

## Aula 25

### Argumentação da Corretude 2

Alessandro Garcia  
LES/DI/PUC-Rio  
Junho 2009


#### Aula passada... Argumentação de um passo de execução

- A notação:  $AE \oplus P \Rightarrow AS$  significa:
  - A expressão resulta em ~~TRUE~~ se for válida a assertiva de entrada  $AE$  e a execução do fragmento de código  $P$  implicar a validade da assertiva de saída  $AS$
- Como ler a expressão:  $A \oplus P \Rightarrow B$ 
  - $\oplus$  significa: **a execução de** (um fragmento de código)
  - dada a validade de  $A$  a execução do fragmento  $P$  implica a validade de  $B$

Exemplo: Fragmento P pode ser...

1. *corpo da função*: `AbrirArquivo( char * NomeArq , tpModo Modo ) => FILE* pArq , tpCondRet CondRet`
2. *qualquer bloco de código em* `AbrirArquivo`

**Especificação**



Laboratório de Engenharia de Software


- Objetivo dessa aula
  - Apresentar as técnicas de argumentação para repetições
- Referência básica:
  - Capítulo 13

Junho 2009

Alessandro Garcia - LES/DI/PUC-Rio

3 / 36

**Sumário**



Laboratório de Engenharia de Software


- Argumentação de repetições
- Assertiva invariante
- Proposições para argumentar a repetição
- Exemplo: intercalação de arquivos

Junho 2009

Alessandro Garcia - LES/DI/PUC-Rio

4 / 36

## Argumentação de Repetições



Laboratório de Engenharia de Software


- Como argumentar repetições?
  - devemos argumentar as  $n$  repetições?
  - tempo gasto não justificaria os benefícios
    - além disso, prova seria basicamente a mesma, com diferenças triviais nos valores sendo variados
- Que recurso matemático podemos usar para facilitar a argumentação de repetições?

Junho 2009

Alessandro Garcia - LES/DI/PUC-Rio

5 / 36

## Argumentação de Repetições



Laboratório de Engenharia de Software

- Como argumentar repetições?
  - devemos argumentar as  $n$  repetições?
  - tempo gasto não justificaria os benefícios
    - além disso, prova seria basicamente a mesma, com diferenças triviais nos valores sendo variados
- Que recurso matemático podemos usar para facilitar a argumentação de repetições?
  - **indução matemática**

Junho 2009

Alessandro Garcia - LES/DI/PUC-Rio

6 / 36

## Indução Matemática, recordação



Uma prova por indução consiste de três partes:

**Passo 1:** Especificação da **hipótese da indução**:

- $H(n) ::$  deveria valer para  $1, 2, \dots, n, \dots \infty$

**Passo 2:** Provar a validade de  $H(n)$  para caso base:

- $H(1)$

**Passo 3:** Passo indutivo:

- assumindo a validade para um  $n$  genérico, provar que  $H(n) \Rightarrow H(n+1)$ 
  - como vale para qualquer  $n$ , pode-se concluir que vale para todos os  $n$

Junho 2009

Alessandro Garcia - LES/DI/PUC-Rio

7 / 36

## Exemplo: Identidade de Gauss



- Queremos provar que:  $1 + 2 + \dots + m = \frac{m(m+1)}{2}$
- **Passo 1:** Especificação da hipótese da indução:
  - $H(m) = \frac{m(m+1)}{2}$
- **Passo 2:** Caso base:
  - $m = 1$
  - $H(1) = 1 = 1 * (1 + 1)/2 = 1$  🍷
- **Passo 3:** O passo indutivo:
  - Provar que  $H(m + 1) = (m + 1)*(m + 2)/2$

Junho 2009

Alessandro Garcia - LES/DI/PUC-Rio

8 / 36

Laboratório de Engenharia de Software

## Exemplo: Identidade de Gauss

LES

Assume-se  $H(m)$  e tenta provar  $H(m+1)$

- **Passo 3:** O passo indutivo (cont.):  $\rightarrow H(m) \Rightarrow H(m+1)$ 
  - Assumindo a hipótese da indução:  $1 + 2 + \dots + m = \frac{m(m+1)}{2}$
  - Provar que:  $H(m+1) = \frac{(m+1)(m+2)}{2}$

tirado da hipótese de indução

$$(1 + 2 + \dots + m) + (m+1) = \frac{m(m+1)}{2} + (m+1)$$

fatorando

$$\frac{m(m+1)}{2} + \frac{2(m+1)}{2} = \frac{(m+2)(m+1)}{2} = \frac{(m+1)(m+2)}{2}$$

Junho 2009

Alessandro Garcia - LES/DI/PUC-Rio

9 / 36

Laboratório de Engenharia de Software

## Argumentação de repetições

LES

- A argumentação segue as seguintes regras:
  - formular uma **hipótese de indução**
    - é estabelecida por uma **assertiva invariante** (AINV)
  - argumentar um a um a veracidade dos **casos especiais**
    - tipicamente terminar antes de concluir a primeira iteração é um caso especial, exemplos:
      - condição da repetição não é satisfeita na primeira iteração
      - uso de *break* ou *return*
    - *AS deve valer!*
  - argumentar a veracidade da **base de indução**
    - tipicamente executar completamente a primeira iteração
    - *AINV deve valer!*
  - assumindo a veracidade da hipótese de indução após uma k-ésima **iteração genérica**: argumentar que a execução do **corpo da repetição** restabelece a validade da hipótese de indução para a k+1 ésimas iteração
- Isso corresponde à **corretude parcial**

Junho 2009

Alessandro Garcia - LES/DI/PUC-Rio

10 / 36

## Argumentação de repetições



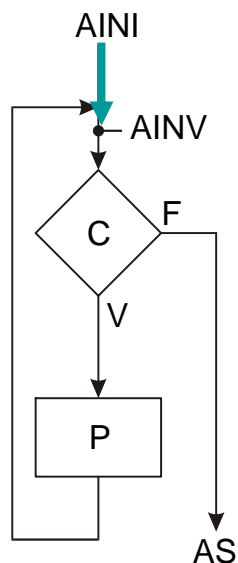
- Deve-se argumentar ainda que o fragmento de código pára
  - agora temos a **corretude total da repetição**
- A **assertiva invariante** (AINV) - hipótese de indução – deve satisfazer os seguintes critérios
  - a assertiva invariante deve inter-relacionar, pelo menos, todas as variáveis e estados que podem ser **alterados** no decorrer da repetição
  - as relações devem ser estabelecidas de modo que a (mesma) assertiva invariante **reflita o estado esperado da repetição para qualquer número** de 0 ou mais iterações

Junho 2009

Alessandro Garcia - LES/DI/PUC-Rio

11 / 36

## 6 proposições básicas para a repetição



Devem ser argumentadas as proposições:

1.  $AINI \Rightarrow AINV$

- a invariante vale ao iniciar


Junho 2009

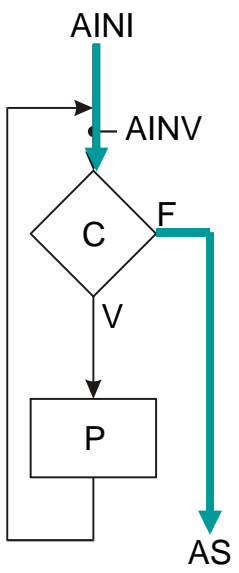
Alessandro Garcia - LES/DI/PUC-Rio

12 / 36

Laboratório de Engenharia de Software

## Proposições para a repetição






Devem ser argumentadas as proposições:

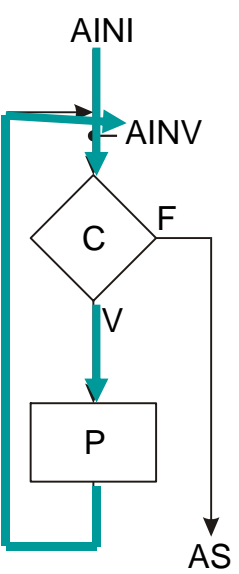
1.  $AINI \Rightarrow AINV$ 
  - a invariante vale ao iniciar
2.  $AINI \ \&\& \ (C == F) \Rightarrow AS$ 
  - executa corretamente 0 iterações

Junho 2009
Alessandro Garcia - LES/DI/PUC-Rio
13 / 36

Laboratório de Engenharia de Software

## Proposições para a repetição






Devem ser argumentadas as proposições:

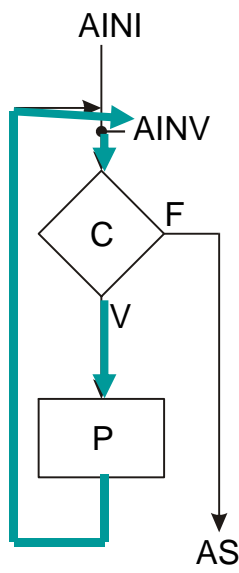
1.  $AINI \Rightarrow AINV$ 
  - a invariante vale ao iniciar
2.  $AINI \ \&\& \ (C == F) \Rightarrow AS$ 
  - executa corretamente 0 iterações
3.  $AINI \ \&\& \ (C == V) \oplus P \Rightarrow AINV$ 
  - executa corretamente a primeira iteração (base da indução)

Junho 2009
Alessandro Garcia - LES/DI/PUC-Rio
14 / 36

Laboratório de Engenharia de Software



## Proposições para a repetição



```

graph TD
    AINI --> AINV
    AINV --> C{C}
    C -- V --> P[P]
    P --> AINV
    C -- F --> AS[AS]


```

Devem ser argumentadas as proposições:

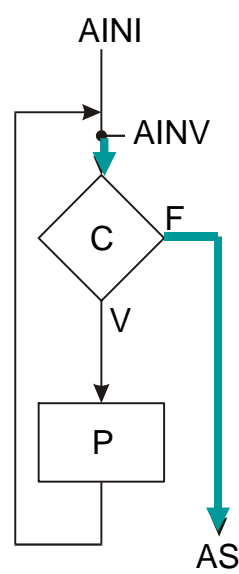
1.  $AINI \Rightarrow AINV$ 
  - a invariante vale ao iniciar
2.  $AINI \ \&\& \ (C == F) \Rightarrow AS$ 
  - executa corretamente 0 iterações
3.  $AINI \ \&\& \ (C == V) \oplus P \Rightarrow AINV$ 
  - executa corretamente a primeira iteração (base da indução)
4.  $AINV \ \&\& \ (C == V) \oplus P \Rightarrow AINV$ 
  - acrescenta corretamente mais uma iteração

Junho 2009
Alessandro Garcia - LES/DI/PUC-Rio
15 / 36

Laboratório de Engenharia de Software



## Proposições para a repetição



```

graph TD
    AINI --> AINV
    AINV --> C{C}
    C -- V --> P[P]
    P --> AINV
    C -- F --> AS[AS]

```


Devem ser argumentadas as proposições:

1.  $AINI \Rightarrow AINV$ 
  - a invariante vale ao iniciar
2.  $AINI \ \&\& \ (C == F) \Rightarrow AS$ 
  - executa corretamente 0 iterações
3.  $AINI \ \&\& \ (C == V) \oplus P \Rightarrow AINV$ 
  - executa corretamente a primeira iteração (base da indução)
4.  $AINV \ \&\& \ (C == V) \oplus P \Rightarrow AINV$ 
  - acrescenta corretamente mais uma iteração
5.  $AINV \ \&\& \ (C == F) \Rightarrow AS$ 
  - sai correto após  $n > 0$  iterações

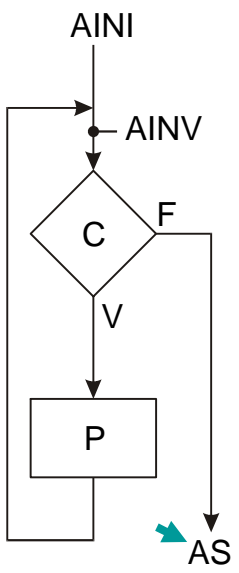
Junho 2009
Alessandro Garcia - LES/DI/PUC-Rio
16 / 36



Laboratório de Engenharia de Software



## Proposições para a repetição



```


graph TD
    AINI --> AINV
    AINV --> C{C}
    C -- F --> AS
    C -- V --> P[P]
    P --> AINV
    
```

Devem ser argumentadas as proposições:

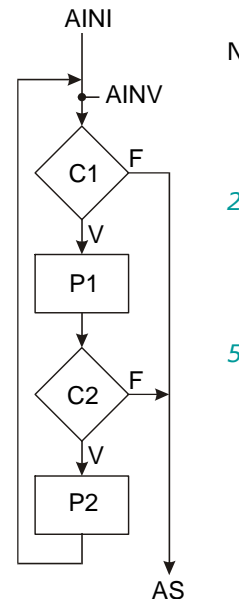
1.  $AINI \Rightarrow AINV$ 
  - a invariante vale ao iniciar
2.  $AINI \ \&\& \ (C == F) \Rightarrow AS$ 
  - executa corretamente 0 iterações
3.  $AINI \ \&\& \ (C == V) \oplus P \Rightarrow AINV$ 
  - executa corretamente a primeira iteração (base da indução)
4.  $AINV \ \&\& \ (C == V) \oplus P \Rightarrow AINV$ 
  - acrescenta corretamente mais uma iteração
5.  $AINV \ \&\& \ (C == F) \Rightarrow AS$ 
  - sai correto após  $n > 0$  iterações
6. *término*
  - a repetição pára depois de um número finito de iterações

Junho 2009
Alessandro Garcia - LES/DI/PUC-Rio
17 / 36

Laboratório de Engenharia de Software



## Proposições para a repetição



```

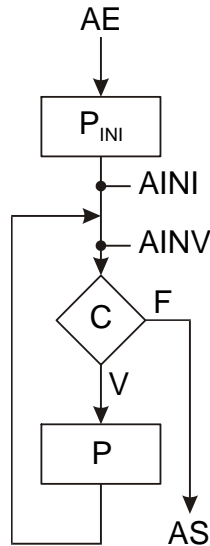
graph TD
    AINI --> AINV
    AINV --> C1{C1}
    C1 -- F --> AS
    C1 -- V --> P1[P1]
    P1 --> C2{C2}
    C2 -- F --> AS
    C2 -- V --> P2[P2]
    P2 --> AINV
    
```

No caso de outras saídas condicionais de uma repetição (**break** ou **return**), devem ser argumentados também:

2.  $AINI \ \&\& \ (C1 == F) \Rightarrow AS$  ,  
 $AINI \ \&\& \ ((C1 == V) \oplus P1) \ \&\& \ (C2 == F) \Rightarrow AS$ 
  - executa corretamente 0 iterações
5.  $AINV \ \&\& \ (C1 == F) \Rightarrow AS$  ,  
 $AINV \ \&\& \ ((C1 == V) \oplus P1) \ \&\& \ (C2 == F) \Rightarrow AS$ 
  - sai correto após  $n > 0$  iterações

Junho 2009
Alessandro Garcia - LES/DI/PUC-Rio
18 / 36

## Proposições para a repetição



- Todas as repetições requerem um **estado inicial**
  - de maneira geral o estabelecimento do estado inicial antecede imediatamente o código da repetição
- Temos então que argumentar mais uma proposição:

$$0. AE \oplus P_{INI} \Rightarrow AINI$$

Junho 2009

Alessandro Garcia - LES/DI/PUC-Rio

19 / 36

## Exemplo: intercalação de arquivos



- Sejam dois arquivos *ArqA* e *ArqB* contendo **registros do mesmo tipo e ordenados crescentemente** segundo o campo **Chave** contido no registro
  - Os arquivos **não contêm Chaves repetidas**, no entanto, podem existir **Chaves iguais em registros dos dois arquivos**
  - **Todas as Chaves são menores do que MAX**
- O que se deseja é um programa que intercale estes dois arquivos, gerando
  - o arquivo *ArqS* contendo **todos os registros** de *ArqA* e *ArqB* **sem Chaves repetidas**, ordenados crescentemente segundo **Chave**
  - o arquivo *ArqE* contendo **todos os pares de registros** de *ArqA* e *ArqB* **com Chaves iguais**, ordenado crescentemente e tendo o registro de *ArqA* precedendo o registro de *ArqB* em cada par de registros com a mesma **Chave**.

Junho 2009

Alessandro Garcia - LES/DI/PUC-Rio

20 / 36

## Intercalação: primeira tentativa



```
AE: ---
LerRegistro de ArqA para BufferA
LerRegistro de ArqB para BufferB
AINI: ---
while ( BufferA.Chave < MAX || BufferB.Chave < MAX )
{
    AINV: ---
    Selecione o registro de menor chave, seja este o
        que está contido no BufferX
    TransferirRegistro de ArqX para ArqS
    if ( BufferA.Chave == BufferB.Chave )
    {
        TransferirRegistro de ArqA para ArqE
        TransferirRegistro de ArqB para ArqE
    } /* if */
} /* while */
AS: ---
```

Junho 2009

Alessandro Garcia - LES/DI/PUC-Rio

21 / 36

## Intercalação: primeira tentativa



Funções de apoio

- **LerRegistro** de ArqX para BufferX

se o arquivo ArqX ainda não terminou  
lê o próximo registro do ArqX para o BufferX  
senão cria um registro vazio em BufferX com  
Chave == MAX

- Esta função simula um  $n+1$  ésimo registro para um arquivo com  $n$  registros

- **TransferirRegistro** de ArqX para ArqY

grava o conteúdo de BufferX em ArqY  
LerRegistro de ArqX para BufferX

Junho 2009

Alessandro Garcia - LES/DI/PUC-Rio

22 / 36

## Intercalação: primeira tentativa



**AE: ??????**

LerRegistro de ArqA para BufferA

LerRegistro de ArqB para BufferB

**AINI: ---**

```
while ( BufferA.Chave < MAX || BufferB.Chave < MAX )
{
    AINV: ---
    Selecione o registro de menor chave, seja este o
        que está contido no BufferX
    TransferirRegistro de ArqX para ArqS
    if ( BufferA.Chave == BufferB.Chave )
    {
        TransferirRegistro de ArqA para ArqE
        TransferirRegistro de ArqB para ArqE
    } /* if */
} /* while */
AS: ---
```

Junho 2009

Alessandro Garcia - LES/DI/PUC-Rio

23 / 36

## Intercalação: primeira tentativa



- **AE:**
  - ArqA, ArqB
    - estão abertos para leitura
    - estão posicionados antes do primeiro registro
    - estão ordenados em ordem crescente segundo *Chave*
    - não possuem *Chave* repetida
    - todas *Chave* < MAX
  - ArqS, ArqE
    - estão abertos para gravação
    - estão vazios
- **Por que BufferA e BufferB não aparecem na AE?**

Junho 2009

Alessandro Garcia - LES/DI/PUC-Rio

24 / 36

## Intercalação: primeira tentativa



- **AE:**
  - *ArqA, ArqB*
    - estão abertos para leitura
    - estão posicionados antes do primeiro registro
    - estão ordenados em ordem crescente segundo *Chave*
    - não possuem *Chave* repetida
    - todas *Chave* < MAX
  - *ArqS, ArqE*
    - estão abertos para gravação
    - estão vazios
- **Por que BufferA e BufferB não aparecem na AE?**
  - por que são criados no corpo da função sendo argumentada

Junho 2009

Alessandro Garcia - LES/DI/PUC-Rio

25 / 36

## Intercalação: primeira tentativa



```
AE: ---
LerRegistro de ArqA para BufferA
LerRegistro de ArqB para BufferB
AINI: ?????
while ( BufferA.Chave < MAX || BufferB.Chave < MAX )
{
    AINV: ---
    Selecione o registro de menor chave, seja este o
        que está contido no BufferX
    TransferirRegistro de ArqX para ArqS
    if ( BufferA.Chave == BufferB.Chave )
    {
        TransferirRegistro de ArqA para ArqE
        TransferirRegistro de ArqB para ArqE
    } /* if */
} /* while */
AS: ---
```

Junho 2009

Alessandro Garcia - LES/DI/PUC-Rio

26 / 36

## Intercalação: primeira tentativa



- **AE:**
  - *ArqA, ArqB*
    - estão abertos para leitura
    - estão posicionados antes do primeiro registro
    - estão ordenados em ordem crescente segundo *Chave*
    - não possuem *Chave* repetida
    - todas *Chave* < MAX
  - *ArqS, ArqE*
    - estão abertos para gravação
    - estão vazios
- **AINI:**
  - *ArqA, ArqB* simulam um registro adicional com *Chave* == MAX caso número de registros == 0
  - *BufferA* e *BufferB* contêm respectivamente os primeiros registros de *ArqA* e *ArqB*
  - *ArqS, ArqE*: estão abertos e vazios

Junho 2009

Alessandro Garcia - LES/DI/PUC-Rio

27 / 36

## Intercalação: primeira tentativa



```
AE: ---
LerRegistro de ArqA para BufferA
LerRegistro de ArqB para BufferB
AINI: ---
while ( BufferA.Chave < MAX || BufferB.Chave < MAX )
{
    AINV: ---
    Selecione o registro de menor chave, seja este o
        que está contido no BufferX
    TransferirRegistro de ArqX para ArqS
    if ( BufferA.Chave == BufferB.Chave )
    {
        TransferirRegistro de ArqA para ArqE
        TransferirRegistro de ArqB para ArqE
    } /* if */
} /* while */

AS: ????
```

Junho 2009

Alessandro Garcia - LES/DI/PUC-Rio

28 / 36

## Intercalação: primeira tentativa



- **AS:**
  - *ArqS* contém todos os registros de *ArqA* e *ArqB*, sem conter *Chaves* repetidas, e com *Chave* < MAX
  - *ArqE* contém todos os pares de registros de *ArqA* e *ArqB*
    - em que os pares têm *Chaves* iguais, e *Chave* < MAX,
    - sendo que em cada par o registro proveniente de *ArqA* antecede o proveniente de *ArqB*

Junho 2009

Alessandro Garcia - LES/DI/PUC-Rio

29 / 36

## Intercalação: primeira tentativa



```
AE: ---
LerRegistro de ArqA para BufferA
LerRegistro de ArqB para BufferB
AINI: ---
while ( BufferA.Chave < MAX || BufferB.Chave < MAX )
{
    AINV: ?????
    Selecione o registro de menor chave, seja este o
        que está contido no BufferX
    TransferirRegistro de ArqX para ArqS
    if ( BufferA.Chave == BufferB.Chave )
    {
        TransferirRegistro de ArqA para ArqE
        TransferirRegistro de ArqB para ArqE
    } /* if */
} /* while */
AS: ---
```

Junho 2009

Alessandro Garcia - LES/DI/PUC-Rio

30 / 36

## Intercalação: primeira tentativa



- **AINV:**

- Os registros correntes estão armazenados em *BufferA* e *BufferB*
  - estes são os de menor valor de *Chave* ainda não gravados em *ArqS* ou *ArqE*
  - Ou um deles contém *Chave == MAX*
- *ArqS* contém todos os registros de *ArqA* e *ArqB*
  - com *Chave* menor do que a menor selecionável
  - sem conter *Chaves* repetidas, e com *Chave < MAX*
- *ArqE* contém todos os pares de registros de *ArqA* e *ArqB*
  - com *Chaves* menores do que a menor selecionável, em que os pares têm *Chaves* iguais,
  - e com *Chave < MAX*
  - sendo que em cada par o registro proveniente de *ArqA* antecede o proveniente de *ArqB*

Junho 2009

Alessandro Garcia - LES/DI/PUC-Rio

31 / 36

## Intercalação: primeira tentativa



0.  $AE \oplus P_{INI} \Rightarrow AINI$     inicialização da função é correta?

- AE:

- *ArqA*, *ArqB*: estão abertos para leitura; estão posicionados antes do primeiro registro; estão ordenados em ordem crescente segundo *Chave*; não possuem *Chave* repetida; todas *Chave < MAX*
- *ArqS*, *ArqE*: estão abertos para gravação; estão vazios

**LerRegistro de ArqA para BufferA**

**LerRegistro de ArqB para BufferB**

- AINI:

- *ArqA*, *ArqB* simulam um registro adicional com *Chave == MAX* (caso entrada:ArqA e entrada:ArqB fossem vazios)
- *BufferA* e *BufferB* contêm respectivamente os primeiros registros de *ArqA* e *ArqB*
- *ArqS*, *ArqE*: estão abertos e vazios

ArqS e ArqE não são modificados pelas funções de leitura

- Correto : Consequência direta da definição de LerRegistro

Junho 2009

Alessandro Garcia - LES/DI/PUC-Rio

32 / 36



## Intercalação: primeira tentativa



### 1. $AINI \Rightarrow AINV$ a invariante vale ao iniciar?

- **AINI:**

- *ArqA*, *ArqB* simulam um registro adicional com *Chave* == MAX após o último registro do arquivo
- *BufferA* e *BufferB* contêm respectivamente os primeiros registros de *ArqA* e *ArqB*
- *ArqS*, *ArqE*: estão abertos e vazios

Verdade pois...

- **AINV:**

- São selecionáveis os registros em *BufferA* e *BufferB* estes são os de menor valor de *Chave* ainda não gravados em *ArqS* ou *ArqE*
- *ArqS* contém todos os registros de *ArqA* e *ArqB* com *Chave* menor do que a menor selecionável, sem conter *Chaves* repetidas, e com *Chave* < MAX
- *ArqE* contém todos os pares de registros de *ArqA* e *ArqB* com *Chaves* menores do que a menor selecionável, em que os pares têm *Chaves* iguais, e com *Chave* < MAX, sendo que em cada par o registro proveniente de *ArqA* antecede o proveniente de *ArqB*

- Correto

Junho 2009

Alessandro Garcia - LES/DI/PUC-Rio

33 / 36

## Intercalação: primeira tentativa



### 2. $AINI \ \&\& \ (C == F) \Rightarrow AS$ executa corretamente 0 iterações?

- **AINI:**

- *ArqA*, *ArqB* simulam um registro adicional com *Chave* == MAX após o último registro do arquivo
- *BufferA* e *BufferB* contêm respectivamente os primeiros registros de *ArqA* e *ArqB*
- *ArqS*, *ArqE*: estão abertos e vazios

**while ( BufferA.Chave < MAX || BufferB.Chave < MAX )**

- **AS:**

- *ArqS* contém todos os registros de *ArqA* e *ArqB*, sem conter *Chaves* repetidas, e com *Chave* < MAX
- *ArqE* contém todos os pares de registros de *ArqA* e *ArqB* em que os pares têm *Chaves* iguais, e *Chave* < MAX, sendo que em cada par o registro proveniente de *ArqA* antecede o proveniente de *ArqB*

- Para ocorrer isso *ArqA* e *ArqB* devem estar vazios. Neste caso correto em consequência direta de LerRegistro

Junho 2009

Alessandro Garcia - LES/DI/PUC-Rio

34 / 36

## Intercalação: primeira tentativa



### 3. $AINI \ \&\& \ (C == V) \oplus P \Rightarrow AINV$

executa corretamente a primeira iteração?

- Condições possíveis
  - $ArqA$  não vazio e  $ArqB$  vazio 👍
  - $ArqA$  vazio e  $ArqB$  não vazio 👍
  - $ArqA$  não vazio e  $ArqB$  não vazio, Chaves diferentes ao entrar
    - Nos três casos  $AINV$  valerá, pois terá sido gravado o primeiro registro de menor valor em  $ArqS$  e um novo registro terá sido lido
  - $ArqA$  não vazio e  $ArqB$  não vazio, Chaves iguais ao entrar
    - E neste caso?

Junho 2009

Alessandro Garcia - LES/DI/PUC-Rio

35 / 36

## Intercalação: primeira tentativa



```
AE: ---
LerRegistro
LerRegistro de ArqB para BufferB
AINI: ---
while ( BufferA.Chave < MAX || BufferB.Chave < MAX )
{
    AINV: ---
    Selecione o registro de menor chave, seja este o
        que está contido no BufferX
    TransferirRegistro de ArqX para ArqS
    if ( BufferA.Chave == BufferB.Chave )
    {
        TransferirRegistro de ArqA para ArqE
        TransferirRegistro de ArqB para ArqE
    } /* if */
} /* while */
AS: ---
```

$ArqA$  não vazio e  $ArqB$  não vazio, Chaves iguais ao entrar

E neste caso? Qual assertiva em  $AINV$  será inválida?

Junho 2009

Alessandro Garcia - LES/DI/PUC-Rio

36 / 36

Laboratório de Engenharia de Software


## Intercalação: primeira tentativa

- **AINV:**

*ArqA* não vazio e *ArqB* não vazio, *Chaves* iguais ao entrar

E neste caso? Qual assertiva será inválida?

- Os registros correntes estão armazenados em *BufferA* e *BufferB*
  - estes são os de menor valor de *Chave* ainda não gravados em *ArqS* ou *ArqE*
  - Ou um deles contém *Chave* == *MAX*
- *ArqS* contém todos os registros de *ArqA* e *ArqB*
  - com *Chave* menor do que a menor selecionável
  - sem conter *Chaves* repetidas, e com *Chave* < *MAX*
- *ArqE* contém todos os pares de registros de *ArqA* e *ArqB*
  - com *Chaves* menores do que a menor selecionável, em que os pares têm *Chaves* iguais,
  - e com *Chave* < *MAX*
  - sendo que em cada par o registro proveniente de *ArqA* antecede o proveniente de *ArqB*



Junho 2009
Alessandro Garcia - LES/DI/PUC-Rio
37 / 36


Laboratório de Engenharia de Software

## Intercalação: primeira tentativa

3.  $AINI \ \&\& \ ( \ C == V ) \oplus P \Rightarrow AINV$

executa corretamente a primeira iteração?

- Condições possíveis
  - *ArqA* não vazio e *ArqB* vazio 👍
  - *ArqA* vazio e *ArqB* não vazio 👍
  - *ArqA* não vazio e *ArqB* não vazio, *Chaves* diferentes ao entrar 👍
    - Nos três casos *AINV* valerá, pois terá sido gravado o primeiro registro de menor valor em *ArqS* e um novo registro terá sido lido
  - *ArqA* não vazio e *ArqB* não vazio, *Chaves* iguais ao entrar
    - Neste caso *AINV* não valerá, pois terá sido gravado um dos registros de *Chave* igual em *ArqS*, e o outro permanecerá para ser selecionado na próxima iteração e que, consequentemente, também será gravado em *ArqS*, contrário à exigência que ambos sejam gravados em *ArqE*



Junho 2009
Alessandro Garcia - LES/DI/PUC-Rio
38 / 36

## Intercalação: primeira tentativa



```

AE: ---
LerRegistro de ArqA para BufferA
LerRegistro de ArqB para BufferB
AINI: ---
while ( BufferA.Chave < MAX || BufferB.Chave < MAX )
{
    AINV: ---
    Selecione o registro de menor chave, seja este o
    que está contido no BufferX
    TransferirRegistro de ArqX para ArqS
    if ( BufferA.Chave == BufferB.Chave )
    {
        TransferirRegistro de ArqA para ArqE
        TransferirRegistro de ArqB para ArqE
    } /* if */
} /* while */
AS: ---

```

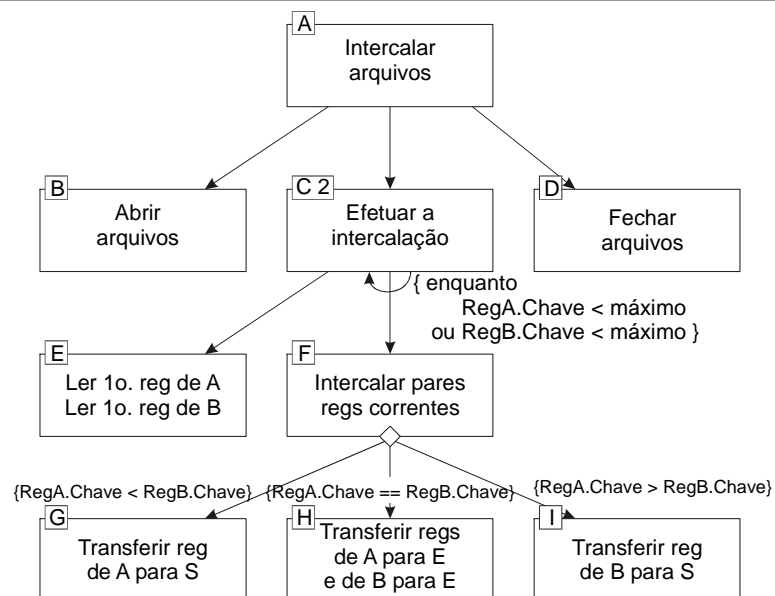
Problema: como Transferir Registro sempre ocorre, já que não está associada com uma condição, BufferX será escrito em ArqS mesmo que chaves sejam iguais

Junho 2009

Alessandro Garcia - LES/DI/PUC-Rio

39 / 36

## Intercalação: segunda tentativa




Junho 2009

Alessandro Garcia - LES/DI/PUC-Rio

40 / 36

Laboratório de Engenharia de Software

## Intercalação: segunda tentativa




0.  $AE \oplus P_{INI} \Rightarrow AINI$
1.  $AINI \Rightarrow AINV$ 
  - a invariante vale ao iniciar
2.  $AINI \ \&\& \ (C == F) \Rightarrow AS$ 
  - executa corretamente 0 iterações
3.  $AINI \ \&\& \ (C == V) \oplus P \Rightarrow AINV$ 
  - executa corretamente a primeira iteração (base da indução)
4.  $AINV \ \&\& \ (C == V) \oplus P \Rightarrow AINV$ 
  - acrescenta corretamente mais uma iteração
5.  $AINV \ \&\& \ (C == F) \Rightarrow AS$ 
  - sai correto após  $n > 0$  iterações
6. Pára após um número finito de iterações

assumindo que estes casos já foram argumentados

Junho 2009
Alessandro Garcia - LES/DI/PUC-Rio
41 / 36

Laboratório de Engenharia de Software

## Intercalação: segunda tentativa



3.  $AINI \ \&\& \ (C == V) \oplus P \Rightarrow AINV$ 

executa corretamente a primeira iteração?

  - Condições possíveis
    - ArqA não vazio e ArqB vazio 👍
    - ArqA vazio e ArqB não vazio 👍
    - ArqA não vazio e ArqB não vazio, Chaves diferentes ao entrar 👍
      - Nos três casos *AINV* valerá, pois terá sido gravado o primeiro registro de menor valor em ArqS e um novo registro terá sido lido
    - ArqA não vazio e ArqB não vazio, Chaves iguais ao entrar 👍
      - Neste caso *AINV* valerá, pois não será gravado nenhum dos registros de Chave igual em ArqS, mas sim em ArqE

```

while ( BufferA.Chave < MAX || BufferB.Chave < MAX )
{
    if (BufferA.Chave < BufferB.Chave){ TransferirRegistro de ArqA para ArqS }
    else if (BufferA.Chave > BufferB.Chave){ TransferirRegistro de ArqB para ArqS}
    else if { // BufferA.Chave == BufferB.Chave
        TransferirRegistro de ArqA para ArqE
        TransferirRegistro de ArqB para ArqE
    } /* if */
} /* while */
    
```

Junho 2009
Alessandro Garcia - LES/DI/PUC-Rio
42 / 36

## Intercalação: segunda tentativa



### 4. $AINV \ \&\& \ (C == V) \oplus P \Rightarrow AINV$ acrescenta a n-ésima iteração?

#### • AINV:

- São selecionáveis os registros em *BufferA* e *BufferB* estes são os de menor valor de *Chave* ainda não gravados em *ArqS* ou *ArqE*
- *ArqS* contém todos os registros de *ArqA* e *ArqB* com *Chave* menor do que a menor selecionável, sem conter *Chaves* repetidas e  $< MAX$
- *ArqE* contém todos os pares de registros de *ArqA* e *ArqB* com *Chaves* menores do que a menor selecionável, em que os pares têm *Chaves* iguais, e com *Chave*  $< MAX$ , sendo que em cada par o registro proveniente de *ArqA* antecede o proveniente de *ArqB*

```
while ( BufferA.Chave < MAX || BufferB.Chave < MAX )
{
    if (BufferA.Chave < BufferB.Chave){ TransferirRegistro de ArqA para ArqS }
    else if (BufferA.Chave > BufferB.Chave){ TransferirRegistro de ArqB para ArqS}
    else if { // BufferA.Chave == BufferB.Chave
        TransferirRegistro de ArqA para ArqE
        TransferirRegistro de ArqB para ArqE
    } /* if */
} /* while */
```

## Intercalação: segunda tentativa



### 5. $AINV \ \&\& \ (C == F) \Rightarrow AS$ sai correto após $n > 0$ iterações?

#### • AINV:

- São selecionáveis os registros em *BufferA* e *BufferB* estes são os de menor valor de *Chave* ainda não gravados em *ArqS* ou *ArqE*
- *ArqS* contém todos os registros de *ArqA* e *ArqB* com *Chave* menor do que a menor selecionável, sem conter *Chaves* repetidas e  $< MAX$
- *ArqE* contém todos os pares de registros de *ArqA* e *ArqB* com *Chaves* menores do que a menor selecionável, em que os pares têm *Chaves* iguais, e com *Chave*  $< MAX$ , sendo que em cada par o registro proveniente de *ArqA* antecede o proveniente de *ArqB*
- $\&\& \text{ BufferA.Chave} == MAX \text{ e } \text{BufferB.Chave} == MAX$

```
while ( BufferA.Chave < MAX || BufferB.Chave < MAX )
{
    if (BufferA.Chave < BufferB.Chave){ TransferirRegistro de ArqA para ArqS }
    else if (BufferA.Chave > BufferB.Chave){ TransferirRegistro de ArqB para ArqS}
    else if { // BufferA.Chave == BufferB.Chave
        TransferirRegistro de ArqA para ArqE
        TransferirRegistro de ArqB para ArqE
    } /* if */
} /* while */
```

implicação direta e...

bloco while não será executado e, portanto, regs. em *ArqS* não receberão os buffers

#### • AS:

- *ArqS* contém todos registros de *ArqA* e *ArqB*, sem *Chaves* repetidas e  $< MAX$
- *ArqE* contém todos os pares de registros de *ArqA* e *ArqB* em que os pares têm *Chaves* iguais, e *Chave*  $< MAX$ , sendo que em cada par o registro proveniente de *ArqA* antecede o proveniente de *ArqB*

## Intercalação: segunda tentativa



### 5. $AINV \ \&\& \ (C == F) \Rightarrow AS$ sai correto após $n > 0$ iterações?

#### • AINV:

- São selecionáveis os registros em *BufferA* e *BufferB* estes são os de menor valor de *Chave* ainda não gravados em *ArqS* ou *ArqE*
- *ArqS* contém todos os registros de *ArqA* e *ArqB* com *Chave* menor do que a menor selecionável, sem conter *Chaves* repetidas e  $< MAX$
- *ArqE* contém todos os pares de registros de *ArqA* e *ArqB* com *Chaves* menores do que a menor selecionável, em que os pares têm *Chaves* iguais, e com *Chave*  $< MAX$ , sendo que em cada par o registro proveniente de *ArqA* antecede o proveniente de *ArqB*
- $\&\& BufferA.Chave == MAX$  e  $BufferB.Chave == MAX$

```
while ( BufferA.Chave < MAX || BufferB.Chave < MAX )
{ if (BufferA.Chave < BufferB.Chave){ TransferirRegistro de ArqA para ArqS }
  else if (BufferA.Chave > BufferB.Chave){ TransferirRegistro de ArqB para ArqS }
  else if { // BufferA.Chave == BufferB.Chave
    TransferirRegistro de ArqA para ArqE
    TransferirRegistro de ArqB para ArqE
  } /* if */
} /* while */
```

implicação  
direta e...

bloco while não será  
executado e, portanto,  
regs. em *ArqE* não receberão  
os buffers

#### • AS:

- *ArqS* contém todos registros de *ArqA* e *ArqB*, sem *Chaves* repetidas e  $< MAX$
- *ArqE* contém todos os pares de registros de *ArqA* e *ArqB* em que os pares têm *Chaves* iguais, e *Chave*  $< MAX$ , sendo que em cada par o registro proveniente de *ArqA* antecede o proveniente de *ArqB*



## Intercalação: segunda tentativa



### 6. Para após um número finito de iterações?

```
while ( BufferA.Chave < MAX || BufferB.Chave < MAX )
{ if (BufferA.Chave < BufferB.Chave){ TransferirRegistro de ArqA para ArqS }
  else if (BufferA.Chave > BufferB.Chave){ TransferirRegistro de ArqB para ArqS }
  else if { // BufferA.Chave == BufferB.Chave
    TransferirRegistro de ArqA para ArqE
    TransferirRegistro de ArqB para ArqE
  } /* if */
}
```

#### • Dado que...

- em cada interação, a leitura sempre implica o avanço de pelo menos um registro em um dos arquivos (o que contém o registro com chave menor)
  - ou avança em ambos arquivos (caso chaves são iguais)
- caso processamento de um dos arquivos seja terminado antes:
  - laço continua até que todos os registros do outro arquivo sejam lidos, já que o outro buffer terá  $MAX$  (certamente maior que a maior chave não processada) como chave
  - e sempre um dos 2 primeiros if's será executado, garantindo avanço do arquivo até a condição da parada ser satisfeita:  $BufferA.Chave == MAX \ \&\& \ BufferB.Chave == MAX$
- se processamentos dos arquivos terminarem juntos – isto é, últimas chaves são iguais, a condição da parada valerá:  $BufferA.Chave == MAX \ \&\& \ BufferB.Chave == MAX$

## Exemplo: Função Index



```
int Index( char * sBase , char * sProc ,  
          int inxOrg , int inxFim )
```

- **sBase** é o string dentro do qual será procurado o string **sProc**
- **sProc** é o string a ser encontrado
- **inxOrg** é o índice do primeiro caractere inclusive de **sBase** a partir do qual o string **sProc** será procurado
- **inxFim** é o índice do último caractere inclusive de **sBase** a ser considerado ao procurar **sProc**
- Retorna o índice inicial da primeira ocorrência de **sProc** dentro de **sBase**, ou -1 caso não encontre **sProc**
- Obs. o algoritmo usado é "ingênuo". Existem outros algoritmos muito melhores.

## Função Index: Assertiva entrada



AE:

- **sBase** != NULL
- $0 \leq \text{strlen}(sBase) \leq 32767$ 
  - OBS 1: **sBase** pode ser o *string* nulo ( $\text{strlen} == 0$ ). Note que *string* nulo é um *string* legal com **sBase** != NULL
- **sProc** != NULL,
- $0 \leq \text{strlen}(sProc) \leq 32767$
- **inxOrg** <= **inxFim**



## Função Index: Assertiva saída



AS:

- Index  $\geq 0 \Rightarrow$ 
  - `strlen( sBase )  $\geq$  strlen( sProc )`
  - Index  $\geq$  `inxOrg`  $\geq 0$
  - Index  $\leq$  `inxFim`  $\leq$  `strlen( sBase ) - strlen( sProc )`
  - `sBase[ Index .. strlen( sBase ) - strlen( sProc ) ] == sProc`
- Index == -1  $\Rightarrow$ 
  - `sProc` não existe em `sBase`
- Index == -2  $\Rightarrow$ 
  - dados de entrada ilegais
- Index < -2  $\Rightarrow$ 
  - ilegal

É possível encontrar um string em `sBase` se `strlen( sBase ) == 0` ?

Junho 2009

Alessandro Garcia - LES/DI/PUC-Rio

49 / 36

## Função Index: Código parcial



```
int Index( char * sBase, char * sProc, int inxOrg, int inxFim )
{
    if ( dados de entrada ilegais ) return -2 ;
    Acertar limites
    cInic = sProc[ 0 ] ;
    tamProc = strlen( sProc ) ;
    AINI: 0 <= inxOrg <= inxFim <= strlen(sBase) - strlen(sProc)
    for ( i = inxOrg ; i <= inxFim ; i ++ )
    { AINV: sBase[ inxOrg .. i - 1 ] não contém sProc
        if ( sBase[ i ] == cInic )
        {
            if ( memcmp( sBase + i, sProc, tamProc ) == 0 )
            {
                return i ;
            } /* if */
        } /* if */
    } /* for */
    return -1 ;
} /* function Index */
```

Otimiza um pouco, pois não precisa comparar todo o string quando o primeiro caractere não vale

Junho 2009

Alessandro Garcia - LES/DI/PUC-Rio

50 / 36

## Função Index: Proposições a provar



0.  $AE \oplus P_{INI} \Rightarrow AINI$
1.  $AINI \Rightarrow AINV$ 
  - a invariante vale ao iniciar
2.  $AINI \ \&\& \ (C == F) \Rightarrow AS$ 
  - executa corretamente 0 iterações
3.  $AINI \ \&\& \ (C == V) \oplus P \Rightarrow AINV$ 
  - executa corretamente a primeira iteração (base da indução)
4.  $AINV \ \&\& \ (C == V) \oplus P \Rightarrow AINV$ 
  - acrescenta corretamente mais uma iteração
5.  $AINV \ \&\& \ (C == F) \Rightarrow AS$ 
  - sai correto após  $n > 0$  iterações
6. Pára após um número finito de iterações



FIM