UFSM
Universidade Federal de Santa Maria

Engenharia de CONTROLE e AUTOMAÇÃO

Filas

Aula 02

DPEE 1038 – Estrutura de Dados para Automação
Curso de Engenharia de Controle e Automação
Universidade Federal de Santa Maria

Prof. Rafael Concatto Beltrame
beltrame@mail.ufsm.br

Sumário

- **Operações primitivas**
  - **Operação empty ( )**
  - **Operação remove ( )**
  - **Operação insert ( )**

Operações Primitivas

- **Seja uma fila definida pela seguinte estrutura em C**
  - **Um vetor para armazenar os elementos da fila**
  - **Duas variáveis para indicar a posição atual do primeiro e do último elemento da fila**

```
#define MAXQUEUE 5          // Dimensão da fila (5)

struct queue {
   int front, rear;         // End. inicial e final
   int items[MAXQUEUE];     // Vetor de dados
};

struct queue q;             // Declaração da fila "q"
```

Operações Primitivas

remove ( )

- Usada para remover um item do início da fila
- Sintaxe: `x = remove(&q);`
- Deve-se evitar o underflow
- **Primeira tentativa**
  - Supondo: q.front = 0 e q.rear = -1

```
// Oper. remove (ignorando a possib. de underflow)
x = q.items[q.front++];
          // Retira o dado q.items[q.front] (início)
          // Incrementa q.front (novo início)
```

Operações Primitivas

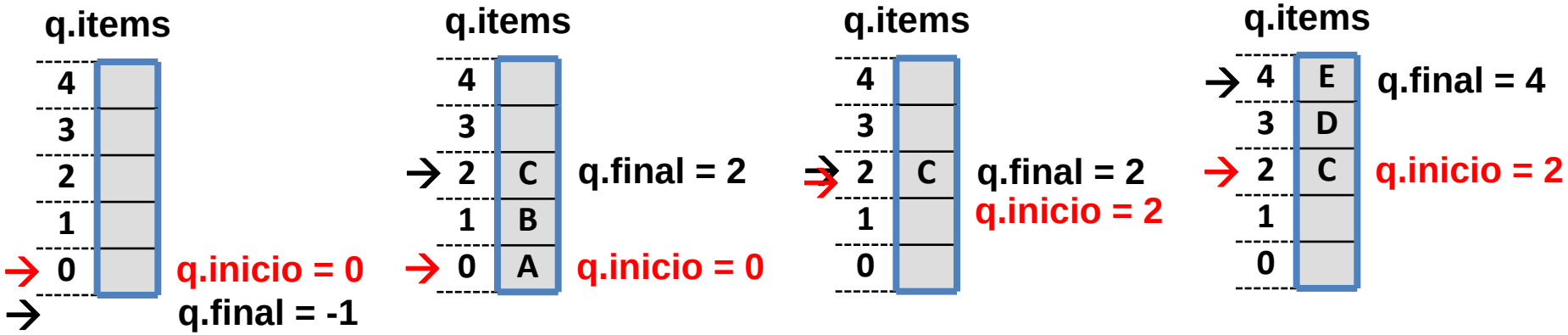
- Nesse caso, o número de elementos é sempre

$$n = (q.rear - q.front) + 1$$

- E a fila estará vazia se

$$q.rear < q.front$$

Problema:
Impossibilidade de inserir novos dados na fila, mesmo com espaços desocupados no vetor



Operações Primitivas

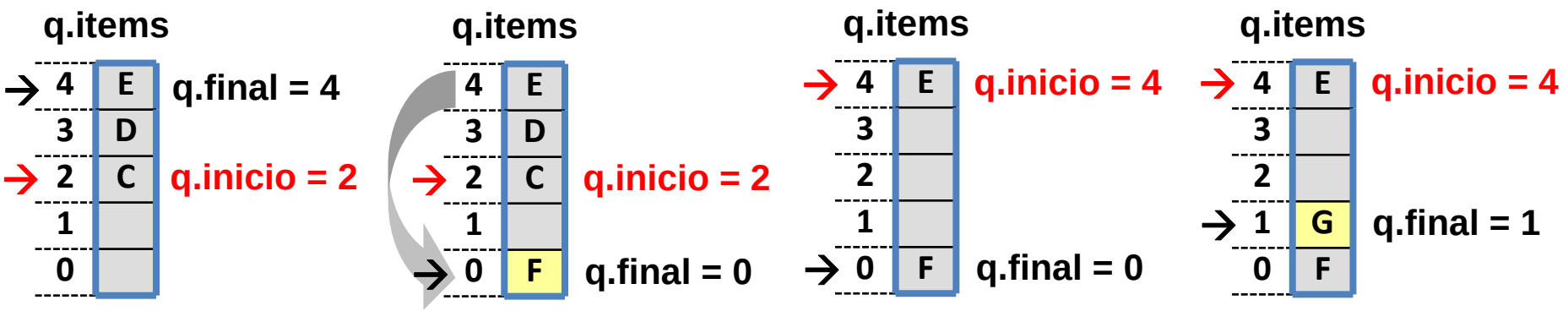
- **Segunda tentativa**
  - Deslocar todos os elementos da fila quando o primeiro item fosse removido → Análogo a "fazer a fila andar"

```
// Operação remove
x = q.items[0];                    // Retira dado início
for (i = 0; i < q.rear; i++)       // Desloca fila no
   q.items[i] = q.items[i+1]; // sentido do inicio
q.rear--;                          // Decrementa final
```

  - A fila não precisa conter o campo front → sempre 0
  - **Método ineficiente:** cada eliminação envolve deslocar todos os elementos restantes da fila

Operações Primitivas

- **Terceira tentativa**
  - Visualizar o vetor como um círculo e não como uma linha reta
  - O primeiro elemento do vetor (0) segue seu último elemento

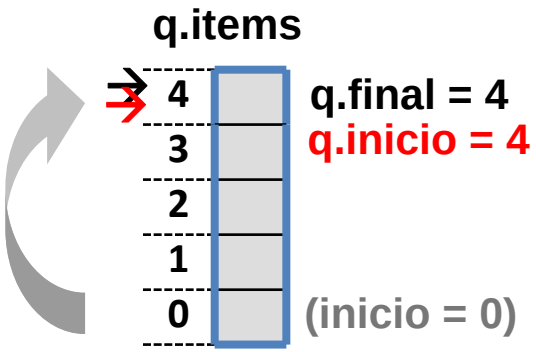


  - Observar que o teste $q.rear < q.front$ não é mais válido!
  - **Problema:** como determinar se a fila está vazia?
    (a implementação completa será mostrada a seguir)

Operações Primitivas

empty ( )

- Usada para testar se a fila está vazia
- Sintaxe: `y = empty(&q);`
- Nova convenção
  - q.front é o índice do elemento imediatamente anterior ao primeiro
  - q.rear é o índice do último elemento da fila



Se, inicialmente: q.front = q.rear = MAXQUEUE – 1

A fila estará vazia se

$$q.rear == q.front$$

Operações Primitivas

- **Implementação da função empty**

```
// Função empty

empty(pnt_q)
   struct queue *pnt_q;      // Entrada como ponteiro
{                            // Teste condicional
   if (pnt_q -> front == pnt_q -> rear)
      return(TRUE);          // Fila vazia
   else
      return(FALSE);         // Fila não vazia
}
```

```
if (empty(&q)) { comandos } // Exemplo de uso
else           { comandos }
```

Operações Primitivas

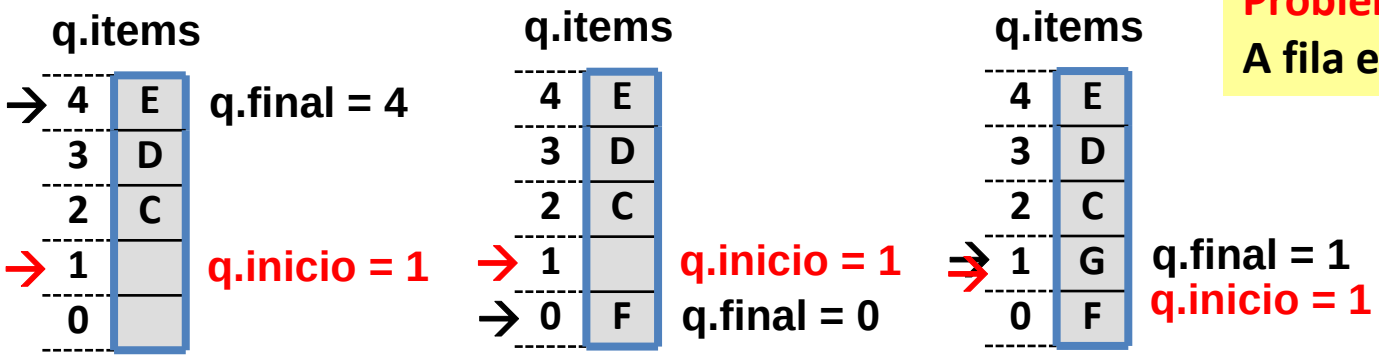
- **Implementação final da função remove**

```
// Função remove

remove(pnt_q)
   struct queue *pnt_q;     // Entrada como ponteiro
{
   if(empty(pnt_q)) {       // Teste de fila vazia
      printf("Underflow na fila");
      exit(1);
   }                        // Atualização de front
   if (pnt_q -> front == MAXQUEUE - 1)
      pnt_q -> front = 0;
   else
      (pnt_q -> front)++;

   return(pnt_q -> items[pnt_q -> front]);
}
```

Operações Primitivas

insert ( )

- Usada para inserir um item no final da fila
- Sintaxe: `insert(&q, x);`
- A operação insert involve um teste de estouro (overflow)
  - Se o vetor estiver cheio e tentarmos inserir um novo item



Operações Primitivas

- **Solução: não usar todos os elementos do vetor**
  - **Exemplo: um vetor com 100 elementos será usado para uma fila de 99 elementos**
    - **A tentativa de inserir o 100° elemento provocará um estouro (overflow)**
  - **O teste para evitar o underflow (empty) devido ao uso da função remove, continua o mesmo**

Operações Primitivas

```
// Função insert
insert(pnt_q, x)
   struct queue *pnt_q;     // Entrada como ponteiro
   int x;                   // Novo dado
{
   // Espaço para um novo elemento
   if (pnt_q -> rear == MAXQUEUE - 1)
      pnt_q -> rear = 0;    // Último passa a ser o 1°
   else
      (pnt_q -> rear)++;
   // Verifica ocorrência de estouro
   if (pnt_q -> rear == pnt_q -> front) {
      printf("Overflow na fila");
      exit(1);
   }
   // Retorna por ponteiro
   pnt_q -> items[pnt_q -> rear] = x;
   return;
}
```