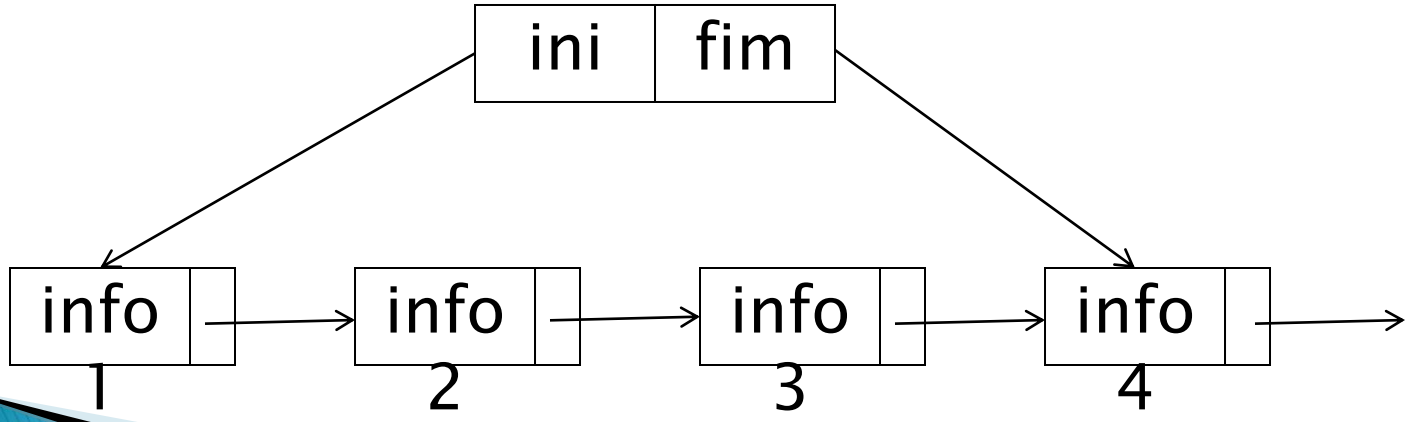
Estrutura de Dados

Prof. Adriano Teixeira de Souza

Implementação de fila com lista

- Utilizaremos também uma lista simplesmente encadeada para tal.
- Como teremos que inserir e retirar elementos nas extremidades opostas da lista, que representarão o início e o fim da fila, teremos que usar dois ponteiros, **ini** e fim, que aprontam respectivamente para o primeiro e para o último elemento da fila.



Implementação da fila com lista

- O nó da lista para armazenar valores reais pode ser dado por:

```
typedef struct {
 float info;
 struct No* proximo;
} No;
```

- A estrutura da fila agrupa os ponteiros para o início e o fim da lista:

```
typedef struct
 {
 No* inicio;
 No* fim;
} Fila ;
```

Filas :: Criação de uma fila

- A função cria aloca a estrutura da fila e inicializa a lista como sendo vazia.

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

Filas :: Funções Auxiliares

- Cada novo elemento é inserido no fim da lista e sempre retiramos o elemento do início da lista. Portanto, precisamos das funções:

Filas :: Funções Auxiliares

```
/* função auxiliar: insere no fim */
No* ins_fim (No* fim, float v) {

    No* p = (No*) malloc(sizeof(No));
    p->info = v;
    p->proximo = NULL;
    if (fim != NULL) // verifica se a lista não estava vazia
        fim->proximo = p;
    return p;
}
/* função auxiliar: retira do início */
No* ret_ini (No* inicio) {
    No* p = inicio->proximo;
    free(inicio);
    return p;
}
```

Filas :: Inserção de elemento

- As funções que manipulam a fila fazem uso dessas funções de lista.
- Note que a função de inserção deve atualizar ambos os ponteiros, **ini** e **fim**, quanto da inserção do primeiro elemento:

```
void insere (Fila* f, float v)
{
   f->fim = ins_fim(f->fim,v);
   if (f->inicio == NULL) // fila antes vazia?
      f->inicio = f->fim;
}
```

Filas :: Remoção de elemento

- A função de retirar deve atualizar ambos os ponteiros (**ini** e **fim)** se a fila tornar-se vazia após a remoção do elemento.

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

Filas :: Funções Vazia e Libera

- A fila estará vazia se a lista estiver vazia:

```
int vazia (Fila* f)
{
    return (f->inicio == NULL);
}
```

- A função que libera a fila deve antes liberar todos os elementos da lista.

```
void libera (Fila* f) {
    No* q = f->inicio;
    while (q!=NULL) {
        No* t = q->proximo;
        free(q);
        q = t;
    }
    free(f);
}
```

Filas :: Função Imprime

- Para testar o código, pode ser útil implementarmos um função que imprima os valores armazenados na fila. A ordem de impressão adotada é do início para o fim.

```
/* versão com lista*/
void imprime (Fila* f){
   No* q;
   for (q=f->inicio; q!=NULL; q=q->proximo){
        printf("%f\n", q->info) ;
   }
}
```

Filas :: Exemplo de utilização

```
main(){

    Fila* f = cria();
    insere (f,20.0);
    insere (f,20.8);
    insere (f,20.2);
    insere (f,20.3);
    imprime (f);
    printf ("Primeiro elemento: %f\n", retira(f));
    printf ("Segundo elemento: %f\n", retira(f));
    printf ("Configuracao da fila:\n");
    imprime (f);
    libera (f);

    system("pause");

}
```