



Algoritmos e Estrutura de Dados

Webaula 3

Operações e Problemas com Filas



Na webaula anterior estudamos sobre as operações e problemas com pilhas, e como implementar as funções necessárias para criar um código dessa estrutura. Para darmos prosseguimento em nossos estudos, nesta seção você vai conhecer e aprender sobre as operações de inserção, remoção e verificação e se uma fila está vazia ou não.



Inserir elementos na fila

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

Cria uma fila vazia.

Insere um elemento no fim da fila.

Remove o elemento do início da fila.

Verifica se a fila está vazia.

Libera a fila.





Explore a galeria e conheça a programação para efetuar a inserção dos elementos.

Para iniciar uma fila é necessário definir o tipo da estrutura a ser utilizada. A definição da estrutura pode ser dada pela implementação do trecho abaixo:

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





Remover elementos da fila

Como você já aprendeu, em uma fila só é possível remover um elemento pelo início. Veja como podemos implementar o trecho de código para remoção do elemento e apresentar seu valor no retorno da função:

```
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;  
}
```





Informar se a fila está vazia

Outra função que podemos implementar na fila, é a de verificação se a fila está vazia ou não. No trecho de remoção de um elemento da fila já realizamos a chamada a essa função, veja como ela pode ser implementada com o trecho do código:

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

Podemos utilizar uma função importante para podermos liberar a alocação de memória após o uso da fila e assim, limpar as variáveis utilizadas pelo sistema.

A implementação do trecho pode ser dado por:

```
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 essa situação. A fila circular possui a seguinte definição em sua implementação:

Um vetor para os elementos.

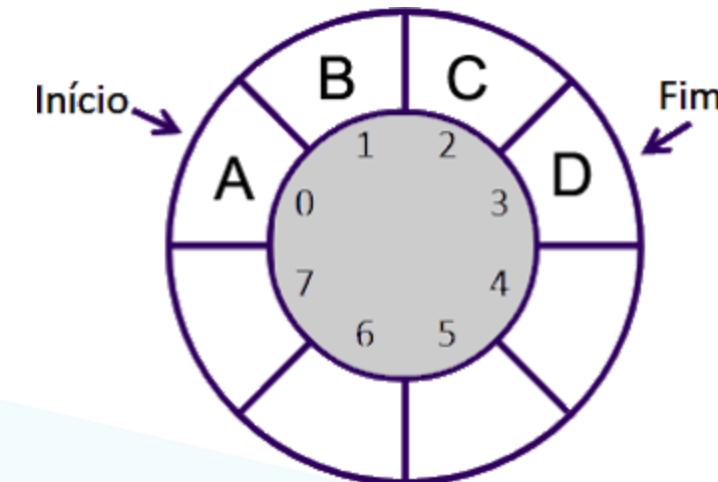
Um valor inteiro para o tamanho da fila.

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. Veja a imagem de uma estrutura da fila circular.



Fonte: elaborado pelo autor.



Explore a galeria e conheça a implementação de uma estrutura de fila circular.

```
/* 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;
```





Para a solução de desafios referentes a filas, aprendemos as estruturas de dados dinâmicas essenciais, bem como suas aplicações na solução de problemas. Com essas informações, será possível criar novos algoritmos muito bem estruturados e que serão utilizados para soluções de novos problemas



Vídeo de encerramento





Você já conhece o Saber?

Aqui você tem na palma da sua mão a **biblioteca digital** para sua **formação profissional**.

Estude no celular, tablet ou PC em qualquer hora e lugar sem pagar mais nada por isso.

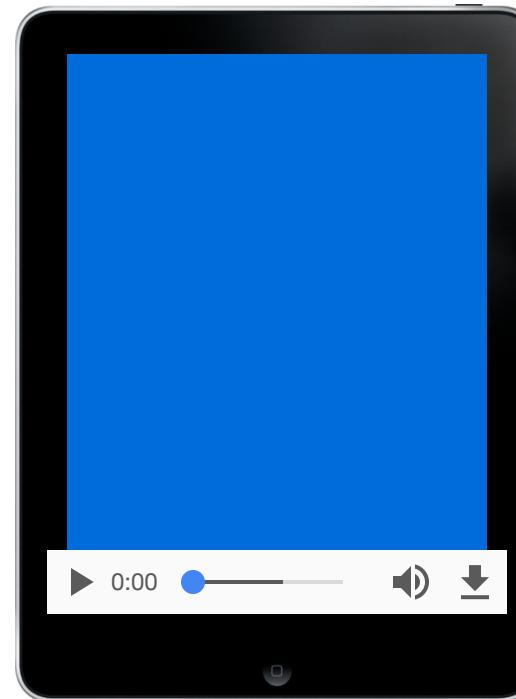
Mais de 450 livros com interatividade, vídeos, animações e jogos para você.



Android:
<https://goo.gl/yAL2Mv>



iPhone e iPad - IOS:
<https://goo.gl/OFWqcq>





Bons estudos!

