

# Estruturas de Dados

## Módulo 12 - Filas



# Referências

Waldemar Celes, Renato Cerqueira, José Lucas Rangel,  
*Introdução a Estruturas de Dados*, Editora Campus  
(2004)

Capítulo 12 – Filas

# Tópicos

- Introdução
- Interface do tipo fila
- Implementação de fila com vetor
- Implementação de fila com lista
- Fila dupla
- Implementação de fila dupla com lista

# Introdução

- Fila
  - um novo elemento é inserido no final da fila e um elemento é retirado do início da fila
    - fila = “o primeiro que entra é o primeiro que sai” (FIFO)
    - pilha = “o último que entra é o primeiro que sai” (LIFO)

# Interface do tipo fila

- Implementações:
  - usando um vetor
  - usando uma lista encadeada
  - simplificação:
    - fila armazena valores reais

# Interface do tipo fila

- Interface do tipo abstrato Fila: *fila.h*
  - função *fila\_cria*
    - aloca dinamicamente a estrutura da fila
    - inicializa seus campos e retorna seu ponteiro
  - função *fila\_insere* e função *fila\_retira*
    - insere e retira, respectivamente, um valor real na fila
  - função *fila\_vazia*
    - informa se a fila está ou não vazia
  - função *fila\_libera*
    - destrói a fila, liberando toda a memória usada pela estrutura

```
typedef struct fila Fila;
```

```
Fila* fila_cria (void);
```

```
void fila_insere (Fila* f, float v);
```

```
float fila_retira (Fila* f);
```

```
int fila_vazia (Fila* f);
```

```
void fila_libera (Fila* f);
```

tipo Fila:

- definido na interface
- depende da implementação do struct fila

# Implementação de fila com vetor

- Implementação de fila com vetor
  - vetor (vet) armazena os elementos da fila
  - estrutura de fila:

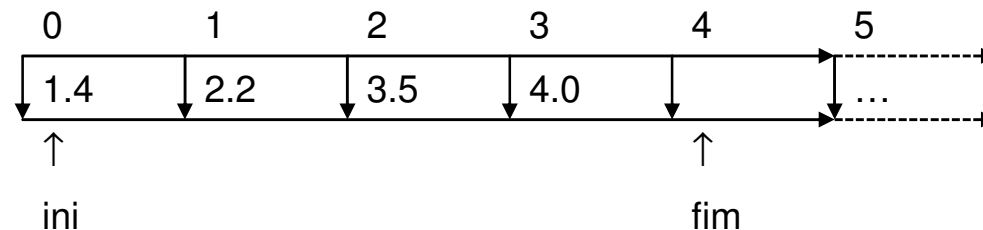
```
#define N 100      /* número máximo de elementos */

struct fila {
    int n;          /* número de elementos na fila */
    int ini;        /* posição do próximo elemento a ser retirado da fila */
    float vet[N];
};
```

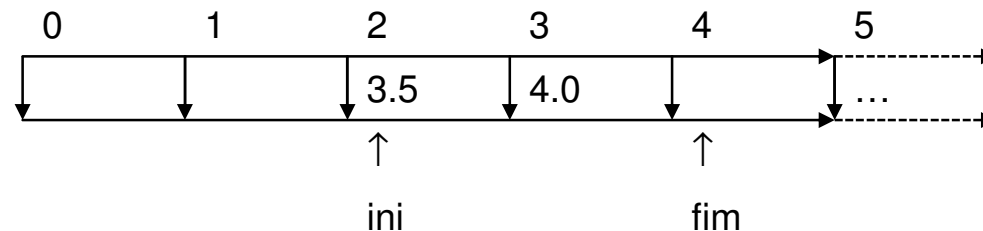


# Implementação de fila com vetor

- Implementação de fila com vetor
  - processo de inserção e remoção em extremidades opostas da fila faz com que a fila “ande” no vetor
    - inserção dos elementos 1.4, 2.2, 3.5, 4.0

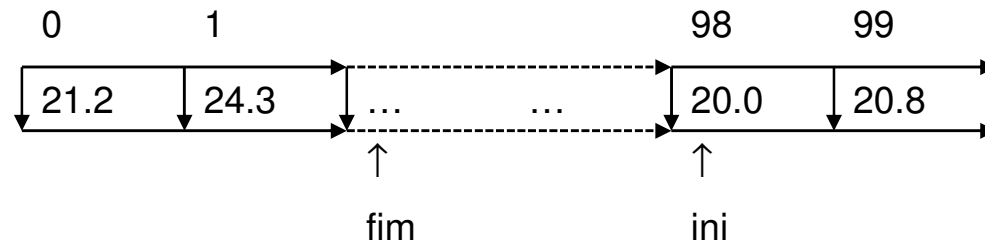


- remoção de dois elementos



# Implementação de fila com vetor

- Implementação de fila com vetor
  - incremento das posições do vetor de forma “circular”:
    - se o último elemento da fila ocupa a última posição do vetor, os novos elementos são inseridos a partir do início do vetor
    - exemplo:
      - quatro elementos, 20.0, 20.8, 21.2 e 24.3
      - distribuídos dois no fim do vetor e dois no início



# Implementação de fila com vetor

- Implementação de fila com vetor
  - incremento das posições do vetor de forma “circular”:
    - usa o operador módulo “%”
  - parâmetros da fila:
    - n = número de elementos na fila
    - ini = posição do próximo elemento a ser retirado da fila
    - fim = posição onde será inserido o próximo elemento

```
...  
fim = (ini+n)%N  
...
```

# Implementação de fila com vetor

- função `fila_cria`
  - aloca dinamicamente um vetor
  - inicializa a fila como sendo vazia (número de elementos = 0)

```
Fila* fila_cria (void)
```

```
{
```

```
    Fila* f = (Fila*) malloc(sizeof(Fila));
```

```
    f->n = 0;          /* inicializa fila como vazia          */
```

```
    f->ini = 0;        /* escolhe uma posição inicial          */
```

```
    return f;
```

```
}
```

tipo Fila: definido na interface  
struct fila: determina a implementação

# Implementação de fila com vetor

- função `fila_insere`
  - insere um elemento no final da fila
  - usa a próxima posição livre do vetor, se houver

```
void fila_insere (Fila* f, float v)
{ int fim;
  if (f->n == N) { /* fila cheia: capacidade esgotada */
    printf("Capacidade da fila estourou.\n");
    exit(1);      /* aborta programa */
  }
  /* insere elemento na próxima posição livre */
  fim = (f->ini + f->n) % N;
  f->vet[fim] = v;
  f->n++;
}
```

# Implementação de fila com vetor

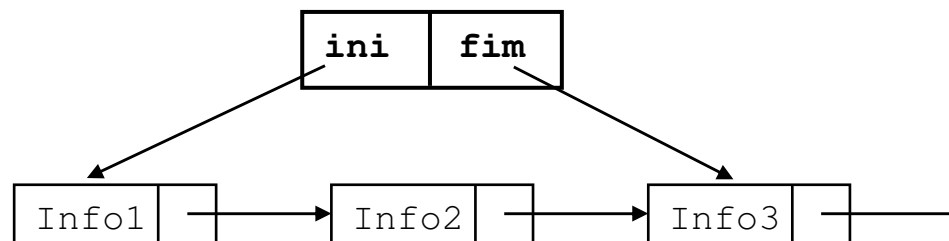
- função `fila_retira`
  - retira o elemento do início da fila, retornando o seu valor
  - verifica se a fila está ou não vazia

```
float fila_retira (Fila* f)
{ float v;
  if (fila_vazia(f)) {
    printf("Fila vazia.\n");
    exit(1);      /* aborta programa */
  }
  /* retira elemento do início */
  v = f->vet[f->ini];
  f->ini = (f->ini + 1) % N;
  f->n--;
  return v;
}
```

9/8/2

# Implementação de fila com lista

- Implementação de fila com lista
  - elementos da fila armazenados na lista
  - usa dois ponteiros
    - ini        aponta para o primeiro elemento da fila
    - fim        aponta para o último elemento da fila



# Implementação de fila com lista

- Implementação de fila com lista
  - elementos da fila armazenados na lista
  - fila representada por um ponteiro para o primeiro nó da lista

```
/* nó da lista para armazenar valores reais */
```

```
struct lista {  
    float info;  
    struct lista* prox;  
};  
typedef struct lista Lista;
```

```
/* estrutura da fila */
```

```
struct fila {  
    Lista* ini;  
    Lista* fim;  
};
```



# Implementação de fila com lista

- função `fila_cria`
  - cria aloca a estrutura da fila
  - inicializa a lista como sendo vazia

```
Fila* fila_cria (void)
{
    Fila* f = (Fila*) malloc(sizeof(Fila));
    f->ini = f->fim = NULL;
    return f;
}
```

# Implementação de fila com lista

- função fila\_inserere
  - insere novo elemento **n** no final da lista

```
void fila_inserere (Fila* f, float v)
{
    Lista* n = (Lista*) malloc(sizeof(Lista));
    n->info = v;           /* armazena informação */
    n->prox = NULL;        /* novo nó passa a ser o último */
    if (f->fim != NULL)    /* verifica se lista não estava vazia */
        f->fim->prox = n;
    else                   /* fila estava vazia */
        f->ini = n;
    f->fim = n;            /* fila aponta para novo elemento */
}
```

# Implementação de fila com lista

- função fila\_retira
  - retira o elemento do início da lista

```
float fila_retira (Fila* f)
{
    Lista* t;
    float v;
    if (fila_vazia(f)) { printf("Fila vazia.\n");
                        exit(1); } /* aborta programa */

    t = f->ini;
    v = t->info;
    f->ini = t->prox;
    if (f->ini == NULL) /* verifica se fila ficou vazia */
        f->fim = NULL;
    free(t);
    return v;
}
```

# Implementação de fila com lista

- função `fila_libera`
  - libera a fila depois de liberar todos os elementos da lista

```
void fila_libera (Fila* f)
{
    Lista* q = f->ini;
    while (q!=NULL) {
        Lista* t = q->prox;
        free(q);
        q = t;
    }
    free(f);
}
```

# Fila dupla

- Fila dupla:
  - fila na qual é possível:
    - inserir novos elementos no início e no fim
    - retirar elementos de ambos os extremos
  - simula, dentro de uma mesma estrutura, duas filas, com os elementos em ordem inversa uma da outra

# Interface do tipo fila dupla

- Interface do tipo abstrato Fila2: [fila2.h](#)
  - função [fila2\\_cria](#)
    - aloca dinamicamente a estrutura da fila
    - inicializa seus campos e retorna seu ponteiro
  - função [fila2\\_insere\\_fim](#) e função [fila2\\_retira\\_ini](#)
    - insere no fim e retira do início, respectivamente, um valor real na fila
  - função [fila2\\_insere\\_ini](#) e função [fila2\\_retira\\_fim](#)
    - insere no início e retira do fim, respectivamente, um valor real na fila
  - função [fila2\\_vazia](#)
    - informa se a fila está ou não vazia
  - função [fila2\\_libera](#)
    - destrói a fila, liberando toda a memória usada pela estrutura

```
typedef struct fila2 Fila2;  
  
Fila2* fila2_cria (void);  
  
void fila2_inserir_ini (Fila2* f, float v);  
  
void fila2_inserir_fim (Fila2* f, float v);  
  
float fila2_retira_ini (Fila2* f);  
  
float fila2_retira_fim (Fila2* f);  
  
int fila2_vazia (Fila2* f);  
  
void fila2_libera (Fila2* f);
```

# Implementação de fila dupla com vetor

- função fila2\_inserere\_ini
  - insere elemento no início da fila
    - índice do elemento que precede ini é dado por  $(ini - 1 + N) \% N$

```
void fila2_inserere_ini (Fila* f, float v)
{ int prec;
  if (f->n == N) { /* fila cheia: capacidade esgotada */
    printf("Capacidade da fila estourou.\n");
    exit(1);      /* aborta programa */
  }
  /* insere elemento na posição precedente ao início */
  prec = (f->ini - 1 + N) % N; /* decremento circular */
  f->vet[prec] = v;
  f->ini = prec; /* atualiza índice para início */
  f->n++;
}
```



# Implementação de fila dupla com vetor

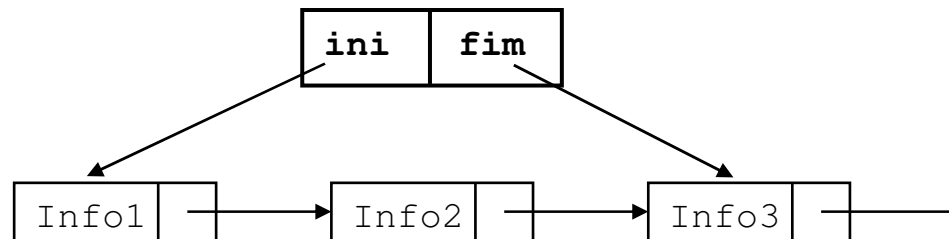
- função `fila2_retira_final`
  - retira elemento do final da fila
    - índice do último elemento é dado por  $(ini+n-1)\%N$

```
float fila_retira (Fila* f)
{ float v;
  if (fila_vazia(f)) {
    printf("Fila vazia.\n");
    exit(1);      /* aborta programa */
  }
  /* retira elemento do início */
  v = f->vet[f->ini];
  f->ini = (f->ini + 1) % N;
  f->n--;
  return v;
}
```

9/8/2

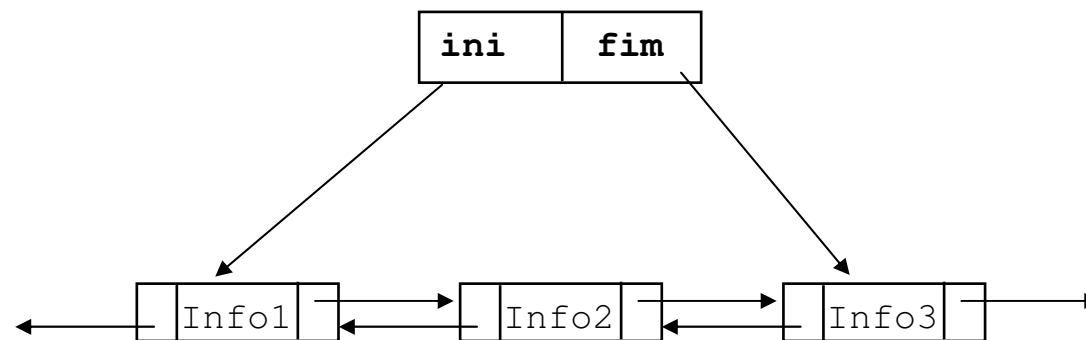
# Implementação de fila dupla com lista duplamente encadeada

- Implementação de fila dupla com lista simplesmente encadeada
  - função para retirar do fim
    - não pode ser implementada de forma eficiente
    - dado o ponteiro para o último elemento da lista, não é possível acessar de forma eficiente o anterior, que passaria a ser o último elemento



# Implementação de fila dupla com lista duplamente encadeada

- Implementação de fila dupla com lista duplamente encadeada
  - dado o ponteiro de um nó, é possível acessar ambos os elementos adjacentes
  - resolve o problema de acessar o elemento anterior ao último



# Implementação de fila dupla com lista duplamente encadeada

- Implementação de fila dupla com lista duplamente encadeada

```
/* nó da lista para armazenar valores reais */  
struct lista2 {  
    float info;  
    struct lista2* ant;  
    struct lista2* prox;  
};  
typedef struct lista2 Lista2;  
  
/* estrutura da fila */  
struct fila2 {  
    Lista2* ini;  
    Lista2* fim;  
};
```

# Implementação de fila dupla com lista duplamente encadeada

- função auxiliar: insere no início
  - insere novo elemento **n** no inicio da lista duplamente encadeada

```
/* função auxiliar: insere no início */
static Lista2* ins2_ini (Lista2* ini, float v)
{
    Lista* p = (Lista2*) malloc(sizeof(Lista2));
    p->info = v;
    p->prox = ini;
    p->ant = NULL;
    if (ini != NULL)                /* verifica se lista não estava vazia */
        ini->ant = p;
    return p;
}
```

# Implementação de fila dupla com lista duplamente encadeada

- função auxiliar: insere no fim
  - insere novo elemento **n** no fim da lista duplamente encadeada

```
/* função auxiliar: insere no fim */
static Lista2* ins2_fim (Lista2* fim, float v)
{
    Lista2* p = (Lista2*) malloc(sizeof(Lista2));
    p->info = v;
    p->prox = NULL;
    p->ant = fim;
    if (fim != NULL)           /* verifica se lista não estava vazia */
        fim->prox = p;
    return p;
}
```

# Implementação de fila dupla com lista duplamente encadeada

- função auxiliar: retira do início
  - retira elemento do início da lista duplamente encadeada

```
/* função auxiliar: retira do início */  
static Lista2* ret2_ini (Lista2* ini)  
{  
    Lista2* p = ini->prox;  
    if (p != NULL) /* verifica se lista não ficou vazia */  
        p->ant = NULL;  
    free(ini);  
    return p;  
}
```

# Implementação de fila dupla com lista duplamente encadeada

- função auxiliar: retira do fim
  - retira elemento do fim da lista duplamente encadeada

```
/* função auxiliar: retira do fim */
static Lista2* ret2_fim (Lista2* fim)
{
    Lista2* p = fim->ant;
    if (p != NULL)                /* verifica se lista não ficou vazia */
        p->prox = NULL;
    free(fim);
    return p;
}
```



# Implementação de fila dupla com lista duplamente encadeada

- funções fila2\_inserir\_ini e fila2\_inserir\_fim

```
void fila2_inserir_ini (Fila2* f, float v) {  
    f->ini = ins2_ini(f->ini,v);  
    if (f->fim==NULL)          /* fila antes vazia? */  
        f->fim = f->ini;  
}
```

```
void fila2_inserir_fim (Fila2* f, float v) {  
    f->fim = ins2_fim(f->fim,v);  
    if (f->ini==NULL)          /* fila antes vazia? */  
        f->ini = f->fim;  
}
```

# Implementação de fila dupla com lista duplamente encadeada

- função fila2\_retira\_ini

```
float fila2_retira_ini (Fila2* f) {  
    float v;  
    if (fila2_vazia(f)) {  
        printf("Fila vazia.\n");  
        exit(1);           /* aborta programa */  
    }  
    v = f->ini->info;  
    f->ini = ret2_ini(f->ini);  
    if (f->ini == NULL)     /* fila ficou vazia? */  
        f->fim = NULL;  
    return v;  
}
```

# Implementação de fila dupla com lista duplamente encadeada

- função fila2\_retira\_fim

```
float fila2_retira_fim (Fila2* f) {  
    float v;  
    if (vazia(f)) {  
        printf("Fila vazia.\n");  
        exit(1);           /* aborta programa */  
    }  
    v = f->fim->info;  
    f->fim = ret2_fim(f->fim);  
    if (f->fim == NULL)    /* fila ficou vazia? */  
        f->ini = NULL;  
    return v;  
}
```

# Implementação de fila dupla com lista duplamente encadeada

- função fila\_retira
  - retira o elemento do início da lista

```
float fila_retira (Fila* f)
{
    Lista* t;
    float v;
    if (fila_vazia(f)) { printf("Fila vazia.\n");
                        exit(1); } /* aborta programa */

    t = f->ini;
    v = t->info;
    f->ini = t->prox;
    if (f->ini == NULL) /* verifica se fila ficou vazia */
        f->fim = NULL;
    free(t);
    return v;
}
```

# Resumo

## fila

*insere*      insere novo elemento no final da fila

*remove*      remove o elemento do inicio da fila

