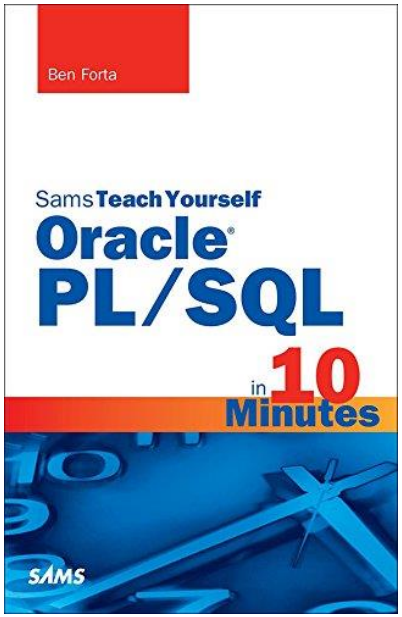


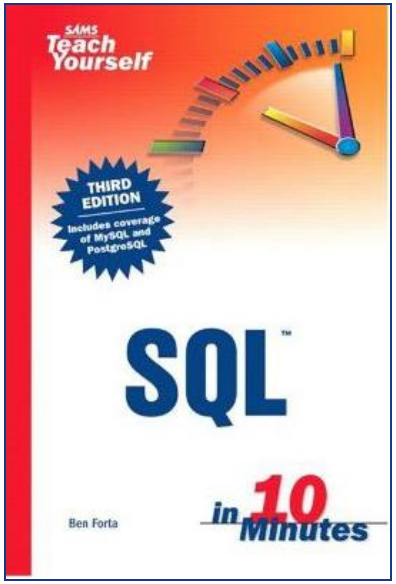


“Aprenda em 10 min.”









Prof. Dr. Ausberto S. Castro Vera
Ciência da Computação
UENF-CCT-LCMAT
Campos, RJ

ascv@uenf.br
ausberto.castro@gmail.com