Algoritmos e Estrutura de Dados

Filas

Prof. Nilton Luiz Queiroz Jr.

Filas

- Filas são um tipo abstrato de dados onde os dados estão em sequência, e que se permite a inserção em uma extremidade, e eliminação em outra;
- São usadas em:
  - Atendimento de requisições em banco de dados;
  - Emissão de instruções para o processamento;
  - Alocação de recursos para impressão de documentos em uma impressora;
  - etc;

Filas

- Alguns modelos intuitivos de Filas:
  - Fila de pessoas;
  - Fila de carros;
  - etc;

Filas

- Filas obedecem a política FIFO (first in, first out);
  - O primeiro a entrar é o primeiro a sair;
- Toda inserção é feita no fim da fila;
- Toda remoção é feita no início da fila;

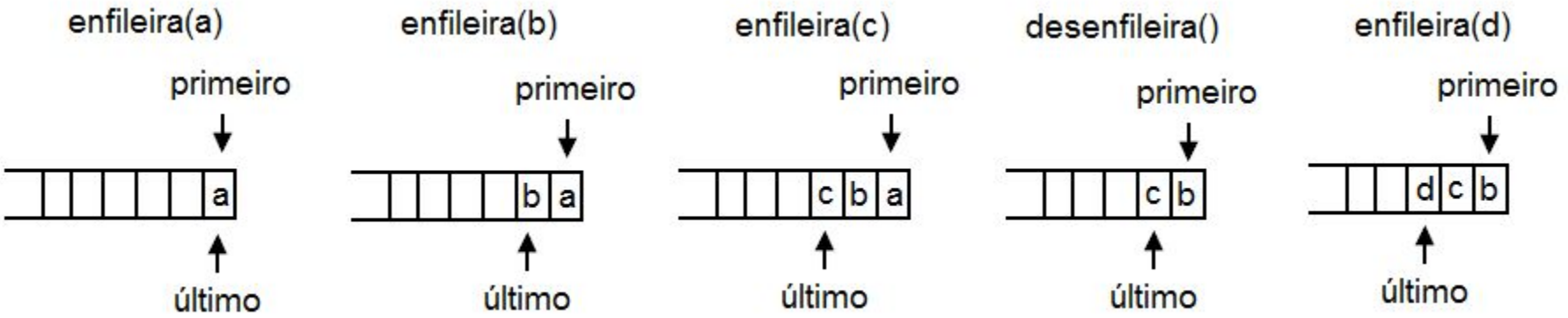
Filas

- Filas são usadas para processar dados que tem necessidade de serem processados em ordem de chegada;
- O item com maior tempo de permanência na fila será o que tem maior prioridade a sair;
    - Análogas às filas que usamos em nosso dia a dia;

Filas

- Toda fila tem duas operações básicas:
  - Enfileirar;
    - Inserir um elemento no fim da fila;
  - Desenfileirar;
    - Remover o elemento do início da fila;

Filas


enfileira(a)
primeiro
a
último
enfileira(b)
primeiro
b a
último
enfileira(c)
primeiro
c b a
último
desenfileira()
primeiro
c b
último
enfileira(d)
primeiro
d c b
último

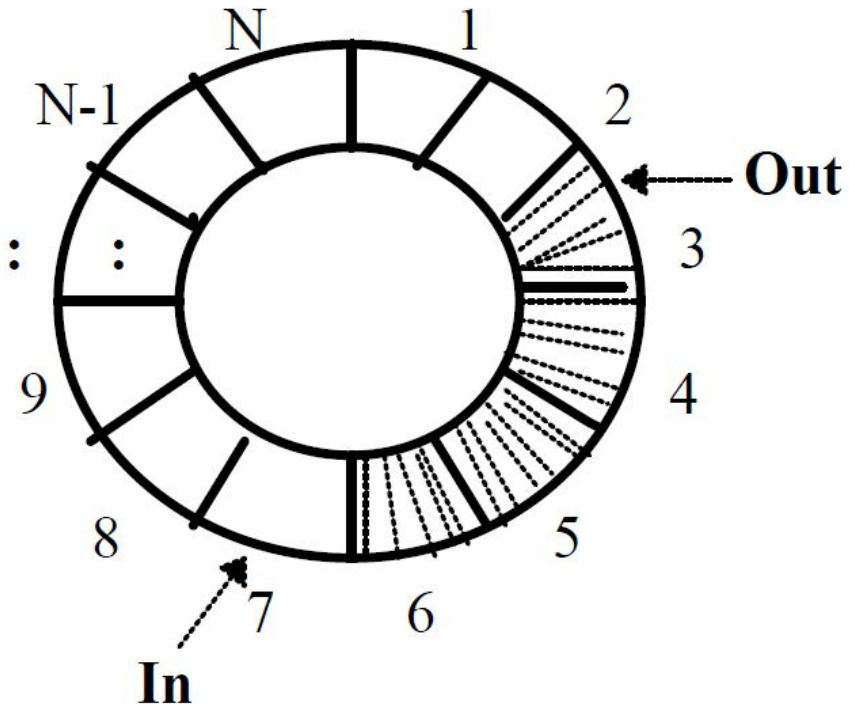
Filas

- As filas podem ser implementadas das seguintes maneiras:
  - Estáticas;
  - Dinâmicas

Filas Estáticas

- As filas estáticas podem ser implementadas:
  - Com deslocamento;
    - Sempre se remove na posição 1;
      - A cada remoção reposicionam-se todos os elementos;
        - O custo da remoção depende da quantidade de elementos na fila;
    - Sempre se insere na posição t+1, onde t é o número de elementos na fila;
  - Circular;
    - Em um vetor de N posições a posição 1 é a sucessora da posição N;
    - Usam-se variáveis para controlar o local de inserção e remoção;
      - Essas variáveis devem ser incrementadas de maneira a não ultrapassar o valor de N;
        - primeiro := (primeiro mod N) +1;
      - Remoções em tempo constantes;

Filas Estáticas



Filas estáticas circulares

- Devemos implementar as seguintes operações:
  - Inicializar a fila;
  - Verificar se a fila está vazia;
  - Verificar se a fila está cheia;
  - Inserção;
    - Enfileirar;
  - Remoção;
    - Desenfileirar;
  - Obter o primeiro da fila;
- Em todas as operações deve-se tomar cuidado com o fato de que o sucessor da última posição é a primeira posição;
- Além disso precisamos das estruturas e o tamanho máximo;

Filas estáticas circulares

- Estruturas

```
#define TAM 51 // a fila irá armazenar no máximo 50 elementos;

struct tipo_item{
        int chave;
};

struct tipo_fila{
        int primeiro,ultimo;
        struct tipo_item dados[TAM];
};
```

Filas estáticas circulares

- Pelo modo que uma fila estática circular se comporta é interessante usar uma função que obtém a próxima posição;
- Supondo uma fila que armazenará no máximo TAM-1 elementos, a função de obter o próximo é a seguinte:

```
int proximo(int pos){
        return (pos+1)%TAM;
}
```

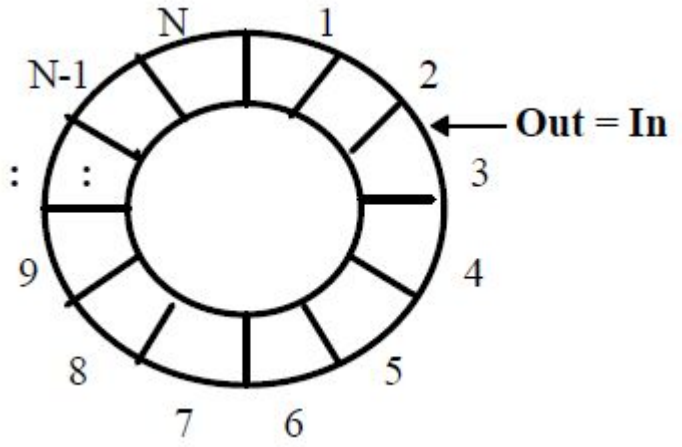
Filas estáticas circulares

- Para inicializar uma fila estática circular devemos colocar o inicio e o fim na posição 0 do vetor;

```
void inicializa(struct tipo_fila *f){
        f->primeiro=0;
        f->ultimo=0;
}
```

Filas estáticas circulares

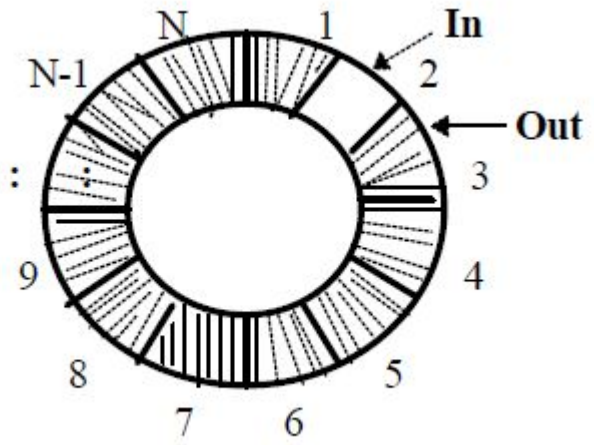
- Para verificar se a fila está vazia devemos ver se o primeiro e o último estão na mesma posição;




```
int vazia(struct tipo_fila *f){
        return f->primeiro==f->ultimo;
}
```

Filas estáticas circulares

- Para verificar se a fila está cheia devemos ver se o próximo do último é o primeiro




```
int cheia(struct tipo_fila *f){
        return proximo(f->ultimo)==f->primeiro;
}
```

Filas estáticas circulares

- Inserção em uma fila estática circular:
  - Lembre-se que ao inserir na última posição, a variável que guarda o fim da fila deverá valer 0;

```
int insere_fila(struct tipo_fila *f, struct tipo_item x){
        /*algoritmo*/
}
```

Filas estáticas circulares

- Inserção em uma fila estática circular:
  - Lembre-se que ao inserir na última posição, a variável que guarda o fim da fila deverá valer 0;

```
int insere_fila(struct tipo_fila *f, struct tipo_item x){
        if(!cheia(f)){
                f->dados[f->ultimo]=x;
                f->ultimo=proximo(f->ultimo);
                return 1;
        }else{
                return 0;
        }
}
```

Filas estáticas circulares

- Remoção na fila

```
int remove_fila(struct tipo_fila *f, struct tipo_item *x){
        /*algoritmo*/
}
```

Filas estáticas circulares

- Remoção na fila

```
int remove_fila(struct tipo_fila *f, struct tipo_item *x){
        if(!vazia(f)){
                *x=f->dados[f->primeiro];
                f->primeiro=proximo(f->primeiro);
                return 1;
        }else{
                return 0;
        }
}
```

Filas estáticas circulares

- Escrever todos elementos da fila
  - Lembre-se que nem sempre a posição do primeiro será menor que a do último
  - Vale lembrar também que o último elemento da fila está uma posição atrás da variável último

```
void escreve(struct tipo_fila *f){
        /*algoritmo*/
}
```

Filas estáticas circulares

- Escrever todos elementos da fila
  - Lembre-se que nem sempre a posição do primeiro será menor que a do último
  - Vale lembrar também que o último elemento da fila está uma posição atrás da variável último

```
void escreve(struct tipo_fila *f){
        int i;
        printf("=================\n");
        for(i=f->primeiro; i!=f->ultimo; i=proximo(i)){
                printf("%d ",f->dados[i].chave);
        }
        printf("\n");
        printf("=================\n");
}
```

Filas Dinâmicas

- As filas dinâmicas são implementadas em listas ligadas;
- Devemos implementar as seguintes operações:
  - Inicializar a fila;
  - Verificar se a fila está vazia;
  - Inserção;
    - Enfileirar;
  - Remoção;
    - Desenfileirar;
  - Obter o primeiro da fila;
- Além disso é necessário as estruturas da fila;

Filas Dinâmicas

- Estruturas da fila dinâmica

```
struct tipo_item{
    int chave;
};
struct tipo_celula{
    struct tipo_item item;
    struct tipo_celula *prox;
};
struct tipo_fila{
    struct tipo_celula *primeiro;
    struct tipo_celula *ultimo;
};
```

Filas Dinâmicas

- Inicializar

```
void inicializa(struct tipo_fila *f){
    f->primeiro=(struct tipo_celula *)malloc(sizeof(struct tipo_celula ));
    f->ultimo=f->primeiro;
    f->ultimo->prox=NULL;
}
```

Filas Dinâmicas

- Verificar se a fila está vazia

```
int vazia(struct tipo_fila *f){
    return f->primeiro->prox == NULL;
}
```

Filas Dinâmicas

- Inserir

```
void insere_fila(struct tipo_fila *f, struct tipo_item x){
    struct tipo_celula *novo;
    novo=(struct tipo_celula *)malloc(sizeof(struct tipo_celula));
    novo->prox=NULL;
    novo->item=x;
    f->ultimo->prox=novo;
    f->ultimo=novo;
}
```

Filas Dinâmicas

- Remover

```
int remove_fila(struct tipo_fila *f, struct tipo_item *x){
    struct tipo_celula *ptr;
    if(!vazia(f)){
        ptr=f->primeiro->prox;
        f->primeiro->prox=ptr->prox;
        *x=ptr->item;
        free(ptr);
        if(vazia(f)){
            f->ultimo=f->primeiro;
        }
        return 1;
    }else{
        return 0;
    }
}
```

Exercícios

1. Faça uma função que receba uma fila e a inverta. Só é permitido acessar os elementos da fila com operações de fila e a única variável que pode ser declarada é uma variável do tipo item.

```
void inverte(struct tipo_fila *f){//
        struct tipo_item x;//única variável que pode ser declarada na função
        //algoritmo para inverter a fila f
}
```

   Dica: Use recursividade.

2. Faça uma operação para obter o primeiro elemento de uma fila para:
   a. Uma fila estática circular;
   b. Uma fila dinâmica usando sentinela;

Exercícios

3. Altere as estruturas e as operações de fila para que a fila armazene seu tamanho em um campo para
    a. Uma fila estática circular;
    b. Uma fila dinâmica usando sentinela;
4. Faça uma função para copiar os dados de uma fila para outra. Use apenas filas auxiliares e operações sobre fila.
5. Faça uma função para esvaziar uma fila. Essa função deve ser escrita de forma que irá funcionar tanto para filas estáticas quanto para filas dinâmicas.
6. Utilizando apenas operações de fila, faça uma função que receba um campo de um elemento da fila e descubra quantos elementos se encontram em sua frente na fila.

Referências

CELES, W; RANGEL, J. L. Apostila de estrutura de dados. PUC-Rio, 2002.

GONÇALVES, R. A. L; Algoritmos e estrutura de dados: Técnicas de programação usando a linguagem pascal.