Algoritmos e Programação Estruturada

Filas

Você sabia que seu material didático é interativo e multimídia? Isso significa que você pode interagir com o conteúdo de diversas formas, a qualquer hora e lugar. Na versão impressa, porém, alguns conteúdos interativos ficam desabilitados. Por essa razão, fique atento: sempre que possível, opte pela versão digital. Bons estudos!

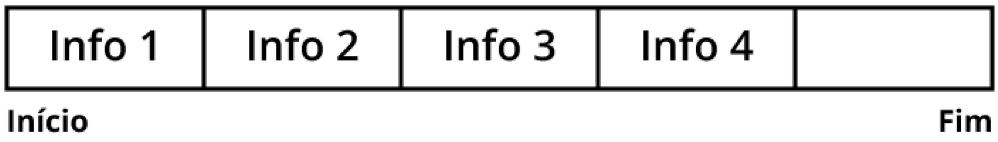
Nesta webaula, vamos conhecer sobre o funcionamento de filas.

Fila

Uma fila é a representação de um conjunto de elementos. Podemos remover esses elementos deste conjunto por uma extremidade chamada de início da fila, e pela outra extremidade, na qual são inseridos os elementos, chamada de final da fila (TENENBAUM, LANGSAM e AUGENSTEIN, 2007).

Assim como uma pilha, as filas também são estruturas dinâmicas com tamanho variável, podendo aumentar ou diminuir conforme são inseridos ou removidos elementos da fila.

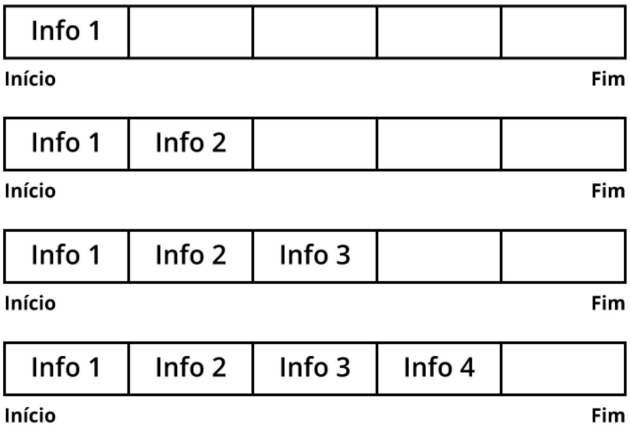
Exemplo de uma estrutura de dados do tipo fila



Fonte: elaborada pelo autor.

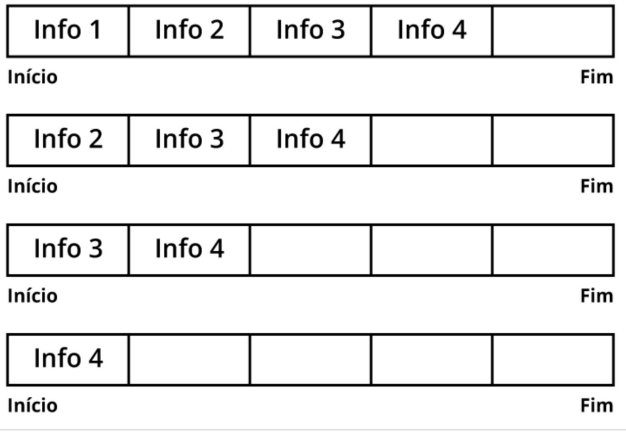
Em uma fila os elementos entram por uma extremidade e são removidos pela outra extremidade. Isso é conhecido como FIFO (first in, first out, ou seja, o primeiro que entra é o primeiro a sair). No caso desta fila, sabemos quais os elementos com base em seu número de índice. Assim, a fila apresenta sua ordem de entrada (fim da fila) e sua ordem de saída dos elementos (início da fila).

Entrada de elemento pelo final da fila



Fonte: elaborada pelo autor.

Saída de elemento pelo início da fila



Fonte: elaborada pelo autor.

Operações de filas

Segundo Drozdek (2016), a estrutura de dados de fila possui operações similares as da estrutura de pilha para gerenciamento de uma fila, como:

- Criar uma fila vazia.
- Inserir um elemento no fim da fila.
- Remover o elemento do início da fila.
- Verificar se a fila está vazia.
- Liberar a fila.

Conforme Celes, Cerqueira e Rangel (2004), podemos simplesmente utilizar um vetor para armazenar os elementos e implementarmos uma fila nessa estrutura de dados, ou podemos utilizar uma alocação dinâmica de memória para armazenar esses elementos.

A seguir, veja as implementações de fila:

Declaração da estrutura inicial

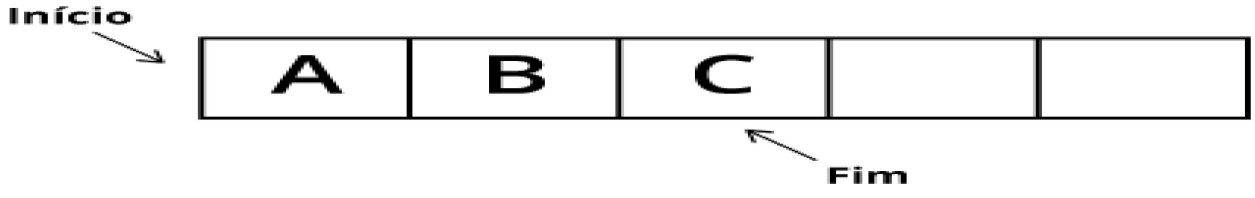
```
#define N 100
struct fila {
        int n;
        int ini;
        char vet[N];
};
typedef struct fila Fila;
```

Inicializar a fila

```
Fila* inicia_fila (void){
        Fila* f = (Fila*) malloc(sizeof(Fila));
        f -> n = 0;
        f -> ini = 0;
        return f;
}
```

Inserir elemento na fila

```
void insere_fila (Fila* f, char elem){
        int fim;
        if (f -> n == N){
                printf("A fila está cheia.\n");
                exit(1);
        }
        fim = (f -> ini + f -> n) % N;
        f -> vet[fim] = elem;
        f -> n++;
}
```



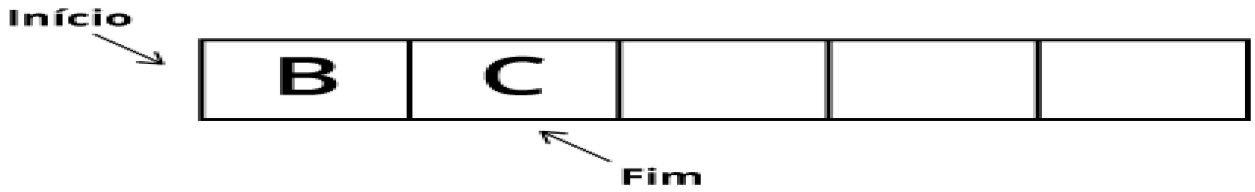
Fonte: elaborada pelo autor

Remover elemento na fila

```
float remove_fila (Fila* f){
```

```
        char elem;

        if (fila_vazia(f)){
                printf("A Fila esta vazia\n");
                exit(1);
        }
        elem = f -> vet[f -> ini];
        f -> ini = (f -> ini + 1) % N;
        f -> n--;
        return elem;
}
```



Fonte: elaborada pelo autor

Informar se a fila está vazia

```
int fila_vazia (Fila* f){
        return (f -> n == 0);
}
```

Liberar alocação de memória

```
void libera_fila(Fila* f){
        free(f);
}
```

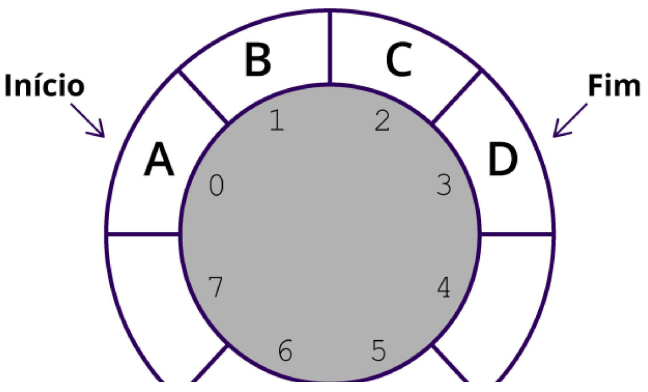
Filas Circulares

Segundo Silva (2007), as filas não apresentam uma solução completa, sendo que, mesmo chegando ao final do vetor poderemos ter a fila cheia mesmo não estando cheia, uma vez que elementos podem ter sido removidos e para isso, podemos utilizar as Filas Circulares como solução para esta situação. A Fila Circular possui a seguinte definição, em sua implementação:

Um vetor para os elementos	Um valor inteiro para o tamanho da fila	Um valor inteiro para o início da fila	Um valor inteiro para o fim da fila

Conforme Drozdek (2016), em uma Fila Circular, o conceito de circularidade se baseia quando o último elemento da fila está na última posição do vetor, e é adjacente à primeira. Assim, são os ponteiros, e não os elementos da fila que se movem em direção ao início do vetor.

Estrutura da fila circular



Fonte: elaborada pelo autor.

Para implementar uma estrutura de fila circular, podemos utilizar como exemplo o código a seguir:

Definição da constante

```
/* Vamos definir a constante N com valor de 10 */
#define N 10
struct filacirc {  /* Criação da estrutura da Fila Circular */
        int tam, ini, fim;
        char vet[N];
};
typedef struct filacirc FilaCirc;
```

Função para inicializar a fila

```
/* Função para inicializar a Fila */
void inicia_fila (FilaCirc *f){
        f -> tam = 0;
        f -> ini = 1;
        f -> fim = 0;
}
```

Função para inserir a fila

```
/* Função para inserir na Fila */
void insere_fila (FilaCirc* f, char elem){
        if (f -> tam == N - 1){   /* Verifica se a Fila está completa */
                printf("A fila esta cheia\n");
        } else { /* Caso a Fila não esteja completa, inserimos o elemento */
                f -> fim = (f -> fim % (N - 1)) + 1;
                f -> vet[f -> fim] = elem;
                f -> tam++;
        }
}
```

Verificar se a fila está vazia

```
int fila_vazia (FilaCirc* f){
        return (f -> tam == 0); /* Retorna verdadeiro se a Fila estiver vazia */
}
```

Remover da fila

```
char remove_fila (FilaCirc* f){
        if (fila_vazia(f)){ /* Verifica se a Fila está vazia */
                printf("Fila vazia\n");
        } else {  /* Caso a Fila contenha elemento, é removido o primeiro */
                f -> ini = (f -> ini % (N-1)) + 1;
                f -> tam--;
        }
}
```

Problemas com a utilização de fila

Considerando a utilização de uma fila, alguns problemas podem surgir.

Utilização de vetor

Se utilizarmos um vetor, teremos o problema de possuir um armazenamento de tamanho fixo e limitado, enquanto a fila pode crescer com a necessidade de uso do sistema. Para resolver essa problemática, teríamos que limitar o tamanho máximo da fila ao tamanho do vetor.

Fila cheia ou fila vazia

Em ambos os casos seria impossível realizar as operações. Como solução, é importante sempre implementar as funções para verificar se a fila está cheia (`fila_cheia(F)`) e para verificar se ela está vazia (`fila_vazia(F)`).

Identificação das posições da fila

Podem surgir problemas relacionados aos controles de início e fim de fila, em que não é possível identificar as posições em que se encontram. Como solução, é preciso incluir duas variáveis (`inicio` e `fim`) para armazenar a posição do início e do fim da fila, e sempre atualizar esses valores conforme a fila aumenta ou diminui.

Para visualizar o vídeo, acesse seu material digital.