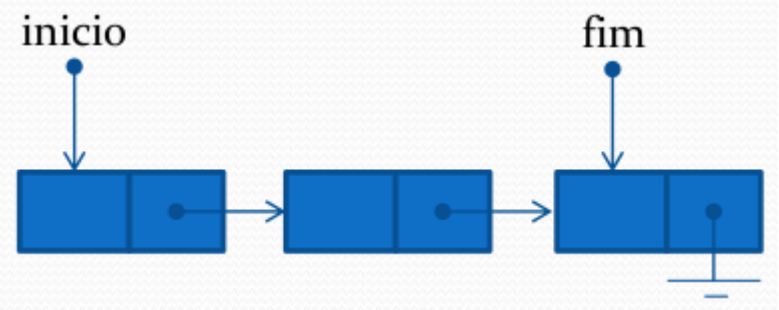
Estruturas de Dados Clássicas – Filas – Parte 2

Prof. Bárbara Quintela

barbaraquintela@pucminas.cesjf.br

Fila Dinâmica

- Fila implementada através de uma lista linear encadeada.
- É uma lista linear encadeada em que a operação de inserção de um elemento (enfileirar) é realizada numa extremidade da lista enquanto a de retirada de um elemento (desenfileirar) é realizada na outra extremidade da lista.
- Para facilitar as operações de enfileirar e desenfileirar utiliza-se doi ponteiros um para o primeiro elemento (inicio da fila) e outra para o último elemento (fim da fila)



Operações com Fila Dinâmica

- **Criar** a fila - Declarar os ponteiros para o inicio e fim da fila
- **Inicializar** a fila - Determinar o status inicial da fila, a fim de prepará-la para a inserção de dados. Ponteiros para o inicio e para o fim da fila são NULL – apontam para nada
- **Enfileirar** - consiste em inserir um valor no fim da fila.
- **Verificar** se a fila está **vazia** - Caracterizada pelo ponteiro para inicio estar com NULL.
- **Desenfileirar** - Consiste em retirar um valor do inicio da fila. É preciso verificar previamente se a fila está vazia.
- **Percorrer** - Mostrar a fila

Observações sobre Fila Dinâmica

- Na fila **dinâmica** não há necessidade de verificar se a fila está cheia, pois o limite na inserção de dados, está na capacidade do computador
- Uma das **vantagens** da fila dinâmica sobre a estática (sequencial - implementada com vetor) é não ter limite de sua capacidade. Outra vantagem: como os nós são alocados a medida que são necessários, a fila dinâmica só usa o espaço realmente necessário.
- As operações em um fila estática tem código mais simples.
- As aplicações para a fila dinâmica são as mesmas da fila estática.

Criar a Fila

- Supondo a descrição de um nó:

```
struct no {
    int dado;
    struct no *prox;
};
typedef struct no No;
```

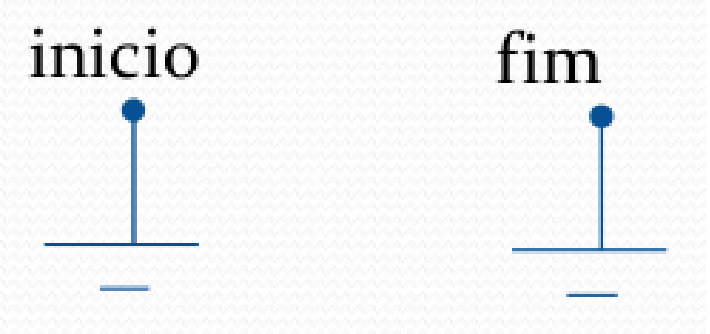
- Criar a fila é declarar os ponteiros para o inicio e para o fim da fila:

```
No *inicio, *fim;
```

Inicializar a Fila

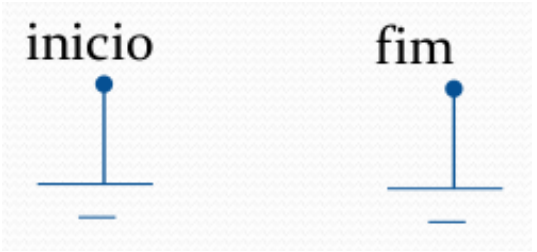
- Determina o status inicial da fila, a fim de prepará-la para a inserção de dados.
- Inicialmente o ponteiro para o inicio não aponta para nada, assim como o ponteiro para o fim:

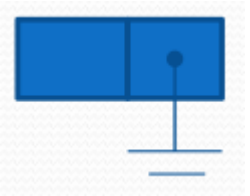
  inicio = fim = NULL;



Inicializar a Fila

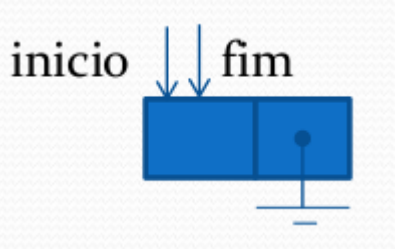
- Consiste em inserir um valor no fim da fila.
  - Cria-se um novo nó:





  - Novo nó será apontado pelo último nó da fila

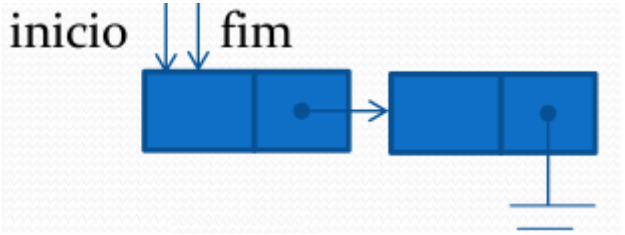
  - Se novo nó é o primeiro a ser inserido, inicio também irá apontar para novo nó



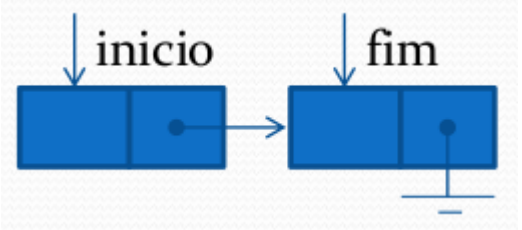
Enfileirar (Enqueue)

- Enfileirando mais um nó
  - Cria-se um novo nó:

  - Novo nó será apontado pelo último nó da fila



  - Ajusta o fim



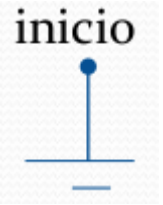
Código para Enfileirar

```
void enfileirar(int valor) {
    // cria um novo no
    No* novo = (No*)malloc(sizeof(No));
    novo->dado = valor;
    novo->prox = NULL;
    if (inicio == NULL) { // se lista está vazia
        inicio = novo;       // novo no será o primeiro elemento da lista
    }
    else { // se lista não estiver vazia
            // ultimo nó aponta para novo nó
        fim->prox = novo;
    }
    fim = novo; // fim indica um novo nó
}
```

Verificar se a fila está vazia

- A fila estará vazia, quando o ponteiro para o início estiver apontando para nada

  (estiver aterrado)



Código par verificar se a fila está vazia

```
if (inicio== NULL) {
    printf(“\nFila vazia!\n”);
}
```

- Ou usando uma função:

```
bool estaVazia() {
    return (inicio == NULL);
}
```

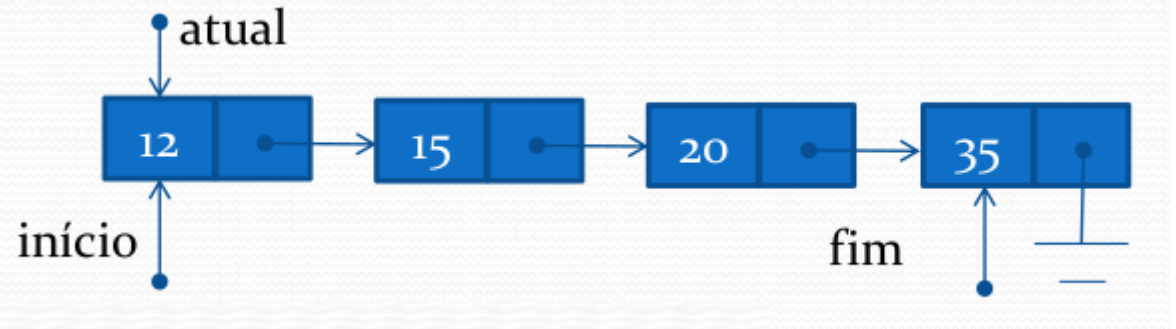
Desenfileirar - Remover da Fila

```
int desenfileirar(){
    int v = 0;
    if (inicio==NULL){//verifica se esta vazia
        printf("Fila Vazia! \n");
    }else{
        v = inicio->dado;
        inicio = inicio->prox;
    }
    return v;
}
```

Percorrer – Mostrar a Fila

Deve-se verificar se existem elementos na fila (ou seja, se a fila não está vazia)

- Utiliza-se uma ponteiro auxiliar (aqui chamado de ***atual***) que irá percorrer a fila
- O ponteiro atual é inicializado com o valor do ponteiro ***inicio***



- ***atual*** percorre a fila até o ***fim***

Código para percorrer a fila

```
void percorrer () {
    No *atual; // ponteiro para percorrer a fila
    atual = inicio;
    printf("\nFila=> ");
    while (atual != NULL) {
        printf(“%d \t”, atual->dado);
        atual = atual->prox;
    }
    printf("\n");
}
```

Exercício

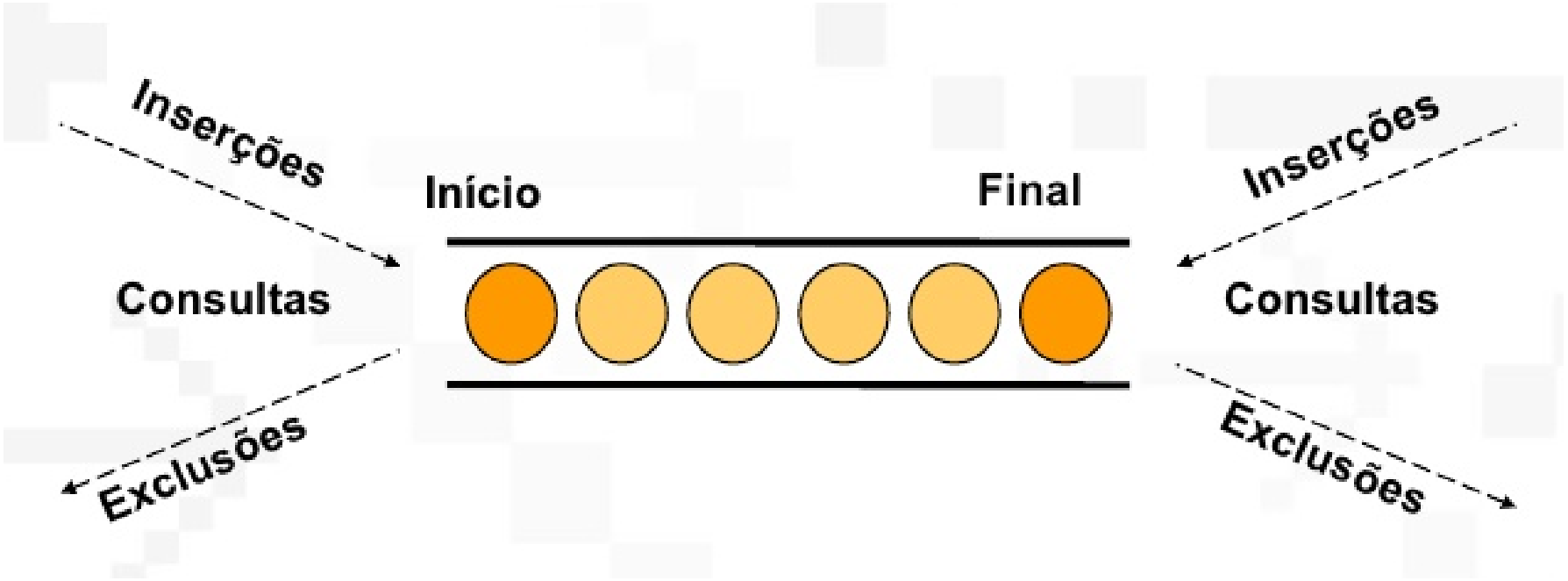
- Dada uma fila de inteiros, escreva um programa que exclua todos os números negativos sem alterar a posição dos outros elementos da fila.

Fila Dupla - “Deque”

Fila na qual é possível inserir dados nas duas extremidades, Início e fim.

- Consequentemente, pode retirar das duas também.

- Do inglês: *Double-Ended QUEue*



Fila Dupla

- Funcionalidades:
  - Criar estrutura de fila dupla
  - Inserir elemento no início
  - Inserir elemento no fim
  - Retirar elemento do inicio
  - Retirar elemento do fim
  - Verificar se está vazia
  - Consultar / modificar elemento no inicio ou fim
  - Liberar a fila

Fila Dupla - Estrutura

- Implementação por encadeamento
- A estrutura do nó armazena ponteiros para o nó anterior e para o próximo e a estrutura da fila armazena ponteiros para o início e o fim da fila:

```
struct lista2{
    float info;
    struct lista2* ant;
    struct lista2* prox;
};
typedef struct lista2 Lista2;
```

```
struct fila2{
    Lista2* ini;
    Lista2* fim;
}
typedef struct fila2 Fila2;
```

Fila Dupla – Auxiliar para Inserção no início

```
/* funcao auxiliar: para inserir elemento no inicio */
static Lista2* ins2_ini(Lista2* ini, float v){
    Lista2* novo = (Lista2*) malloc(sizeof(Lista2));
    novo->info = v;
    novo->prox = ini;
    novo->ant  = NULL;
    if(ini != NULL) /*verifica se a lista esta vazia*/
        ini->ant=novo;
    return novo;
}
```

Fila Dupla – Auxiliar para Inserção no fim

```
/* funcao auxiliar: para inserir elemento no fim */
static Lista2* ins2_fim(Lista2* fim, float v){
    Lista2* novo = (Lista2*) malloc(sizeof(Lista2));
    novo->info = v;
    novo->prox = NULL;
    novo->ant  = fim;
    if(fim!=NULL)/*verifica se a lista esta vazia*/
        fim->prox = novo;
    return novo;
}
```

Fila Dupla – Auxiliar para Remove do início

```
/* funcao auxiliar: para retirar elemento do inicio */
static Lista2* ret2_ini(Lista2* ini){
    Lista2* p = ini->prox;
    if(p!=NULL)/*verifica se a lista nao ficou vazia*/
        p->ant = NULL;
    free(ini);
    return p;
}
```

Fila Dupla – Auxiliar para Remove do fim

```
/* funcao auxiliar: para retirar elemento do fim */
static Lista2* ret2_fim(Lista2* fim){
    Lista2* p = fim->ant;
    if(p!=NULL)/*verifica se a lista nao ficou vazia*/
        p->prox = NULL;
    free(fim);
    return p;
}
```

Fila Dupla – Inserir no Inicio

```
/* funcao para inserir no inicio */
void fila2_insere_ini(Fila2* f, float v){
    f->ini = ins2_ini(f->ini, v);
    if(f->fim == NULL) /* se fila estava vazia */
        f->fim = f->ini;
}
```

Fila Dupla – Inserir no Fim

```
/* funcao para inserir no fim */
void fila2_insere_fim(Fila2* f, float v){
    f->fim = ins2_fim(f->fim, v);
    if(f->ini == NULL) /* se fila estava vazia */
        f->ini = f->fim;
}
```

Fila Dupla – Retirar do Inicio

```
/* funcao para retirar do inicio*/
float fila2_retira_ini(Fila2* f2){
    float v;
    if (fila2_vazia(f2)){
        printf("Fila vazia!\n");
        exit(1); /* aborta programa */
    }
    v = f2->ini->info;
    f2->ini = ret2_ini(f2->ini);
    if(f2->ini == NULL) /* se fila ficou vazia */
        f2->fim = NULL;
    return v;
}
```

Fila Dupla – Retirar do Fim

```
float fila2_retira_fim(Fila2* f2){
    float v;
    if(fila2_vazia(f2)){
        printf("Fila vazia!\n");
        exit(1); /* aborta programa */
    }
    v = f2->fim->info;
    f2->fim = ret2_fim(f2->fim);
    if(f2->fim == NULL) /* se ficou vazia */
        f2->ini = NULL;
    return v;
}
```

Fila Dupla – Criar e liberar

```
Fila2* fila2_cria(void){
    Fila2* f2 = (Fila2*) malloc(sizeof(Fila2));
    f2->ini = f2->fim = NULL;
    return f2;
}
void fila2_libera(Fila2* f2){
    Lista2* q = f2->ini;
    while (q!= NULL){
        Lista2* t = q->prox;
        free(q);
        q = t;
    }
    free(f2);
}
```

Fila Dupla – Imprimir

```
void fila2_imprime(Fila2* f2){
    Lista2* q;
    for (q=f2->ini; q!=NULL; q=q->prox)
        printf("%.2f\t",q->info);
    printf("\n");
}
```

Fila Dupla – Testar se é vazia

```
int fila2_vazia(Fila2* f2){
    return (f2->ini==NULL);
}
```

Fila Dupla – Sugestão

```
int main(){
    Fila2* f2 = fila2_cria();
    fila2_insere_ini(f2,20.8);
    fila2_insere_ini(f2,20.0);
    fila2_imprime(f2);
    fila2_insere_fim(f2,21.2);
    fila2_insere_fim(f2,24.3);
    fila2_imprime(f2);
    fila2_retira_fim(f2);
    fila2_imprime(f2);
    fila2_retira_ini(f2);
    fila2_imprime(f2);
    fila2_libera(f2);
}
```