UNISINOS - **UNIVERSIDADE DO VALE DO RIO DOS SINOS**
CIÊNCIAS EXATAS E TECNOLÓGICAS – Curso: Informática / Ciência da Computação

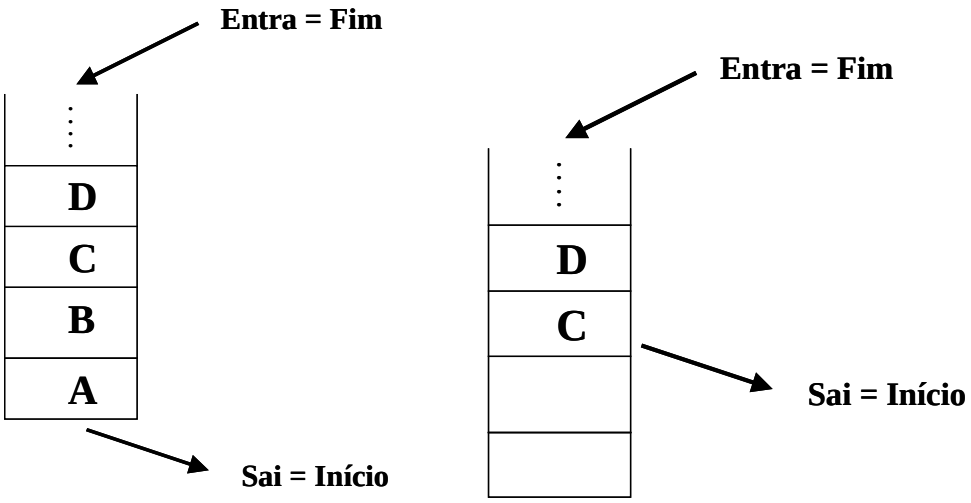
Programação II

Disciplina: Linguagem de Programação PASCAL **E-mail:** ***osorio@exatas.unisinos.br***
Professor responsável: ***Fernando Santos Osório*** **Web:**
Semestre: 2004/2 ***http://www.inf.unisinos.br/~osorio/prog2.html***
Horário: 63 **Xerox : *Pasta 54* – LAB. II (Xerox do "Alemão")**

Filas – FIFO = "First In, First Out"

As estruturas de dados podem ser organizadas de formas diferentes e especiais, de acordo com **a maneira com que os dados são inseridos e retirados**. Um tipo particular de estruturas de dados, é a **FILA**, onde a inserção ocorre sempre no final da fila (final do vetor) e a retirada dos dados ocorre do início da fila (início do vetor). Esta noção de início/fim do vetor pode ser um pouco particular se considerarmos que usualmente as filas serão implementadas em uma estrutura de dados chamada ***"fila circular"***, que descreveremos a seguir. Este tipo de estrutura de dados, as filas, podem "simular" uma fila de pessoas (do mundo real), sendo também muito usada em diversos tipos de aplicações computacionais, onde é importante "preservar" a ordem de chegada e de atendimento, como por exemplo, uma fila de impressão (nem todas filas são realmente filas, pois algumas permitem remover elementos do meio da fila, o que estaria fora do padrão), uma fila de dados a tratar, etc. Nas filas o primeiro dado a entrar nela é sempre o primeiro dado a sair (FIFO).

Manipulação de Fila => Entra = Insere no final da fila.
<= Sai = Retira do início da fila.

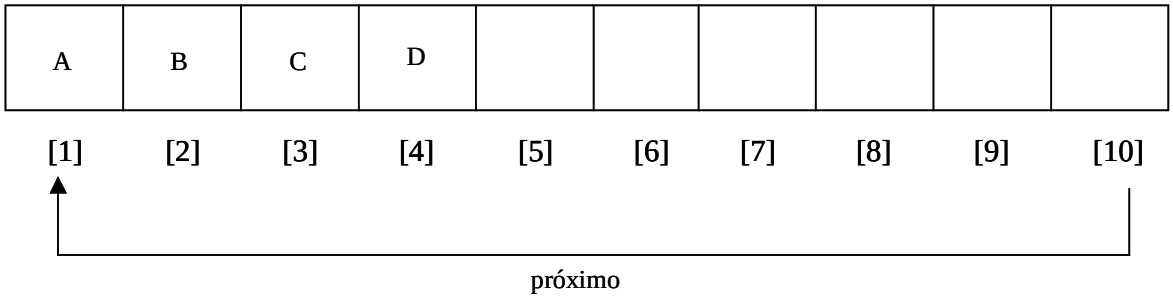


(a) Inserções na fila: A, B, C, D (b) Remoções da fila: A, B

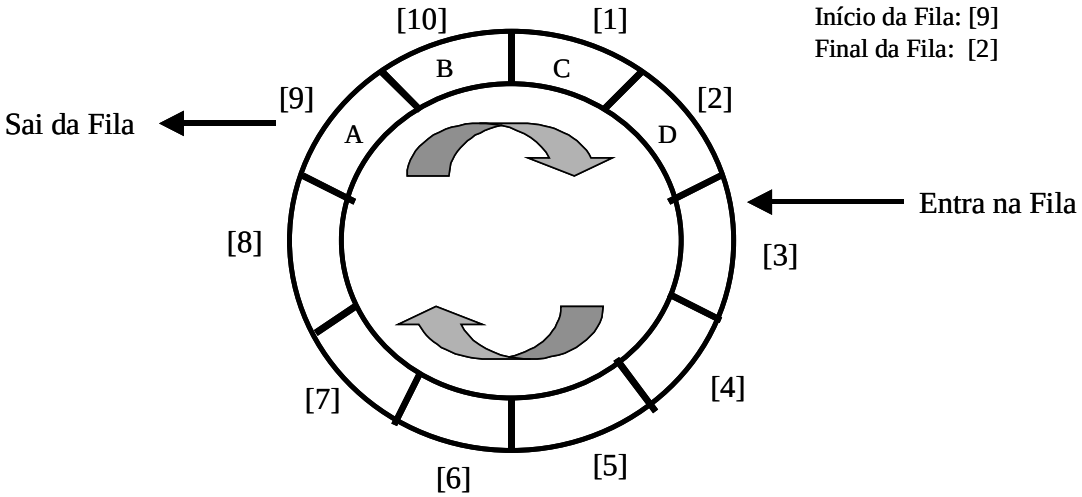
As filas podem usar vetores estáticos para armazenar os dados, mas como pode ser constatado na figura acima, se formos inserindo e retirando dados de um vetor estático, podemos rapidamente alcançar o "topo" deste vetor. Uma solução para este problema é a criação das chamadas FILAS CIRCULARES.

Uma FILA CIRCULAR é uma estrutura de dados que simula de modo virtual uma fila onde o início e o fim do vetor são unidos. A figura abaixo demonstra um exemplo de como poderia ser visualizada uma fila circular usando um vetor de 10 elementos:

Fila Circular => Fila: Array [1..10] of Integer; Início da Fila: [1] - Final da Fila: [4]



Fila Circular => Fila: Array [1..10] of Integer; Representação "Circular" do vetor



Portanto, em uma FILA CIRCULAR o elemento seguinte ao último elemento do vetor é virtualmente o primeiro elemento deste vetor, assim como, o elemento vizinho e anterior ao primeiro elemento deste vetor passa a ser o último elemento do vetor. Com isto, não teremos mais problema relativos ao deslocamento do início da fila que ocorre quando os elementos são retirados da fila. Podemos visualizar este comportamento da seguinte forma: à medida que o final da fila vai "subindo" no vetor em direção aos elementos de índice maior através da inserção de novos elementos, o início também irá "subindo" no vetor em direção os elementos de índice maior, "perseguindo" o final da fila. Se o final da fila alcançar o último elemento do vetor, e se o início do vetor não estiver mais ocupado, "damos a volta" e começamos a ocupar novamente o vetor a partir de seu início.

O uso de uma fila circular vai também implicar na necessidade de diferenciar entre fila completamente vazia (que pode ser vista como início da fila junto ao final da fila => início=fim) e fila completamente cheia (onde o final da fila "dá a volta" e alcança o início da fila => início=fim). Para resolver este problema devemos usar algum tipo de marcação que diferencie fila vazia de fila cheia: ***(a)*** usar um flag para sinalizar fila cheia, ***(b)*** manter atualizado um contador do número de elementos inseridos da fila, ou, ***(c)*** não permitir que o início "encoste" no final, deixando sempre uma posição do vetor livre (não ocupada).

Vamos definir a seguir um conjunto de **rotinas genéricas** para manipulação de filas circulares, que simplificarão o seu uso e sua adaptação para uso em outras aplicações.

Definição dos Dados:

```
Const
    FILA_TAM = 10;

Type
    TFila_Dado = Integer;
    TFila = Record
                Dado: Array [1..FILA_TAM] of TFila_Dado;
                Inicio, Fim: Integer;
                Qtde: Integer;
            End;
```

Definição das Rotinas:

Fila_Inicializa,
Fila_Insere,
Fila_Retira,
Fila_Exibe,
Fila_Esvazia, ...

```
Procedure Fila_Inicializa (Var F: TFila);
Begin
   F.Inicio := 1;
   F.Fim := 0;
   F.Qtde := 0;
End;

Function   Fila_Insere (Var F: TFila; Dado: TFila_Dado): Boolean;
Begin
   IF  F.Qtde = FILA_TAM
   THEN  Fila_Insere := False
   ELSE   Begin
              F.Fim := (F.Fim  MOD  FILA_TAM) + 1;    { Faz índice circular }
              F.Dado [F.Fim] := Dado;
              F.Qtde := F.Qtde + 1;
              Fila_Insere := True;
          End;

{   Sobre o "índice circular:
    F.Fim :=  (F.Fim  MOD  FILA_TAM ) + 1;
    poderia ser reescrito assim... que teria o mesmo resultado
    IF F.Fim = FILA_TAM
    THEN F.Fim := 1
    ELSE  F.Fim := F.Fim +1                                          }
End;
```

```
Function   Fila_Retira (Var F: TFila; Var Dado: TFila_Dado): Boolean;
Begin
    IF  F.Qtde = 0
    THEN  Fila_Retira := False
    ELSE   Begin
                Dado := F.Dado [ F.Inicio ];
                F.Inicio := ( F.Inicio  MOD  FILA_TAM ) + 1;
                F.Qtde := F.Qtde – 1;
                Fila_Retira := True;
              End;
End;
```

```
Procedure   Fila_Exibe (F: TFila);
Var
    cont, i: Integer;
Begin
     i := F.Inicio;
     FOR  cont := 1 to F.Qtde
     DO  Begin
               Writeln ( F.Dado [i] );
               i := ( i  MOD  FILA_TAM ) + 1
          End;
 End;
```

```
Procedure   Fila_Esvazia (F: TFila);
Var
    Dado: TFila_Dado;
Begin
     WHILE Fila_Retira(F,Dado)
     DO  ;
End;
```

Rotinas complementares...

```
Function    Fila_Vazia (F: TFila): Boolean;              { Testa para ver se a fila está vazia }
Function    Fila_Quantidade (F: TFila): Integer;         { Indica quantos dados tem na fila }
Procedure  Fila_Inverte (Var F: TFila);                  { Inverte a ordem dos dados da fila  }
Procedure  Fila_Grava (Var ArqTxt: Text; F: TFila);  {Salva em disco os dados}
Function    Fila_Le (Var ArqTxt: Text; Var F:TFila);  {Lê do disco os dados}
```

Exemplo de uso das rotinas...

```
Program Testa_Fila;

>>> Incluir rotinas de fila, {$I fila.pas }, ou  utilizar uma Unit fila.tpu: “Uses  fila;”  <<<

Var
   F: TFila;  Dado: TFila_Dado;
 Begin
    Dado:= 10;  Fila_Inicializa (F);  Fila_Insere (F, Dado);   {  ...   }
End.
```

EXERCÍCIOS:

1) Faça um programa que simule uma fila de impressão. O usuário do programa terá um menu com 4 opções: 1 = Insere arquivo na fila de impressão, 2 = Executa impressão, 3 = Exibe fila de impressão e 4 = Fim. Quando o usuário selecionar a opção nro. 1, ele deve fornecer o nome do arquivo que deseja imprimir e este arquivo será enviado para a fila de impressão, e ficará lá até que seja executada a sua impressão. Quando o usuário selecionar a opção nro. 2, o primeiro arquivo da fila de impressão será "virtualmente" encaminhado para a impressão, ou seja, será retirado da fila (você não precisa imprimir nada, apenas retire o pedido da fila de impressão). Se o usuário selecionar a opção 3, deve ser exibida na tela a fila, numerando de 1 a N junto com o nome do arquivo que está nesta posição da fila, onde 1 é o primeiro arquivo da fila e N é o último. A opção nro. 4 encerra a execução do programa. Use uma fila circular com capacidade de até 20 arquivos, com nomes de no máximo 30 caracteres. Vide abaixo um exemplo de interação:

```
>> Fila de Impressão <<

1 - Insere arquivo na fila de impressão
2 - Executa impressão
3 – Exibe o estado da fila de impressão
4 – Fim

Qual a sua opção: 1
Nome do arquivo: Teste1

Qual a sua opção: 1
Nome do arquivo: Teste2

Qual a sua opção: 3
Fila de Impressão:
1 – Teste1
2 – Teste2

Qual a sua opção: 2
Arquivo sendo impresso: Teste1

Qual a sua opção: 3
Fila de impressão:
1 – Teste2

Qual a sua opção: 4
>> FIM <<
```

2) Faça um programa que simule um banco, onde este banco possui 3 caixas (três filas de atendimento de clientes), com capacidade de até 10 clientes em cada fila. Simule a chegada de até 50 clientes neste banco que irão entrar nas filas, sendo que a decisão de qual fila cada cliente irá entrar será feita usando um gerador de números pseudo-aleatórios – "rand(3)", onde o valor sorteado poderá ser: 0 = entra na fila 1, 1 = entra na fila 2, e 2 = entra na fila 3. A cada 3 clientes novos que entram nas filas, um cliente deve ser atendido, usando também um gerador de números pseudo-aleatórios para decidir qual a fila que vai avançar (ser atendida). Se ao tentar entrar em uma fila esta estiver lotada, exibir na tela a mensagem "Fila X: Lotada" (onde X é o nro. da fila), e depois tente novamente sortear uma nova fila e colocar o cliente nesta fila. Se ao tentar atender um cliente de uma determinada fila esta fila estiver vazia, exibir na tela a mensagem "Fila X: Vazia". Ao final da execução do programa (quando todos os 50 clientes já entraram em alguma fila), exiba quantos clientes restaram em cada fila.