# Pré-condições e Pós-condições

- Nesta aula são descritas **pré-condições** e **pós-condições**
- Elas permitem que um programador especifique o que um procedimento deve realizar

**Algoritmos e Estruturas de Dados I**

**José Augusto Baranauskas**
**Departamento de Física e Matemática – FFCLRP-USP**

**augusto@usp.br**
**http://dfm.ffclrp.usp.br/~augusto**

# Pré-condições e Pós-condições

- Freqüentemente um programador necessita comunicar precisamente o que um procedimento deve realizar, sem qualquer indicação de como esta tarefa deve ser realizada
- Você pode imaginar uma situação onde isto pode ocorrer?

# Exemplo

- Você é o chefe de um grupo de programadores e você deseja que um dos seus programadores escreva um procedimento como parte de um projeto

# O que são Pré-condições e Pós-condições?

- Um forma de especificar tais requisitos é através de um par de declarações sobre o procedimento
- A declaração de **pré-condição** indica o que deve ser verdade antes que o procedimento seja chamado
- A declaração de **pós-condição** indica o que será verdade quando o procedimento terminar o seu trabalho

# Exemplo

```
void WriteSquareRoot(float Number)
// Pré-condição:  Number  >=  0
// Pós-condição:  A raiz quadrada de Number deve
// ser escrita
{

  ...
}
```

# Exemplo

```
void WriteSquareRoot(float Number)
// Pré-condição:  Number  >=  0
// Pós-condição:  A raiz quadrada de Number deve
// ser escrita
{
```

- A pré-condição e pós-condição aparecem como comentários no seu programa
- Elas são freqüentemente colocadas após o cabeçalho do procedimento

# Exemplo

```
void WriteSquareRoot(float Number)
// Pré-condição:  Number  >=  0
// Pós-condição:  A raiz quadrada de Number deve
// ser escrita
{
```

Neste exemplo, a pré-condição requer que

**Number >= 0**

seja verdade sempre que o procedimento for chamado

# Exemplo

Quais destas chamadas de procedimento respeitam a pré-condição?

```
WriteSquareRoot( -10 );
WriteSquareRoot( 0 );
WriteSquareRoot( 5.6 );
```

# Exemplo

Quais destas chamadas de procedimento respeitam a pré-condição?

```
WriteSquareRoot( -10 );
WriteSquareRoot( 0 );
WriteSquareRoot( 5.6 );
```

✔ ✔

A segunda e terceira chamadas estão corretas, uma vez que seus argumentos são maiores ou iguais a zero

# Exemplo

Quais destas chamadas de procedimento respeitam a pré-condição?

✘ WriteSquareRoot( -10 );
✔ WriteSquareRoot( 0 );
✔ WriteSquareRoot( 5.6 );

Mas a primeira chamada viola a pré-condição, uma vez que seu argumento é menor que zero

# Exemplo

```
void WriteSquareRoot(float Number)
// Pré-condição:  Number  >=  0
// Pós-condição:  A raiz quadrada de Number deve
// ser escrita
{
```

A pós-condição sempre indica qual trabalho deve ser realizado pelo procedimento. Neste caso, quando o procedimento retorna, a raiz quadrada de **Number** deve ser escrita

# Outro Exemplo

```
bool Vowel(char Letter)
//  Pré-condição:  Letter é uma letra maiúscula ou
//  uma letra minúscula (no intervalo 'A' .. 'Z' ou 'a' .. 'z')
//  Pós-condição:  O valor retornado pela função
//  é verdade se Letter é uma vogal; caso-contrário
//  o valor retornado pela função é falso
{

  ...
}
```

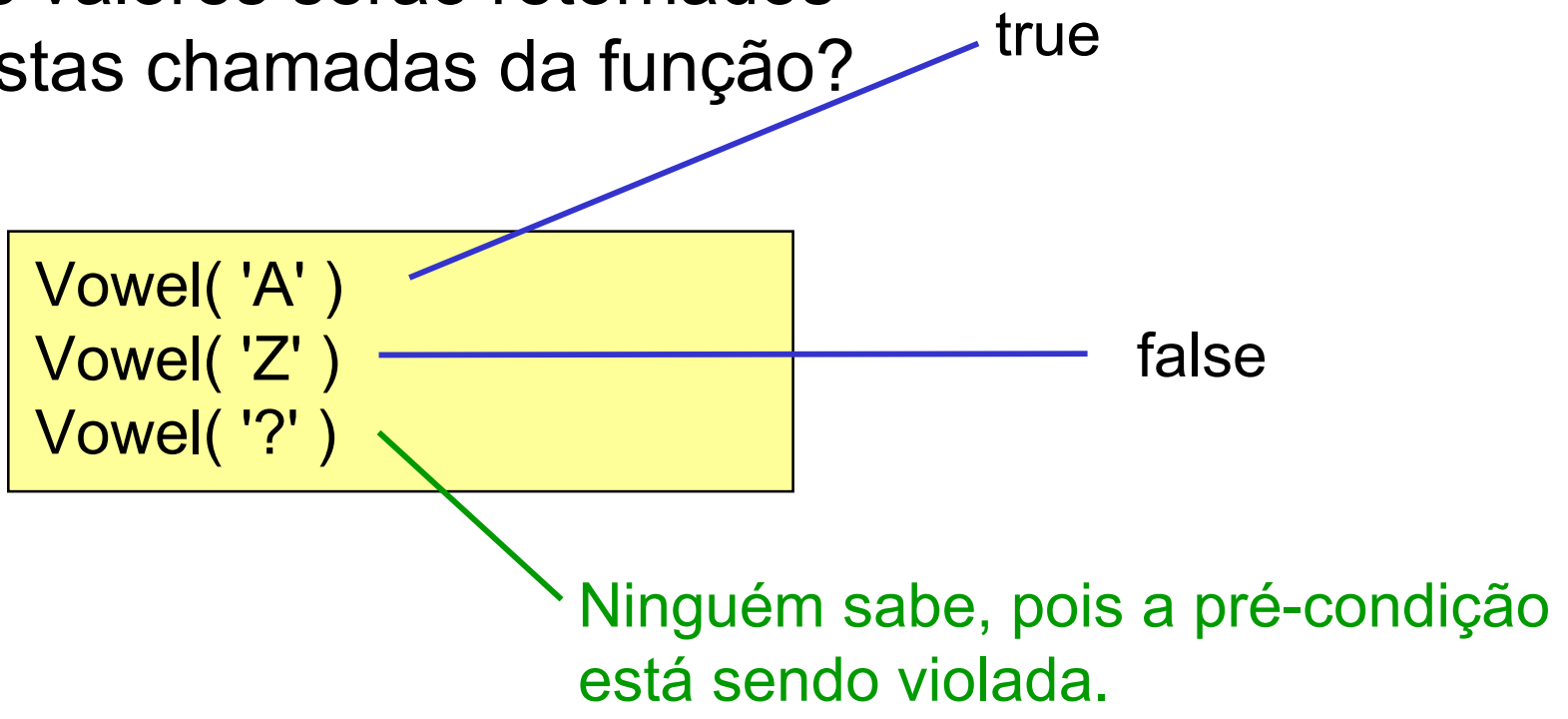
# Outro Exemplo

Quais valores serão retornados
por estas chamadas da função?

```
Vowel( 'A' )
Vowel( 'Z' )
Vowel( '?' )
```

# Outro Exemplo

Quais valores serão retornados por estas chamadas da função?



# Outro Exemplo

Quais valores serão retornados por estas chamadas da função?

```
Vowel( '?' )
```

A violação de uma pré-condição pode até mesmo resultar em efeitos inesperados

# Sempre certifique que a pré-condição seja válida...

- O programador que chama a função ou procedimento é responsável por **certificar que a pré-condição é válida** quando o procedimento é chamado

# ...então a pós-condição se torna verdade ao fim do procedimento

- O programador que escreve a função ou procedimento assume que a pré-condição é válida, e certifica que a pós-condição se torna verdade ao fim do procedimento

# Questão

- Suponha que você chame um procedimento, e você não se certifica que a pré-condição é verdadeira. Quem é responsável caso esta falha cause algum tipo de desastre?

❶ Você

❷ O programador que escreveu o procedimento

❸ Ninguém

# Questão

- Suponha que você chame um procedimento, e você não se certifica que a pré-condição é verdadeira. Quem é responsável caso esta falha cause algum tipo de desastre?

❶ Você

O programador que chama um procedimento é responsável por certificar que a pré-condição é válida

# Por outro lado, programadores cuidadosos também seguem as seguintes regras:

- Quando você escreve um procedimento, você deve detectar em quais condições uma pré-condição será violada
- Se você detectar que uma pré-condição foi violada, estão imprima uma mensagem de erro e aborte o programa...

# Por outro lado, programadores cuidadosos também seguem as seguintes regras:

- Quando você escreve um procedimento, você deve detectar em quais condições uma pré-condição será violada
- Se você detectar que uma pré-condição foi violada, estão imprima uma mensagem de erro e aborte o programa...
- ...em vez de causar um desastre

# Exemplo

```
void WriteSquareRoot(float Number)
// Pré-condição:  Number  >=  0
// Pós-condição:  A raiz quadrada de Number deve
// ser escrita
{
  if (Number < 0)              // Pré-condição falhou
  {  cout << ”ERROR: Number is negative”;
     abort();    // Isto pára um programa em C++
  }

  ...
}
```

# Vantagens do Uso de Pré-condições e Pós-condições

- Resumidamente descreve o comportamento de uma função ou procedimento...
- ...sem entrar em detalhes de implementação
- Posteriormente, você pode re-implementar o procedimento de uma nova forma...
- ...entretanto os programas (os quais dependem exclusivamente da pré-condição/pós-condição) continuarão a funcionar sem alterações

# Resumo

## Pré-condição

- O programador que chama o procedimento certifica que a pré-condição é válida
- O programador que escreve um procedimento assume que a pré-condição é verdade quando o procedimento inicia sua execução

## Pós-condição

- O programador que escreve um procedimento certifica que a pós-condição é verdadeira ao término do procedimento

Translation to portuguese by Prof. Maria Carolina Monard, ICMC-USP.
Modifications for C++ language by Prof. José Augusto Baranauskas, FFCLRP-USP, 2005.