

# Estrutura de Dados

Hamilton José Brumatto

Bacharelado em Ciências da Computação - UESC

26 de maio de 2016

## Filas

## Fila:

- Um conjunto ordenado de itens
- Novos itens podem ser inseridos em uma extremidade: *Final* e excluídos da outra extremidade: *Início*
- Fila é uma estrutura de dados dinâmica: a operação de inserir e remover itens faz parte da estrutura.
- A representação de uma fila é um sequência de objetos, um objeto entra na fila, sendo inserido no final, o objeto que sai da fila é retirado de seu início.

## Exemplo de Pilha e operações:

4	7	
---	---	--

- O que determina o estado da fila é o conjunto de objetos lá existentes.
- A operação que remove um item não especifica qual item será removido, pois será o item que está no início da fila.

## Exemplo de Pilha e operações:

4	7	3	
---	---	---	--

Inserindo 3

- O que determina o estado da fila é o conjunto de objetos lá existentes.
- A operação que remove um item não especifica qual item será removido, pois será o item que está no início da fila.

## Exemplo de Pilha e operações:

4	7	3	5	
---	---	---	---	--

Inserindo 5

- O que determina o estado da fila é o conjunto de objetos lá existentes.
- A operação que remove um item não especifica qual item será removido, pois será o item que está no início da fila.

## Exemplo de Pilha e operações:

7	3	5	
---	---	---	--

Removendo

- O que determina o estado da fila é o conjunto de objetos lá existentes.
- A operação que remove um item não especifica qual item será removido, pois será o item que está no início da fila.

## Exemplo de Pilha e operações:

7	3	5	2	
---	---	---	---	--

Inserindo 2

- O que determina o estado da fila é o conjunto de objetos lá existentes.
- A operação que remove um item não especifica qual item será removido, pois será o item que está no início da fila.



## Exemplo de Pilha e operações:

7	3	5	2	8	
---	---	---	---	---	--

Inserindo 8

- O que determina o estado da fila é o conjunto de objetos lá existentes.
- A operação que remove um item não especifica qual item será removido, pois será o item que está no início da fila.

## Exemplo de Pilha e operações:

3	5	2	8	
---	---	---	---	--

Removendo

- O que determina o estado da fila é o conjunto de objetos lá existentes.
- A operação que remove um item não especifica qual item será removido, pois será o item que está no início da fila.

## Exemplo de Pilha e operações:

5	2	8	
---	---	---	--

Removendo

- O que determina o estado da fila é o conjunto de objetos lá existentes.
- A operação que remove um item não especifica qual item será removido, pois será o item que está no início da fila.

## Enfileirar / Desenfileirar

- Duas operações são realizadas sobre a fila:
  - A operação de inserir na fila é: *Enfileirar(Obj)*.
  - A operação de remover da pilha é: *Desenfileirar()*  $\rightarrow$  *Obj*
- Estas operações são conhecidas, também, como *Enqueue(Obj)* e *Dequeue()*, respectivamente.
- Nos slides anteriores, a sequência de operações foram:  
Enfileirar(3)  
Enfileirar(5)  
Desenfileirar()  $\rightarrow$  4  
Enfileirar(2)  
Enfileirar(8)  
Desenfileirar()  $\rightarrow$  7  
Desenfileirar()  $\rightarrow$  3

## FILO ou LIFO

- Dada as operações sobre a fila e como são inseridos os objetos, tem-se:
- O primeiro objeto a ser inserido será o primeiro a ser removido.
- *FIFO* - First In, First Out.
- Ou então, o último objeto que foi inserido na pilha será o último a ser removido.
- *LIFO* - Last In, Last Out. (pouco usado)

## Aplicações de Filas

- Filas de escalonamento de processos: Um processo por vez usa o processador.
- Filas de lotes batchs de processamento. Por exemplo: Slurm no Cluster Cacau
- Spool de impressão. Impressoras compartilhadas
- Fila de instruções na microarquitetura e em algoritmos superescalares.
- Fila de pacotes numa transmissão de rede...

## Implementando Fila

- A implementação de fila necessita de dois marcadores: Início e Fim.
- O ponteiro início aponta para o primeiro elemento da fila.
- O ponteiro final aponta para a primeira posição livre para inserir elemento.
- É mais prático andar os marcadores do que andar com os dados no vetor.
- O vetor é usado de forma circular:
- A implementação de pilha em vetores também precisa de dois marcadores: vazia e cheia.
- Nas duas situações, a relação entre Início e Fim são idênticas, logo são necessários os marcadores para diferenciar as operações permitidas e proibidas.

## Implementando Fila

- As funções para a fila:
  - A função *Enfileirar(fila,obj)*: Insere um objeto no final e atualiza a posição do final da fila.
  - O ponteiro de final de fila é incrementado, chegando no final do vetor, ele retoma o início.
  - A função *Desenfileirar(fila) → obj*: Remove um objeto do início da fila e atualiza a posição do início.
  - O ponteiro de início de fila é incrementado, chegando no final do vetor, ele retoma o início.
  - Se o ponteiro de início atingir a posição do ponteiro de final da fila, a fila está vazia.
  - Se o ponteiro de final de fila atingir a posição do ponteiro de início da fila, a fila está cheia.
  - A função *ehVazia* não é necessária, por exemplo, quando se utiliza um modelo de captura de exceções. Tentar uma operação *Desenfileirar* em uma fila vazia poderia gerar uma exceção.



## Vetores

### Estrutura e tipos para a fila

```
#define TAMANHOFILA 100
typedef int obj_t;
typedef int boolean;
enum{falso, verdade};

typedef struct fila {
    obj_t itens[TAMANHOFILA];
    int inicio, fim;
    boolean cheia, vazia; } fila;

boolean enfileirar(fila *p, obj_t obj);
obj_t desenfileirar(fila *p);
boolean ehvazia(fila p);
```

## Operações: Enfileirar

```
boolean enfileirar(fila *p, obj_t obj) {  
    boolean ret = falso;  
    if(!p->cheia) {  
        p->itens[p->fim++ -1]=obj;  
        if(p->fim > TAMANHOFILA) p->fim=1;  
        if(p->inicio == p->fim) p->cheia=verdade;  
        p->vazia=falso;  
        ret = verdade;  
    }  
    return ret;  
}
```

## Operações: Desenfileirar e ehVazia

```
obj_t desenfileirar(fila *p) {  
    obj_t o;  
    assert(!p->vazia);  
    o = p->itens[p->inicio++ - 1];  
    if(p->inicio > TAMANHOFILA) p->inicio=1;  
    if(p->inicio == p->fim) p->vazia=verdade;  
    p->cheia=falso;  
    return o;  
}  
  
boolean ehvazia(fila p) { return (p.vazia);  
}
```

## Testando

```
int main(int argc, char **args) {  
    fila f;  
    f.inicio = f.fim = 1;  
    f.cheia=falso;  
    f.vazia=verdade;  
    obj_t o;  
    enfileirar(&f,5); status(f);  
    enfileirar(&f,3); status(f);  
    enfileirar(&f,2); status(f);  
    o = desenfileirar(&f); printf("%d -- > ",o); status(f);  
    enfileirar(&f,4); status(f);  
    o = desenfileirar(&f); printf("%d -- > ",o); status(f);  
    o = desenfileirar(&f); printf("%d -- > ",o); status(f);  
    enfileirar(&f,8); status(f);  
    while(!ehvazia(f)) desenfileirar(&f); status(f);  
    return 0;  
}
```

## Testando

```
void status(fila f) {  
    int i=0;  
    printf("%d : ",f.inicio);  
    i=f.inicio-1;  
    if(!ehvazia(f)) {  
        do {  
            printf("%d ",f.itens[i]);  
            i++;  
            if(i==TAMANHOFILA) i=0;  
        } while(i!=f.fim-1);  
    }  
    printf(": %d\n",f.fim);  
    return;  
}
```

## Saída

```
1 : 5 : 2
1 : 5 3 : 3
1 : 5 3 2 : 4
5 --> 2 : 3 2 : 4
2 : 3 2 4 : 5
3 --> 3 : 2 4 : 5
2 --> 4 : 4 : 5
4 : 4 8 : 6
6 : : 6
```

Saída — > #DEFINE TAMANHOFILA 3

```
1 : 5 : 2
1 : 5 3 : 3
1 : 5 3 2 : 1
5 --> 2 : 3 2 : 1
2 : 3 2 4 : 2
3 --> 3 : 2 4 : 2
2 --> 1 : 4 : 2
1 : 4 8 : 3
3 : : 3
```

## Implementando via Lista Ligada

- Como os itens são enfileirados de um lado e desenfileirados de outro, a melhor solução é uma lista duplamente ligada, sendo que a cabeça aponta para o início da fila e a cauda, o fim.
- Lista Ligada oferece duas vantagens de não haver limite de enfileiramento (exceto o próprio limite de memória)
- A pilha está vazia quando a cabeça e cauda são null



## Tipos e Protótipos

```
typedef int obj_t;
```

```
typedef struct fila {  
    obj_t item;  
    struct fila *ant, *prox;  
} fila;
```

```
void void enfileirar(fila **inicio, fila **fim, obj_t obj);  
obj_t desenfileirar(fila **inicio, fila **fim);
```

## Operações

```
void enfileirar(fila **inicio, fila **fim, obj_t obj) {  
    fila *np;  
    np = malloc(sizeof(fila));  
    np->item = obj;  
    np->ant = (*fim);  if(np->ant != NULL) np->ant->prox = np;  
    np->prox = NULL;  (*fim) = np;  
    if(*inicio == NULL) (*inicio) = np;  
    return;  
}
```

```
obj_t desenfileirar(fila **inicio, fila **fim) {  
    obj_t o;  
    fila *d;  
    assert((*inicio) != NULL);  
    o = (*inicio)->item;  
    d = (*inicio);  
    if(*inicio == *fim) (*fim = NULL);  
    else (*inicio->prox->ant) = NULL;  
    (*inicio) = (*inicio)->prox;  
    free(d);  
    return o;  
}
```

## Testando

```
int main(int argc, char **args) {  
    fila *inicio, *fim;  
    inicio = fim = NULL;  
    obj_t o;  
    enfileirar(&inicio,&fim,5); status(inicio);  
    enfileirar(&inicio,&fim,3); status(inicio);  
    enfileirar(&inicio,&fim,2); status(inicio);  
    o = desenfileirar(&inicio,&fim); printf("%d -- > ",o); status(inicio);  
    enfileirar(&inicio,&fim,4); status(inicio);  
    o = desenfileirar(&inicio,&fim); printf("%d -- > ",o); status(inicio);  
    o = desenfileirar(&inicio,&fim); printf("%d -- > ",o); status(inicio);  
    enfileirar(&inicio,&fim,8); status(inicio);  
    while(inicio!=NULL) desenfileirar(&inicio,&fim); status(inicio);  
    return 0;  
}
```

## Testando

```
void status(fila *f) {  
    while(f != NULL) {  
        printf("%d ", f->item);  
        f = f->prox;  
    }  
    printf(".\n");  
    return;  
}
```

## Saída

```
5 .  
5 3 .  
5 3 2 .  
5 --> 3 2 .  
3 2 4 .  
3 --> 2 4 .  
2 --> 4 .  
4 8 .  
.
```

## Filas com prioridades

- Fila com prioridades representa uma sequência ordenada e classificada.
- O elemento que é retirado da fila sempre é o elemento de maior prioridade.
- Se dois elementos de mesma prioridade estão para ser retirado, primeiro é retirado aquele que entrou primeiro.
- Implementação: duas opções.
  - Classificar a fila na entrada (insert sort)
  - Retirar o elemento de maior prioridade (busca sequencial)

## Comparando as implementações

- Classificar na entrada:
  - Inserir o elemento implica em classificar a sequência.
  - Na média é necessário mover 50% da fila para encaixar o elemento. (Pior caso: custo  $n$ )
  - Retirar o elemento representa retirar o elemento do início da fila.
- Retirar o de maior prioridade:
  - Inserir o elemento representa inserir no fim da fila.
  - Para retirar o elemento é necessário verificar toda a fila para achar o elemento de maior prioridade (Sempre é custo  $n$ ).

Realizar a lista 6 de exercícios