Processamento Cossequencial e Ordenação de Arquivos Grandes

Processamento Cossequencial

6897/9895 – Organização e Recuperação de Dados

Profa. Valéria D. Feltrim

UEM – CTC – DIN

Slides preparados com base no Cap. 7 do livro FOLK, M.J. & ZOELLICK, B. *File Structures*. 2nd Edition, Addison-Wesley Publishing Company, 1992, e nos slides disponibilizados pelo Prof. Pedro de Azevedo Berger (DCC/UnB)

Processamento cossequencial

- Objetivo do processamento cossequencial
  - Processar de forma coordenada duas ou mais listas sequenciais para produzir uma única lista como saída
    - Por ex., duas ou mais listas de chaves de primárias
- As listas de entrada:
  - Devem estar ordenadas por chave
  - Não devem possuir chaves duplicadas
- A lista de saída:
  - Será ordenada por chave
  - Não possuirá chaves duplicadas

Processamento cossequencial

- Tipos de listas resultantes:
  - Interseção (*matching*)
    - A lista de saída é formada por chaves que ocorrem em todas as listas de entrada
    - A lista de saída é ordenada
  - União/Intercalação (*merging*)
    - A lista de saída é formada por todas as chaves das listas de entrada, sem duplicação
    - Os itens são intercalados pela ordem de chave
    - A lista de saída é ordenada

Processamento cossequencial

- Os algoritmos devem contemplar:
  - Inicialização
    - Abertura dos arquivos de entrada e saída
    - Inicialização da variável de controle do laço
    - Inicialização de variáveis usadas na checagem de sequência
  - Leitura dos arquivos
    - Teste da condição de parada
    - Reconhecimento de erro para os arquivos de entrada
      - Verificar se existem chaves duplicadas
      - Verificar se a sequência de chaves está desordenada
  - Sincronização das leituras e geração da saída

Processamento cossequencial

- Interseção (*Matching*)

Lista1	*Lista2*	*Interseção*
Adams	Adams	
Carter	Andrews	
Chin	Bech	
Davis	Burns	
Foster	Carter	
Garwich	Davis	
Johnson	Dempsey	
Rosewald	Rosewald	
Turner	Schmidt	
	Thayer	
	Walsh	
	Willis	

Processamento cossequencial

- Interseção (*Matching*)

Lista1		*Lista2*		*Interseção*
Adams	→	**Adams**	→	**Adams**
Carter		Andrews		
Chin		Bech		
Davis		Burns		
Foster		Carter		
Garwich		Davis		
Johnson		Dempsey		
Rosewald		Rosewald		
Turner		Schmidt		
		Thayer		
		Walsh		
		Willis		

Processamento cossequencial

- Interseção (*Matching*)

Lista1		*Lista2*	*Interseção*
Adams		Adams	**Adams**
Carter	→	Andrews	
Chin		Bech	
Davis		Burns	
Foster		Carter	
Garwich		Davis	
Johnson		Dempsey	
Rosewald		Rosewald	
Turner		Schmidt	
		Thayer	
		Walsh	
		Willis	

Processamento cossequencial

- Interseção (*Matching*)

Lista1		*Lista2*	*Interseção*
Adams		Adams	**Adams**
Carter	|	Andrews	
Chin	→	Bech	
Davis		Burns	
Foster		Carter	
Garwich		Davis	
Johnson		Dempsey	
Rosewald		Rosewald	
Turner		Schmidt	
		Thayer	
		Walsh	
		Willis	

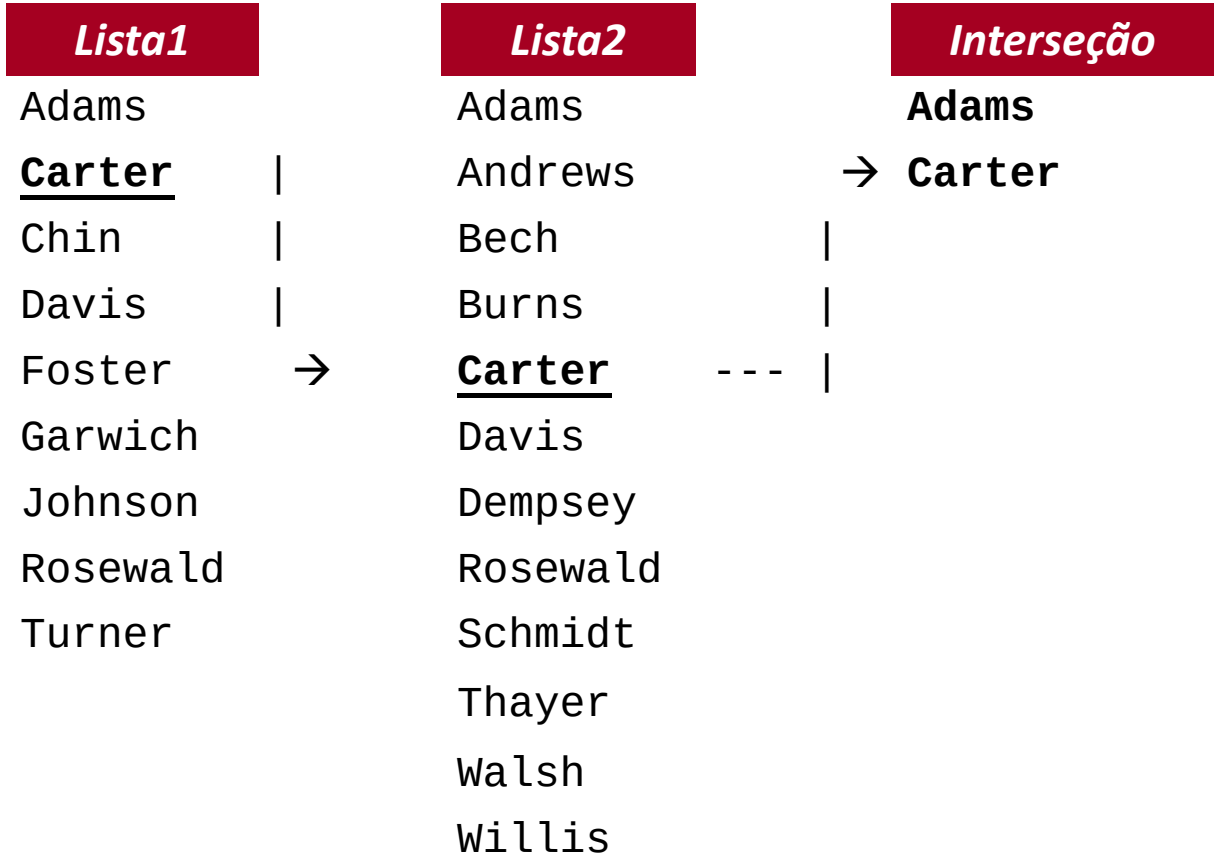
Processamento cossequencial

- Interseção (*Matching*)

Lista1		*Lista2*	*Interseção*
Adams		Adams	**Adams**
Carter	|	Andrews	
Chin	|	Bech	
Davis	→	Burns	
Foster		Carter	
Garwich		Davis	
Johnson		Dempsey	
Rosewald		Rosewald	
Turner		Schmidt	
		Thayer	
		Walsh	
		Willis	

Processamento cossequencial

- Interseção (*Matching*)



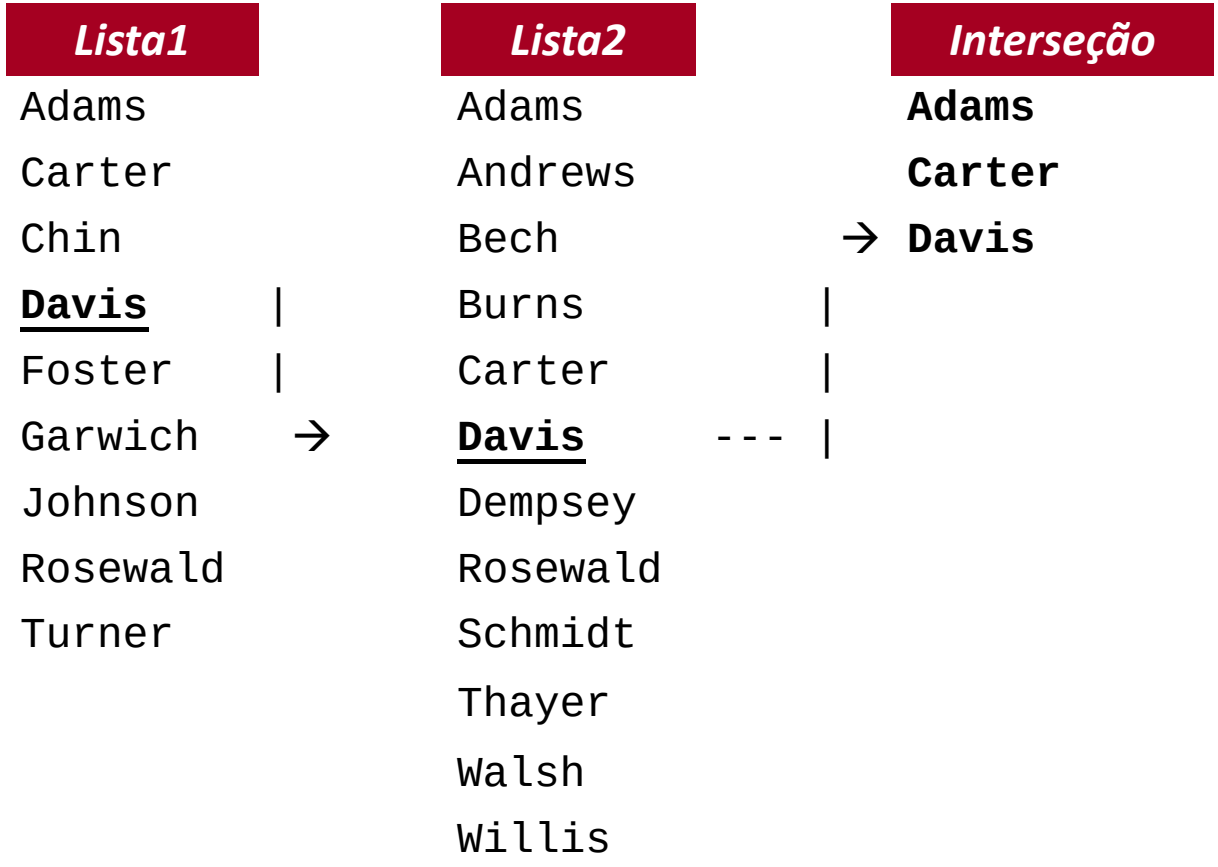
Processamento cossequencial

- Interseção (*Matching*)

Lista1		*Lista2*	*Interseção*
Adams		Adams	**Adams**
Carter		Andrews	**Carter**
Chin	|	Bech	
Davis	|	Burns	
Foster	|	Carter	
Garwich	→	Davis	
Johnson		Dempsey	
Rosewald		Rosewald	
Turner		Schmidt	
		Thayer	
		Walsh	
		Willis	

Processamento cossequencial

- Interseção (*Matching*)



Processamento cossequencial

- Interseção (*Matching*)

Lista1		*Lista2*	*Interseção*
Adams		Adams	**Adams**
Carter		Andrews	**Carter**
Chin		Bech	**Davis**
Davis		Burns	
Foster	|	Carter	
Garwich	|	Davis	
Johnson	→	Dempsey	
Rosewald		Rosewald	
Turner		Schmidt	
		Thayer	
		Walsh	
		Willis	

Processamento cossequencial

- Interseção (*Matching*)

Lista1		*Lista2*	*Interseção*
Adams		Adams	**Adams**
Carter		Andrews	**Carter**
Chin		Bech	**Davis**
Davis		Burns	
Foster	|	Carter	
Garwich	|	Davis	
Johnson	|	Dempsey	
Rosewald	→	Rosewald	
Turner		Schmidt	
		Thayer	
		Walsh	
		Willis	

Processamento cossequencial

- Interseção (*Matching*)

Lista1		*Lista2*	*Interseção*
Adams		Adams	**Adams**
Carter		Andrews	**Carter**
Chin		Bech	**Davis**
Davis		Burns	
Foster		Carter	
Garwich	|	Davis	
Johnson	|	Dempsey	
Rosewald	→	Rosewald	
Turner		Schmidt	
		Thayer	
		Walsh	
		Willis	

Processamento cossequencial

- Interseção (*Matching*)

Lista1	*Lista2*	*Interseção*
Adams	Adams	**Adams**
Carter	Andrews	**Carter**
Chin	Bech	**Davis**
Davis	Burns	
Foster	Carter	
Garwich	Davis	
Johnson	Dempsey	
Rosewald →	Rosewald	
Turner	Schmidt	
	Thayer	
	Walsh	
	Willis	

Processamento cossequencial

- Interseção (*Matching*)

Lista1		*Lista2*		*Interseção*
Adams		Adams		**Adams**
Carter		Andrews		**Carter**
Chin		Bech		**Davis**
Davis		Burns	→	**Rosewald**
Foster		Carter	|	
Garwich		Davis	|	
Johnson		Dempsey	|	
Rosewald	→	**Rosewald**	--- |	
Turner		Schmidt		
		Thayer		
		Walsh		
		Willis		

Processamento cossequencial

- Interseção (*Matching*)

Lista1	*Lista2*	*Interseção*
Adams	Adams	**Adams**
Carter	Andrews	**Carter**
Chin	Bech	**Davis**
Davis	Burns	**Rosewald**
Foster	Carter	
Garwich	Davis	
Johnson	Dempsey	
Rosewald	Rosewald	
Turner →	Schmidt	
	Thayer	
	Walsh	
	Willis	

Processamento cossequencial

- Interseção (*Matching*)

Lista1	*Lista2*	*Interseção*
Adams	Adams	**Adams**
Carter	Andrews	**Carter**
Chin	Bech	**Davis**
Davis	Burns	**Rosewald**
Foster	Carter	
Garwich	Davis	
Johnson	Dempsey	
Rosewald	Rosewald	
Turner	Schmidt	
→	Thayer	
	Walsh	
	Willis	

Processamento cossequencial

- Interseção (*Matching*)

Lista1	*Lista2*	*Interseção*
Adams	Adams	**Adams**
Carter	Andrews	**Carter**
Chin	Bech	**Davis**
Davis	Burns	**Rosewald**
Foster	Carter	
Garwich	Davis	
Johnson	Dempsey	
Rosewald	Rosewald	
Turner	Schmidt	
	Thayer	
	Walsh	
	Willis	

Processamento cossequencial

- Interseção (*Matching*)

Lista1		*Lista2*	*Interseção*
Adams		Adams	**Adams**
Carter		Andrews	**Carter**
Chin		Bech	**Davis**
Davis		Burns	**Rosewald**
Foster		Carter	
Garwich		Davis	
Johnson		Dempsey	
Rosewald		Rosewald	
Turner		Schmidt	
EOF	|	Thayer	
	→	Walsh	
		Willis	

Interseção (*Matching*)

- Sincronização dos arquivos
  - Se *Nome1* < *Nome2*, lemos o próximo da Lista1
  - Se *Nome1* > *Nome2,* lemos o próximo da Lista2
  - Se *Nome1* = *Nome2*, escrevemos o nome na saída e lemos os próximos nomes das listas 1 e 2

Interseção (*Matching*)

PROCEDIMENTO **inicialização**() */* para o programa match*/*

- argumentos passados por referência:

1. PREV1, PREV2: armazenam os últimos nomes lidos das listas 1 e 2
2. LISTA1, LISTA2: descritores dos arquivos de entrada
3. EXISTEM_MAIS_NOMES: *flag* usada para controlar o laço principal
4. SAIDA: descritor do arquivo de saída

/ LOW_VALUE é um valor constante que sempre será*
*menor que qualquer valor da lista de entrada */*

PREV1 := *LOW_VALUE*

PREV2 := *LOW_VALUE*

abra o arquivo com a lista 1 como LISTA1

abra o arquivo com a lista 2 como LISTA2

se (os dois arquivos foram abertos com sucesso) então

EXISTEM_MAIS_NOMES := *TRUE*

crie o arquivo de saída como SAIDA

fim PROCEDIMENTO

Interseção (*Matching*)

PROCEDIMENTO **input**() */* para o programa match*/*

- argumento passado por valor:

1. LISTA: descritor do arquivo a ser lido (pode ser LISTA1 ou LISTA2)

- argumentos passados por referência:

1. NOME_ANT: armazena o último nome lido do arquivo ENTRADA
2. NOME: armazena o nome que será retornado pelo procedimento
3. EXISTEM_MAIS_NOMES: *flag* usada para controlar o laço principal

```
    leia NOME do arquivo LISTA
    se (EOF) então
            EXISTEM_MAIS_NOMES := FALSE
    senão se (NOME <= NOME_ANT) então
            escreva ("Erro na checagem de sequência")
            aborte o processamento
    fim se
    NOME_ANT := NOME
fim PROCEDIMENTO
```

Interseção (*Matching*)

```
PROGRAMA: match
    chame o procedimento inicialização()
    chame o procedimento input() para ler NOME1 da LISTA1
    chame o procedimento input() para ler NOME2 da LISTA2
    enquanto (EXISTEM_MAIS_NOMES) faça
        se (NOME1 < NOME2) então
                chame o procedimento input() para ler NOME1 da LISTA1
        senão se (NOME1 > NOME2) então
                chame o procedimento input() para ler NOME2 da LISTA2
        senão   /* chaves iguais */
                escreva NOME1 em SAIDA
                chame o procedimento input() para ler NOME1 da LISTA1
                chame o procedimento input() para ler NOME2 da LISTA2
        fim se
    fim enquanto
fim PROGRAMA
```

Processamento cossequencial

- União (*Merging*)

Lista1		*Lista2*		*União*
Adams	→	**Adams**	→	**Adams**
Carter		Andrews		
Chin		Bech		
Davis		Burns		
Foster		Carter		
Garwich		Davis		
Johnson		Dempsey		
Rosewald		Rosewald		
Turner		Schmidt		
		Thayer		
		Walsh		
		Willis		

Processamento cossequencial

- União (*Merging*)

Lista1		*Lista2*		*União*
Adams		Adams		**Adams**
Carter	→	**Andrews**	→	**Andrews**
Chin		Bech		
Davis		Burns		
Foster		Carter		
Garwich		Davis		
Johnson		Dempsey		
Rosewald		Rosewald		
Turner		Schmidt		
		Thayer		
		Walsh		
		Willis		

Processamento cossequencial

- União (*Merging*)

Lista1		*Lista2*		*União*
Adams		Adams		**Adams**
Carter	|	Andrews		**Andrews**
Chin	→	**Bech**	→	**Bech**
Davis		Burns		
Foster		Carter		
Garwich		Davis		
Johnson		Dempsey		
Rosewald		Rosewald		
Turner		Schmidt		
		Thayer		
		Walsh		
		Willis		

Processamento cossequencial

- União (*Merging*)

Lista1		*Lista2*		*União*
Adams		Adams		**Adams**
Carter	|	Andrews		**Andrews**
Chin	|	Bech		**Bech**
Davis	→	**Burns**	→	**Burns**
Foster		Carter		
Garwich		Davis		
Johnson		Dempsey		
Rosewald		Rosewald		
Turner		Schmidt		
		Thayer		
		Walsh		
		Willis		

Processamento cossequencial

- União (*Merging*)

Lista1		*Lista2*		*União*
Adams		Adams		**Adams**
Carter	|	Andrews		**Andrews**
Chin	|	Bech		**Bech**
Davis	|	Burns		**Burns**
Foster	→	**Carter**	→	**Carter**
Garwich		Davis		
Johnson		Dempsey		
Rosewald		Rosewald		
Turner		Schmidt		
		Thayer		
		Walsh		
		Willis		

Processamento cossequencial

- União (*Merging*)

Lista1		*Lista2*		*União*
Adams		Adams		**Adams**
Carter		Andrews		**Andrews**
Chin	|	Bech		**Bech**
Davis	|	Burns		**Burns**
Foster	→	Carter		**Carter**
Garwich		Davis	→	**Chin**
Johnson		Dempsey		
Rosewald		Rosewald		
Turner		Schmidt		
		Thayer		
		Walsh		
		Willis		

Processamento cossequencial

- União (*Merging*)

Lista1		*Lista2*		*União*
Adams		Adams		**Adams**
Carter		Andrews		**Andrews**
Chin		Bech		**Bech**
Davis	|	Burns		**Burns**
Foster	|	Carter		**Carter**
Garwich	→	**Davis**	|	**Chin**
Johnson		Dempsey	→	**Davis**
Rosewald		Rosewald		
Turner		Schmidt		
		Thayer		
		Walsh		
		Willis		

Processamento cossequencial

- União (*Merging*)

Lista1		*Lista2*		*União*
Adams		Adams		**Adams**
Carter		Andrews		**Andrews**
Chin		Bech		**Bech**
Davis		Burns		**Burns**
Foster	|	Carter		**Carter**
Garwich	|	Davis		**Chin**
Johnson	→	**Dempsey**	|	**Davis**
Rosewald		Rosewald	→	**Dempsey**
Turner		Schmidt		
		Thayer		
		Walsh		
		Willis		

Processamento cossequencial

- União (*Merging*)

Lista1		*Lista2*		*União*
Adams		Adams		**Adams**
Carter		Andrews		**Andrews**
Chin		Bech		**Bech**
Davis		Burns		**Burns**
Foster	|	Carter		**Carter**
Garwich	|	Davis		**Chin**
Johnson	|	Dempsey		**Davis**
Rosewald	→	Rosewald	|	**Dempsey**
Turner		Schmidt	→	**Foster**
		Thayer		
		Walsh		
		Willis		

Processamento cossequencial

- União (*Merging*)

Lista1		*Lista2*		*União*
Adams		Adams		**Adams**
Carter		Andrews		**Andrews**
Chin		Bech		**Bech**
Davis		Burns		**Burns**
Foster		Carter		**Carter**
Garwich	|	Davis		**Chin**
Johnson	|	Dempsey		**Davis**
Rosewald	→	Rosewald	|	**Dempsey**
Turner		Schmidt	|	**Foster**
		Thayer	→	**Garwich**
		Walsh		
		Willis		

Processamento cossequencial

- União (*Merging*)

Lista1		*Lista2*		*União*
Adams		Adams		**Adams**
Carter		Andrews		**Andrews**
Chin		Bech		**Bech**
Davis		Burns		**Burns**
Foster		Carter		**Carter**
Garwich		Davis		**Chin**
Johnson	|	Dempsey		**Davis**
Rosewald	→	Rosewald	|	**Dempsey**
Turner		Schmidt	|	**Foster**
		Thayer	|	**Garwich**
		Walsh	→	**Johnson**
		Willis		

Processamento cossequencial

- União (*Merging*)

Lista1		*Lista2*		*União*
Adams		Adams		**Adams**
Carter		Andrews		**Andrews**
Chin		Bech		**Bech**
Davis		Burns		**Burns**
Foster		Carter		**Carter**
Garwich		Davis		**Chin**
Johnson		Dempsey		**Davis**
Rosewald	→	**Rosewald**	|	**Dempsey**
Turner		Schmidt	|	**Foster**
		Thayer	|	**Garwich**
		Walsh	|	**Johnson**
		Willis	→	...

Processamento cossequencial

- União (*Merging*)

Lista1		*Lista2*		*União*
Adams		Adams		**Adams**
Carter		Andrews		**Andrews**
Chin		Bech		**Bech**
Davis		Burns		**Burns**
Foster		Carter		**Carter**
Garwich		Davis		**Chin**
Johnson		Dempsey		**Davis**
Rosewald		Rosewald		**Dempsey**
Turner	→	**Schmidt**	|	**Foster**
		Thayer	|	**Garwich**
		Walsh	|	**Johnson**
		Willis	→	...

Processamento cossequencial

- União (*Merging*)

Lista1	*Lista2*	*União*
Adams	Adams	**Adams**
Carter	Andrews	**Andrews**
Chin	Bech	**Bech**
Davis	Burns	**Burns**
Foster	Carter	**Carter**
Garwich	Davis	**Chin**
Johnson	Dempsey	**Davis**
Rosewald	Rosewald	**Dempsey**
Turner |	Schmidt	**Foster**
→	**Thayer** |	**Garwich**
	Walsh |	**Johnson**
	Willis →	...

Processamento cossequencial

- União (*Merging*)

Lista1	*Lista2*	*União*
Adams	Adams	**Adams**
Carter	Andrews	**Andrews**
Chin	Bech	**Bech**
Davis	Burns	**Burns**
Foster	Carter	**Carter**
Garwich	Davis	**Chin**
Johnson	Dempsey	**Davis**
Rosewald	Rosewald	**Dempsey**
Turner |	Schmidt	**Foster**
|	Thayer	**Garwich**
→	Walsh |	**Johnson**
	Willis →	...

Processamento cossequencial

- União (*Merging*)

Lista1	*Lista2*		*União*
Adams	Adams		**Adams**
Carter	Andrews		**Andrews**
Chin	Bech		**Bech**
Davis	Burns		**Burns**
Foster	Carter		**Carter**
Garwich	Davis		**Chin**
Johnson	Dempsey		**Davis**
Rosewald	Rosewald		**Dempsey**
Turner	Schmidt		**Foster**
EOF	Thayer		**Garwich**
	Walsh	|	**Johnson**
	Willis	→	...

Processamento cossequencial

- União (*Merging*)

Lista1	*Lista2*		*União*
Adams	Adams		**Adams**
Carter	Andrews		**Andrews**
Chin	Bech		**Bech**
Davis	Burns		**Burns**
Foster	Carter		**Carter**
Garwich	Davis		**Chin**
Johnson	Dempsey		**Davis**
Rosewald	Rosewald		**Dempsey**
Turner	Schmidt		**Foster**
EOF	Thayer		**Garwich**
	Walsh		**Johnson**
	Willis	|	**...**
		→	**Willis**

Processamento cossequencial

- União (*Merging*)

Lista1
Adams
Carter
Chin
Davis
Foster
Garwich
Johnson
Rosewald
Turner
EOF

Lista2
Adams
Andrews
Bech
Burns
Carter
Davis
Dempsey
Rosewald
Schmidt
Thayer
Walsh
Willis
EOF

União
Adams
Andrews
Bech
Burns
Carter
Chin
Davis
Dempsey
Foster
Garwich
Johnson
...
Willis

União (*Merging*)

- O procedimento de inicialização é igual ao do *match*
- Manipulação dos arquivos de entrada
  - No *merge*, o processamento deve continuar enquanto houver nomes em qualquer uma das listas, i.e., ambos os arquivos de entrada devem ser lidos até o fim
  - Para tal, é preciso fazer algumas mudanças no procedimento input():
    - Manter a *flag* EXISTEM_MAIS_NOMES como TRUE enquanto existirem entradas em qualquer um dos arquivos, mas...
    - Não ficar tentando ler um arquivo que já terminou
      - Solução: fazer com que NOME (1 ou 2, dependendo do arquivo que terminou antes) receba um valor que seja maior que qualquer entrada válida (definido pela constante *HIGH_VALUE*)
    - Usar um parâmetro NOME_OUTRA_LISTA para o procedimento saber se a outra lista já terminou

União (*Merging*)

PROCEDIMENTO **input**() /* para o programa merge*/

- argumentos passados por valor:

1. LISTA: descritor do arquivo de entrada a ser usado (pode ser LISTA1 ou LISTA2)
2. NOME_OUTRA_LISTA: armazena o último nome lido da outra lista (pode ser LISTA1 ou LISTA2)

- argumentos passados por referência:

1. NOME_ANT: armazena o último nome lido do arquivo de entrada
2. NOME: armazena o nome que será retornado pelo procedimento
3. EXISTEM_MAIS_NOMES: *flag* usada para controlar o laço principal

```
leia NOME do arquivo LISTA
se (EOF) e (NOME_OUTRA_LISTA = HIGH_VALUE) então
    EXISTEM_MAIS_NOMES := FALSE          /*fim das duas listas*/
senão se (EOF) então
    NOME := HIGH_VALUE                   /*fim desta lista de entrada*/
senão se (NOME <= NOME_ANT) então
    escreva ("Erro na checagem de sequência")
    aborte o processamento
fim se
NOME_ANT := NOME
fim PROCEDIMENTO
```

União (*Merging*)

```
PROGRAMA: merge
        chame o procedimento inicialização()
        chame o procedimento input() para ler NOME1 da LISTA1
        chame o procedimento input() para ler NOME2 da LISTA2
        enquanto (EXISTEM_MAIS_NOMES) faça
            se (NOME1 < NOME2) então
               escreva NOME1 em SAIDA
               chame o procedimento input() para ler NOME1 da LISTA1
            senão se (NOME1 > NOME2) então
               escreva NOME2 em SAIDA
               chame o procedimento input() para ler NOME2 da LISTA2
            senão /* chaves iguais */
               escreva NOME1 em SAIDA
               chame o procedimento input() para ler NOME1 da LISTA1
               chame o procedimento input() para ler NOME2 da LISTA2
            fim se
        fim enquanto
fim PROGRAMA
```

Processamento cossequencial múltiplo

- E quando a entrada é composta por mais de dois arquivos?
  - Interseção (*matching*)
    - Faça o processamento utilizando somente dois arquivos de entrada, gerando um de saída
    - Repita o processo com os arquivos de saída até gerar um único arquivo
  - União/Intercalação (*merging*)
    - Para fazer a intercalação de *n* arquivos, modifique o algoritmo de *merging* inserindo uma função que dados *n* elementos de entrada, retorna o menor valor dentre os *n* elementos

Intercalação múltipla (*Multiway Merging*)

- **Algoritmo *K-way Merge***: intercala *K* arquivos de entrada ordenados, gerando um único arquivo ordenado de saída
  - O valor *K* é a ordem do merge
  - A parte mais cara do processo são os testes para verificar em quais arquivos a menor chave ocorreu, para se saber quais arquivos devem ser lidos a seguir

```
enquanto (EXISTEM_MAIS_NOMES) faça
  NOME_SAIDA := min(NOME1, NOME2, NOME3, ..., NOMEK)
  escreva NOME_SAIDA em SAIDA
  se (NOME1 = NOME_SAIDA) então
    chame o procedimento Input() para ler NOME1 da LISTA1
  se (NOME2 = NOME_SAIDA) então
    chame o procedimento Input() para ler NOME2 da LISTA2
  ...
  se (NOMEK = NOME_SAIDA) então
    chame o procedimento Input() para ler NOMEK da LISTAK
fim enquanto
```

K-Way Merge

- Condição de parada
  - Pode-se usar uma variável contadora
    - Inicia com 0 e é incrementada cada vez que um dos arquivos chega ao fim
  - A condição de parada se torna verdadeira se ocorrer EOF e o contador atingir o número de arquivos de entrada

```
PROCEDIMENTO input()
    ...
    se (EOF) então
      NOME :=  HIGH_VALUE
      contFimArq := contFimArq  + 1
      se contFimArq = numArq então EXISTEM_MAIS_NOMES := FALSE
    fim se
    ...
fim PROCEDIMENTO
```

K-Way Merge

- Se for possível garantir que uma chave ocorre somente em uma lista/arquivo, o procedimento ficaria mais simples e eficiente
- Suponha o uso dos vetores LISTA e NOME
  - LISTA[i] contém os descritores de cada arquivo de entrada
  - NOME[i] contém o nome (chave) corrente associado a cada arquivo de entrada

K-Way Merge

- Algoritmo modificado assumindo que não existem chaves duplicadas

```
/*dê início ao processo lendo um nome de cada lista*/
para i := 1 até K faça
  chame o procedimento Input() para ler NOME[i] da LISTA[i]
fim para
/*inicie a união em K-vias*/
enquanto (EXISTEM_MAIS_NOMES) faça
  /*encontre o nome de menor valor entre os nomes das K listas*/
  MENOR := 1
  para i := 2 até K faça
    se (NOME[i] < NOME[MENOR]) então
      MENOR := i
  fim para
  escreva NOME[MENOR] em SAIDA
  /*substitua o nome (leia o próximo) que foi para a saída*/
  chame o procedimento Input() para ler NOME[M
fim enquanto
```

Quanto maior for o valor de K, mais caro será encontrar o menor