



UNIVERSIDADE  
FEDERAL DO CEARÁ  
CAMPUS DE RUSSAS



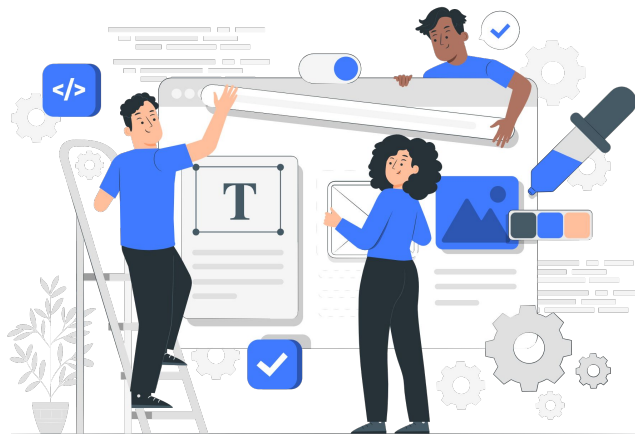
**RUS0059 - Linguagens de Programação**

# Projetando uma linguagem

Profa. Elanne Mendes

# Projetando uma Linguagem

- Hoje vamos parar de pensar um pouco como **programadores** e pensar como projetistas.



# Projetando uma Linguagem

**Suponha que vocês irão criar uma nova linguagem ou então criar uma versão melhorada de uma linguagem já existente.**

- **Liste 3 fatores** que vocês acham importantes e que devem ser levados em consideração durante o projeto;
- **Descreva em poucas palavras** por que estes fatores devem ser levados em consideração;

# Projetando uma Linguagem

## Exemplo: Suporte


Uma boa linguagem de programação deve ser facilmente **acessível** por alguém que queria aprendê-la e instalá-la no seu próprio computador. Tornar o seus compiladores de domínio público, disponibilizar cursos, livros-texto, tutoriais são vantagens que ajudam a preservar e estender a vitalidade de uma linguagem.



# Projetando uma Linguagem

**Como vocês notaram projeto de uma linguagem é um enorme desafio;**

Para atingirem seu objetivo os projetistas devem trabalhar dentro de diversas restrições práticas e adotar objetivos específicos que se combinam para fornecer um foco a esse desafio.



```
Hello World.
```

## Restrições de Projeto

**Os seguintes elementos de configurações computacionais impõem importantes restrições:**

- Arquitetura
- Configuração técnica
- Padrões
- Sistemas Legados

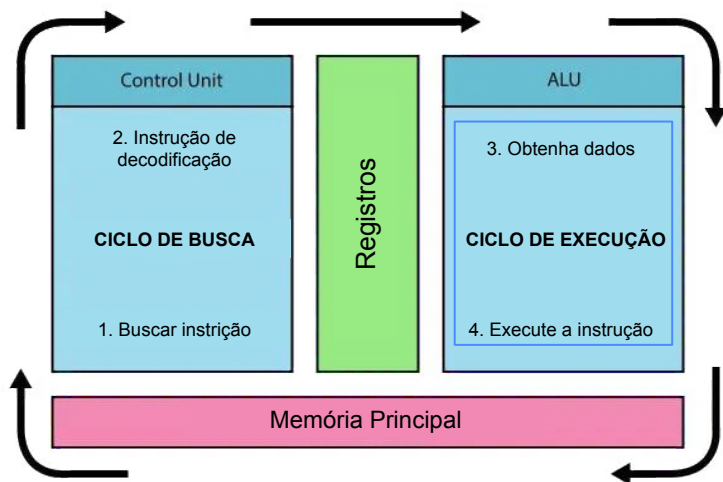
# Arquitetura

## Linguagens de programação são projetadas para computadores

- **Fator bom:** Uma linguagem bem projetada e implementada pode melhorar muito a utilidade do computador em um domínio de aplicação;
- **Fator ruim:** A maioria dos projetos de computadores são limitados pela arquitetura clássica de von Neumann;

# Arquitetura

## Arquitetura de Von Neumann



Inicializa o contador de programa

**Repita** sempre

- carregue a instrução apontada pelo contador de programa
- incremente o contador de programa
- decodifique a instrução
- execute a instrução

**Fim do Repete**



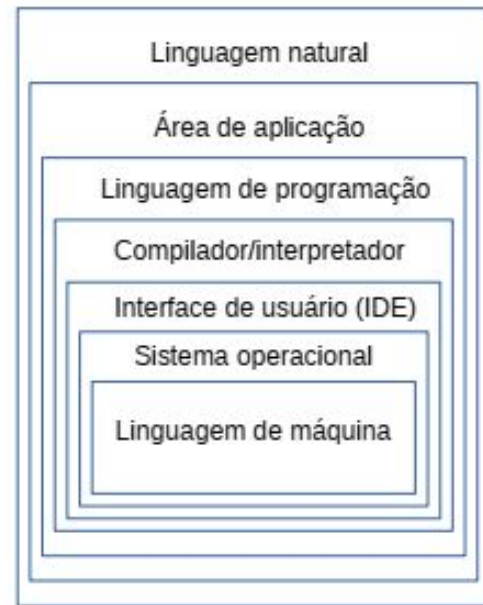
# Arquitetura

## **Mas qual é o problema disso?**

Como todos os computadores atuais seguem a arquitetura de von Neumann... quando você cria uma LP que não está de acordo com essa arquitetura... você terá que criar a arquitetura também!

# Configuração Técnica

- Linguagens devem satisfazer as restrições impostas pelas configurações técnicas nas quais são usadas.
- Níveis de abstração na computação.



Níveis de abstração na Computação

# Configuração Técnica

## Fortran é usado em diferentes plataformas

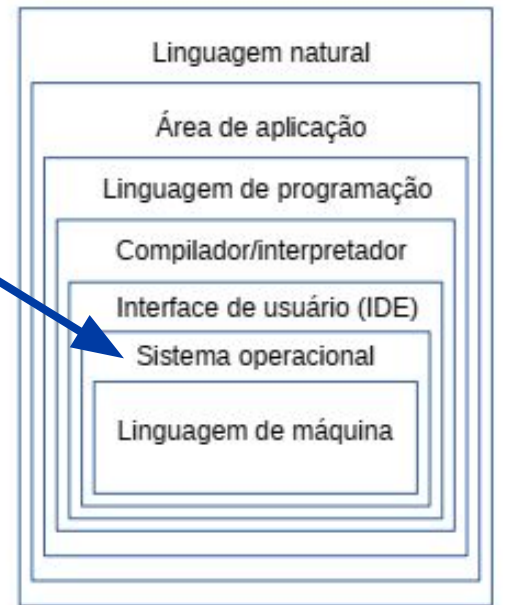
em diferentes compiladores

para se adaptar às necessidades de programadores científicos.

Esses programadores trabalham em diversas profissões

Que usam seus próprios projetos e ferramentas

E suas linguagens naturais para comunicar-se entre eles.



Níveis de abstração na Computação

# Configuração Técnica

Fortran é usado em diferentes plataformas

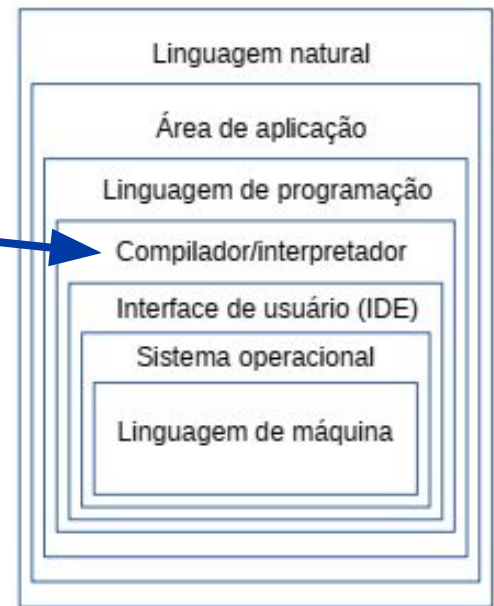
**em diferentes compiladores**

para se adaptar às necessidades de programadores científicos.

Esses programadores trabalham em diversas profissões

Que usam seus próprios projetos e ferramentas

E suas linguagens naturais para comunicar-se entre eles.



Níveis de abstração na Computação

# Configuração Técnica

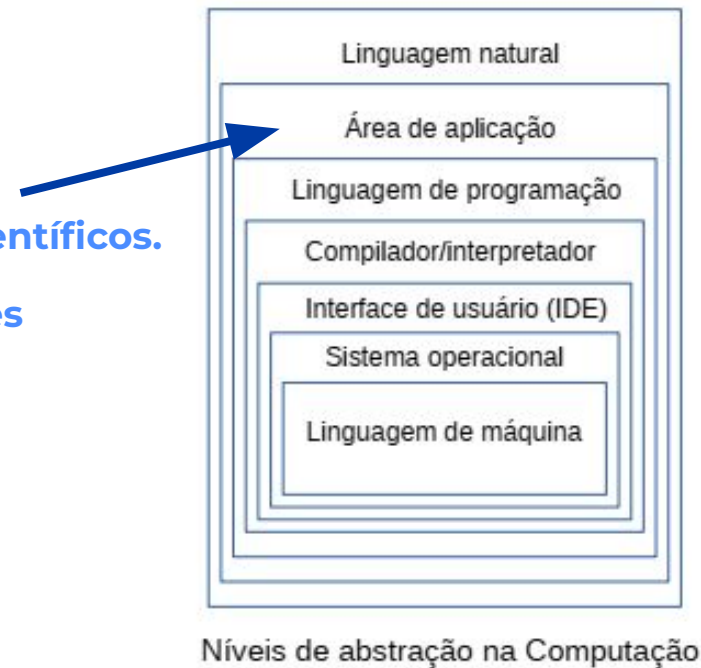
Fortran é usado em diferentes plataformas  
em diferentes compiladores

**para se adaptar às necessidades de programadores científicos.**

**Esses programadores trabalham em diversas profissões**

Que usam seus próprios projetos e ferramentas

E suas linguagens naturais para comunicar-se entre eles.



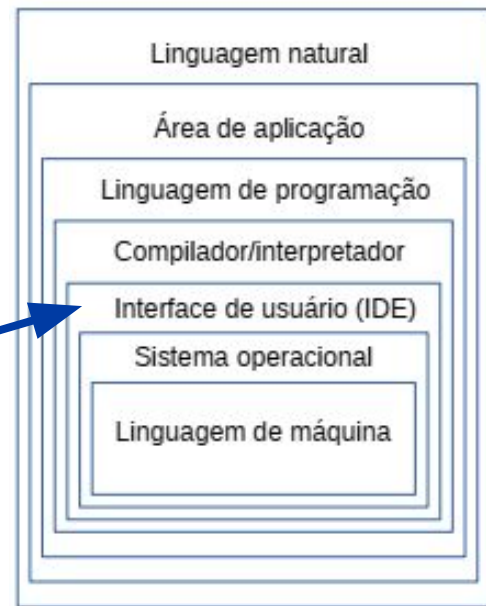
# Configuração Técnica

Fortran é usado em diferentes plataformas  
em diferentes compiladores  
para se adaptar às necessidades de programadores científicos.

Esses programadores trabalham em diversas profissões

**Que usam seus próprios projetos e ferramentas**

E suas linguagens naturais para comunicar-se entre eles.

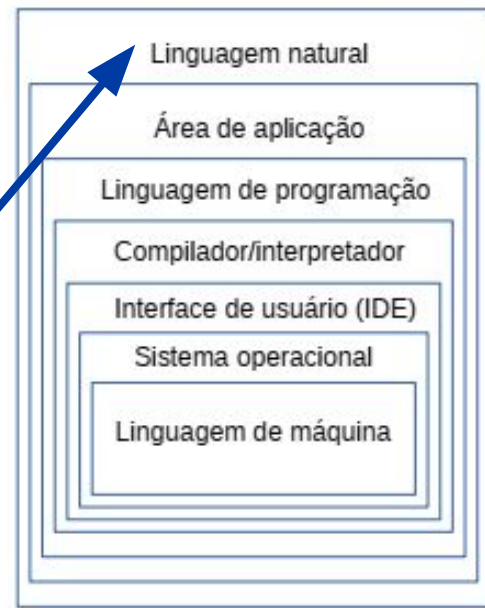


Níveis de abstração na Computação

# Configuração Técnica

Fortran é usado em diferentes plataformas  
em diferentes compiladores  
para se adaptar às necessidades de programadores científicos.  
Esses programadores trabalham em diversas profissões  
Que usam seus próprios projetos e ferramentas

**E suas linguagens naturais para comunicar-se entre eles.**



Níveis de abstração na Computação

# Configuração Técnica

Então sua linguagem tem imposta **restrições** como:

- Área de aplicação
- Sistema operacional
- IDE (Integrated Development Environment)
- Outras preferências de uma determinada comunidade de programadores



# Configuração Técnica

Então sua linguagem tem imposta **restrições** como:

- Área de aplicação
- Sistema operacional
- IDE (Integrated Development Environment)
- Outras preferências de uma determinada comunidade de programadores

<i>Linguagem</i>	<i>Propósito</i>
Ada	Ser útil para todas as aplicações suportadas pelo Departamento de Defesa dos Estados Unidos
Cobol	Suportar todas as aplicações orientadas a negócio.
Prolog	Processamento de linguagem natural, provas de teoremas e sistemas especializados.
C	Programação de sistemas operacionais

# Padrões

A **padronização** de uma linguagem visa a estabilidade em diferentes plataformas e grupos de programação (**portabilidade**);

As duas maiores organizações que supervisionam e mantêm os padrões de LP são:

- **American National Standards Institute (ANSI)**
- **International Standards Organization (ISO)**

O processo de padronização é **complexo** e **lento** e envolve uma **definição volumosa** da **linguagem-padrão** como resultado.

# Padrões

Algumas padronizações:

- ANSI/ISO Cobol (2002)
- ISO Fortran (2004)
- ISO Haskell (1998)
- ISO Prolog (2000)
- ANSI/ISO C (1999)
- ANSI/ISO C++ (2003)
- ANSI/ISO Ada (2005)
- ANSI Smaltalk (2002)
- ISO Pascal (1990)

## Padrões

**Mas afinal, a padronização é bom ou ruim para uma LP?**

Resposta: a padronização LP é uma **influência negativa** porque **inibe inovação** no projeto de linguagens, ou seja, versões-padrão tendem a **durar muito tempo**, perpetuando **características pobres** das linguagens.

Os padrões ISO e ANSI são revisados a cada **5 anos**, o que fornece certa proteção contra a obsolescência prolongada.

# Sistemas Legados

Para suportar sistemas escritos em versões antigas, **versões mais novas devem ser compatíveis com suas predecessoras**.

Como recursos **não são retirados** para não prejudicar **integridade**, linguagens antigas tornam-se sobrecarregadas.

# Sistemas Legados

Para suportar sistemas escritos em versões antigas, **versões mais novas devem ser compatíveis com suas predecessoras**.

Como recursos **não são retirados** para não prejudicar **integridade**, linguagens antigas tornam-se sobrecarregadas.



**1972**

# Sistemas Legados

Para suportar sistemas escritos em versões antigas, **versões mais novas devem ser compatíveis com suas predecessoras**.

Como recursos **não são retirados** para não prejudicar **integridade**, linguagens antigas tornam-se sobrecarregadas.



1972



1985

# Sistemas Legados

Para suportar sistemas escritos em versões antigas, **versões mais novas devem ser compatíveis com suas predecessoras**.

Como recursos **não são retirados** para não prejudicar **integridade**, linguagens antigas tornam-se sobrecarregadas.



1972



1985



1995



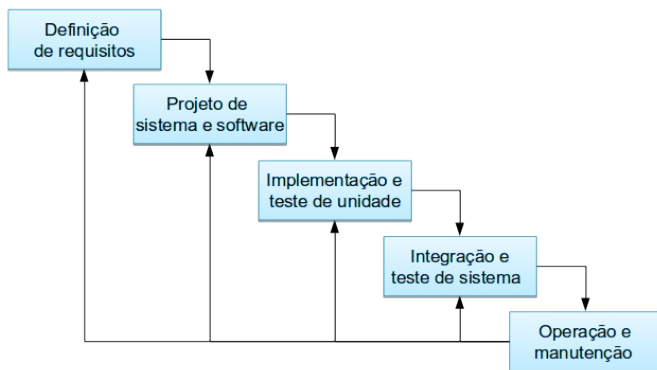
# O que faz uma linguagem ser considerada boa?

# O que faz uma linguagem ser considerada boa?

- Simplicidade e Legibilidade
- Ortogonalidade
- Confiabilidade
- Suporte
- Abstração
- Custos

# Simplicidade e Legibilidade

- Antes dos anos 70, a principal característica positiva de uma linguagem de programação era a eficiência.
- Entretanto, o conceito de ciclo de software foi desenvolvido e notou-se que a manutenção era a maior etapa.
- Manutenção requer constante leitura.



*“Qualquer tolo pode escrever um código que um computador pode entender. Bons programadores escrevem código que humanos podem entender.”*

*Kent Beck*



# Simplicidade e Legibilidade

- Uma linguagem com um grande número de componentes básicos é mais difícil de ser aprendida.
  - Programadores tendem a aprender um subconjunto de uma linguagem e ignorar seus outros recursos.
- Outra fator de dificuldade acontece quando há muitas maneiras de realizar uma mesma operação.



# Simplicidade e Legibilidade

- Uma linguagem com um grande número de componentes básicos é mais difícil de ser aprendida.
  - Programadores tendem a aprender um subconjunto de uma linguagem e ignorar seus outros recursos.
- Outra fator de dificuldade acontece quando há muitas maneiras de realizar uma mesma operação.



C

```
cont = cont + 1  
cont += 1  
cont++  
++cont
```

# Simplicidade e Legibilidade

- Formato dos identificadores e palavras chaves tem forte influência na legibilidade.

# Simplicidade e Legibilidade

- Formato dos identificadores e palavras chaves tem forte influência na legibilidade.

## *Pascal*

```
WHILE a < 10 DO  
BEGIN  
  IF a MOD 2 = 0 THEN  
  BEGIN  
    s := s + a;  
    c := c + 1;  
  END;  
  a := a + 1;  
END;
```

## *C*

```
while(a < 10){  
  if(a % 2 == 0){  
    s = s + a;  
    c = c + 1;  
  }  
  a = a + 1;  
}
```

# Simplicidade e Legibilidade

- Formato dos identificadores e palavras chaves tem forte influência na legibilidade.

## *Pascal*

```
WHILE a < 10 DO
BEGIN
  IF a MOD 2 = 0 THEN
  BEGIN
    s := s + a;
    c := c + 1;
  END;
  a := a + 1;
END;
```

## *C*

```
while(a < 10){
  if(a % 2 == 0){
    s = s + a;
    c = c + 1;
  }
  a = a + 1;
}
```

## *Ada*

```
while (a < 10) loop
  if (a mod 2 == 0) then
    s := s + a;
    c := c + 1;
  end if;
  a := a + 1;
end loop;
```



# Simplicidade e Legibilidade

- Formato dos identificadores e palavras chaves tem forte influência na legibilidade.

## *Pascal*

```
WHILE a < 10 DO
BEGIN
  IF a MOD 2 = 0 THEN
  BEGIN
    s := s + a;
    c := c + 1;
  END;
  a := a + 1;
END;
```

## *C*

```
while(a < 10){
    if(a % 2 == 0){
        s = s + a;
        c = c + 1;
    }
    a = a + 1;
}
```

## *Ada*

```
while (a < 10) loop
    if (a mod 2 == 0) then
        s := s + a;
        c := c + 1;
    end if;
    a := a + 1;
end loop;
```

## *Python*

```
while a < 10:
    if a % 2 == 0:
        s = s + a
        c = c + 1
    a = a + 1
```

# Confiabilidade

- O programa se comporta da **mesma forma** toda vez que ele é executado com os mesmos dados de entrada?
- Ele se comporta **da mesma forma** quando é executado em diferentes plataformas?



# Confiabilidade

- Uma linguagem confiável possui características como:
  - Checagem de tipos
  - Tratamento de exceções
  - Restringir o uso de apelidos
  - Evitar vazamento de memória
  - etc...



# Suporte

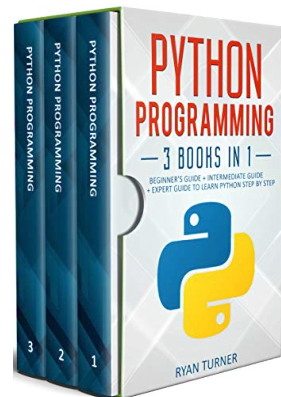
- Uma boa linguagem de programação deve ser **facilmente acessível** por alguém que queira aprendê-la e instalá-la no seu próprio computador.
- Vantagens que ajudam a preservar e estender a vitalidade de uma linguagem:
  - **Compiladores de domínio publico**
  - **Cursos**
  - **Livros-texto**
  - **Tutoriais**
  - **Grande número de pessoas familiarizadas com a linguagem**



**GitHub**



**stackoverflow**



# Abstração

- Habilidade de definir e usar estruturas ou operações complicadas de forma a permitir que muitos dos detalhes sejam ignorados.
- Essa característica permite que programadores reutilizem código sem a necessidade de entender completamente seu funcionamento e/ou reinventá-lo.
  - Exemplo: bibliotecas, framework, etc...



# Ortogonalidade

- Comandos e recursos são construídos sobre um conjunto pequeno e mutuamente independente de operações primitivas.

- Nos mainframes da IBM

```
A Reg1, memory_cell
```

```
AR Reg1, Reg2
```

- Nos minicomputadores da VAX

```
ADDL operand_1, operand_2
```

- **Quanto mais ortogonal uma linguagem, menos regras extras são necessárias para escrever programas corretos.**

# Ortogonalidade

**Muita ortogonalidade também pode levar a problemas, pois o grande número de combinações pode resultar em uma complexidade desnecessária**

- Em JavaScript é possível aplicar o operador **+** entre um número inteiro e uma string.

**3 + 4 + "5"**

**7 + "5"**

**"75"**

# Custos

- Diversos fatores podem influenciar na usabilidade e sucesso de uma linguagem de programação, dentre eles, as suas limitações:
  - Treinamento dos programadores
  - Criação do software
  - Compilação / Interpretação
  - Tempo de Execução
  - Tempo de Compilação
  - Baixa Confiança
  - Manutenção



# Projetando uma Linguagem

Como pode-se notar as ideias de projeto de linguagens são em geral **intuitivas** para quem já é um programador.



A solid blue vertical bar is positioned on the far left side of the image, extending from the top to the bottom.

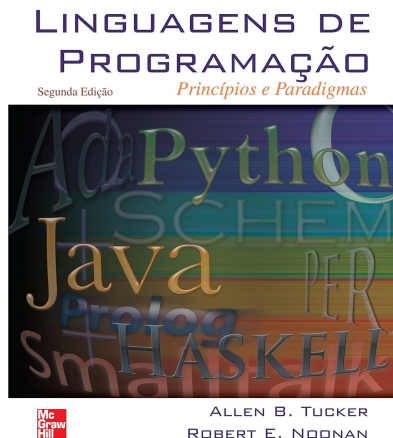
# Atividade

# Volte ao trabalho passado...

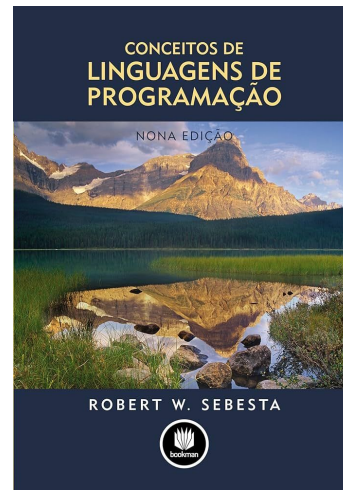
Suponha que vocês irão criar uma nova linguagem ou então criar uma versão melhorada de uma linguagem já existente.

- **Liste 3 fatores** que vocês acham importantes e que devem ser levados em consideração durante o projeto;
- **Descreva em poucas palavras** por que estes fatores devem ser levados em consideração;
- Separe os fatores escolhidos indicando se são **restrições práticas** ou **objetivos específicos**;
  - Caso seja uma **restrição prática** defina se é: **Arquitetura, Configuração técnica, Padrões ou Sistemas Legados**. Inclua mais uma restrição prática que você ache importante, além das já descritas e descreva o motivo da inclusão;
  - Caso seja um **objetivo específico** defina se é: **Simplicidade e legibilidade, Confiabilidade, Suporte, Abstração ou Ortogonalidade**. Inclua mais um objetivo específico que você ache importante, além dos já descritos e descreva o motivo da inclusão;

# Bibliografia recomendada



- TUCKER, Allen B.; NOONAN, Robert. Linguagens de programação: princípios e paradigmas. 2. ed. São Paulo: McGraw-Hill, 2008.



- SEBESTA, Robert W. Conceitos de linguagens de programação. 9. ed. Porto Alegre: Bookman, 2011.

# Próxima aula...

- **Implementação de uma linguagem**

# Obrigada!

**Profa. Elanne Mendes**

elanne@ufc.br



UNIVERSIDADE  
FEDERAL DO CEARÁ  
CAMPUS DE RUSSAS