Fila

1. Introdução

Uma fila é um tipo específico de lista encadeada, onde só é permitido inserir elementos no final da fila, e retirar do início da fila. Assim, o primeiro elemento inserido é o primeiro a ser retirado. Esse comportamento frequentemente é conhecido pela sigla FIFO (First In – First Out).

O comportamento de uma estrutura de dados do tipo Fila simula o comportamento de uma fila de atendimento de pessoas, por exemplo a fila de um banco, em que as pessoas entram no final da fila, e o primeiro que chegou é o primeiro que será atendido. A ordem de atendimento (retirada da fila) é a mesma ordem de chegada (inserção na fila).

Da mesma forma que qualquer lista, a fila pode conter qualquer tipo de dado, mas para nossos exemplos, vamos usar o tipo mais simples possível, que é um dado inteiro. A implementação de fila pode ser feita com alocação estática (vetores) ou alocação dinâmica, utilizando ponteiros. Para nossos exemplos, usaremos alocação dinâmica. A implementação de fila com vetores é deixada como exercício opcional.

Diferentemente das listas genéricas estudadas e da pilha, uma implementação mais eficiente da fila requer um ponteiro para o início da fila (primeiro elemento) e um outro ponteiro para o final da fila (último elemento), de modo a não precisar de laço de repetição para inserir ou retirar elementos. Deste modo, a declaração de tipo da fila dinâmica pode ser:

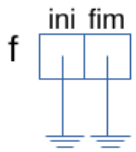
```
struct elemento
{
     int dado;                // Conteúdo (inteiro)
     struct elemento *prox;   // Ponteiro para o próximo registro
};
typedef struct elemento *ApElemento;

typedef struct
{
     ApElemento ini;
     ApElemento fim;
} Fila;
```

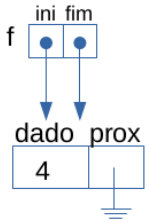
Observe que qualquer variável declarada do tipo `Fila` será uma struct (um registro) contendo dois ponteiros para uma estrutura do tipo elemento. O ponteiro `ini` contém o endereço do primeiro elemento da fila e o ponteiro `fim` contém o endereço do último elemento da fila.

Para exemplificarmos o comportamento da fila, suponha uma fila vazia `p`, na qual vamos inserir os elementos 4, 9, 5 e 3. A sequência de resultados da inserção desses elementos será:

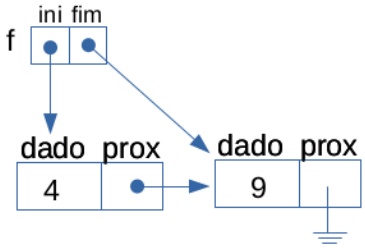
1. fila `f` vazia:



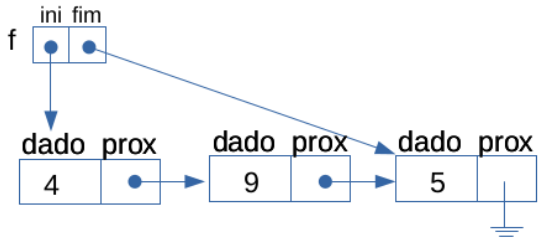
2. Após a inserção do elemento 4:



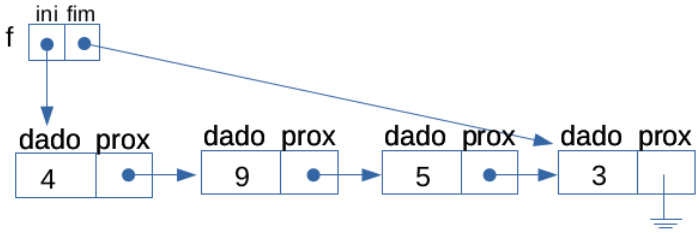
3. Após a inserção dos elementos 4 e 9:



4. Após a inserção dos elementos 4, 9 e 5:



5. Após a inserção dos elementos 4, 9, 5 e 3:



2. Operações com a fila.

As operações definidas sobre uma fila são:

- Criar uma fila vazia;
- Inserir um elemento ao final da fila;
- Retirar um elemento do início da fila;
- Verificar se a fila está vazia.

Embora não seja uma operação definida formalmente sobre uma fila, vamos implementar também uma função para imprimir o conteúdo da fila, apenas para fins didáticos, e de depuração do código.

2.1. Criação de uma fila.

A criação de uma fila precisa produzir uma lista vazia, o que neste caso significa uma struct do tipo `Fila`, contendo o campo `ini` e o campo `fim` apontando para `NULL`. Isso pode ser implementado da seguinte forma.

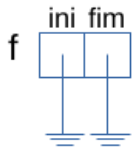
```
Fila criaFila()
{
     Fila f;
     f.ini = NULL;
     f.fim = NULL;
     return (f);
}
```

Para chamar essa função, o retorno da função `criaFila()` deve ser armazenado em uma variável do tipo `Fila`, a exemplo das linhas a seguir.

```
Fila f;              // Declaração da variável f

f = criaFila();      // Inicialização da f (fila vazia)
```

Observe que após a execução dessas linhas a fila `f` contém uma fila vazia, ou seja, `f` contém uma estrutura com os ponteiros `ini` e `fim` com o valor `NULL`. A representação gráfica é:



Observe também que `f` é uma variável com dois campos. Então para acessar o campo `ini` e `fim` da fila `f`, usa-se `f.ini` e `f.fim`.

2.2. Testar se a fila está vazia.

Uma função bastante simples, mas útil para a compreensão do código é a função filaVazia(), que retorna TRUE (1) se a fila está vazia, e FALSE (0) em caso contrário.

```
#define TRUE 1
#define FALSE 0

int filaVazia(Fila f)
{
     if (f.ini == NULL)
          return (TRUE);
     else
          return (FALSE);
}
```

2.3. Exibição do conteúdo de uma fila.

Para imprimir o conteúdo de uma fila é necessário percorrer toda a fila, sequencialmente, do primeiro elemento até o final da fila. Isso pode ser implementado pelo código abaixo.

```
// Imprime todos os nodos da fila p
void imprimeFila(Fila f)
{
     ApElemento ap;

     printf("\nItens da lista\n");
     ap = f.ini;
     while (ap != NULL)
     {
          printf("%d - ",ap->dado);
          ap = ap->prox;
     }
     printf("\n");
}
```

2.4. Inserir um elemento ao final da fila.

Inserir um elemento ao final de uma fila implica em criar um novo nodo, colocar o conteúdo desejado nesse novo nodo e depois conectar esse novo nodo ao final da fila (depois do último elemento da fila). Procure acompanhar o código abaixo, considerando os desenhos acima.

```
Fila insereFila(Fila f, int e)
{
     ApElemento novo;

     // Aloca o espaço e faz as atribuições de valores
     novo = malloc(sizeof(struct elemento));
     novo -> dado = e;
     novo -> prox = NULL;
     if (filaVazia(f))
     {
          f.ini = novo;
          f.fim = novo;
     }
     else
```

```
     {
          f.fim->prox = novo;
          f.fim = novo;
     }
     return (f);
}
```

Observe que a função funciona tanto para o caso onde a fila está vazia (o novo elemento será o primeiro e último da fila), quanto para o caso onde `f` não está vazia (o novo elemento será inserido depois do último, e passará a ser o final da fila).

Essa função sempre inclui um novo elemento no final da fila, e portanto, sempre altera `f.fim`. Pode também alterar `f.ini`, se a fila estava vazia. Uma estrutura do tipo `Fila` é retornada. Nesse caso, considere que no programa principal tenhamos uma lista `f1` e queremos inserir um dado que está na variável inteira `valor`. A chamada da função pode ser:

```
f1 = insereFila(f1, valor);
```

2.5. Retirar um elemento do início da fila.

O único elemento que se pode retirar de uma fila é o elemento que está no início da fila. Dessa forma não é necessário percorrer a lista até localizar o elemento. Nesse caso, como o início da fila será alterado, e também será retornado o elemento retirado, optamos nesse exemplo por passar o parâmetro `e` por referência (ponteiro), de modo que ele possa ser alterado dentro da função.

Antes de retirar um elemento, é necessário apenas verificar se a fila não está vazia.

```
Fila retiraFila (Fila f, int *e)
{
     ApElemento ap;

     if (!filaVazia(f))
     {
          // Retorna o valor do elemento que está no inicio da fila
          *e = f.ini->dado;
          // Salva o endereço do inicio para liberação
          ap = f.ini;
          // Faz o inicio apontar para o segundo elemento
          f.ini = f.ini->prox;
          // Libera o espaço ocupado pelo elemento removido
          free(ap);
          // Se retirou o único elemento, atualiza o fim
          if (filaVazia(f))
             f.fim = NULL;
     }
     else
     {
          *e = -1;
          printf("\nFila Vazia");
     }
     return (f);
}
```

Resumindo textualmente, se a fila não estiver vazia, será retornado no parâmetro `*e` o valor o valor do primeiro elemento da fila. Libera o primeiro elemento, atualizando o início da fila para o próximo elemento. Caso a fila esteja vazia, o valor retornado no parâmetro `*e` será -1, e será impressa a mensagem “Fila Vazia”. Ao final, retorna a estrutura `f` com o endereço do início e fim da fila atualizados.

Outra forma de implementar o `retiraFila` é passando o parâmetro da fila `f` por referência (podendo alterar dentro da função), e fazendo a função retornar o valor do elemento retirado da fila. Segue um exemplo abaixo. Observe que a escolha de qual forma implementar a função retiraFila deve estar coerente com a chamada no programa principal (número e tipos dos parâmetros).

```
int retiraFila (Fila *f)
{
     ApElemento ap;
     int ret;

     if (!filaVazia(*f))
     {
          // Copia o valor do elemento que está no inicio da fila
          ret = f->ini->dado;
          // Salva o endereço do inicio para liberação
          ap = f->ini;
          // Faz o inicio apontar para o segundo elemento
          f->ini = f->ini->prox;
          // Libera o espaço ocupado pelo elemento removido
          free(ap);
          // Se retirou o único elemento, atualiza f->fim
          if (filaVazia(*f))
             f->fim = NULL;
     }
     else
     {
          ret = -1;
          printf("\nFila Vazia");
     }
     return (ret);
}
```