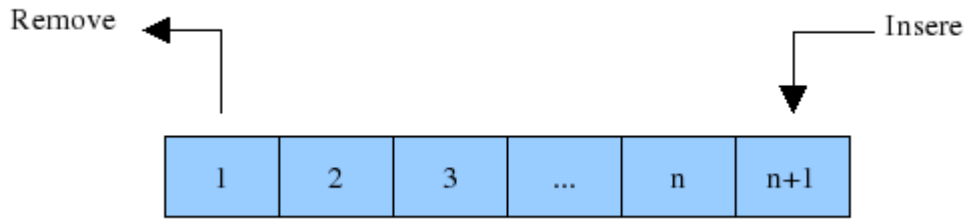
Fila Dinâmica em C

Material de suporte para a vídeo aula da estrutura de dados fila

1_ O que são Filas Dinâmicas?

- Fila é uma estrutura de dados linear que segue a regra do "primeiro a entrar, primeiro a sair" (FIFO - First-In-First-Out). Isso significa que o primeiro elemento inserido na fila será o primeiro a ser removido. Essa estrutura é comumente usada em situações em que a ordem de chegada dos elementos é significativa.



2_ Estrutura da Fila Dinâmica em C:

- 2.1 NODO:

```
typedef struct nodo *ptrNodo;
struct nodo {
                elemento elem;
                ptrNodo prox;
            };

typedef struct {
                ptrNodo inicio;
             ptrNodo fim;
                int tamanho;
              } fila;
```

```
typedef fila tipo_fila;
```

typedef struct nodo *ptrNodo;
struct nodo define um tipo de estrutura chamado nodo. Cada nodo contém um campo elemento e um ponteiro prox para o próximo nodo.
typedef struct nodo *ptrNodo; cria um novo tipo chamado ptrNodo que é um ponteiro para struct nodo. Isso facilita a declaração de ponteiros para nodo.
typedef struct { ... } fila;:
Aqui, uma nova estrutura é definida usando typedef e chamada fila.
fila possui três campos:
ptrNodo inicio: Um ponteiro para o primeiro nó na fila.
ptrNodo fim: Um ponteiro para o último nó na fila.
int tamanho: Uma variável para rastrear o tamanho atual da fila.
typedef fila tipo_fila;
Isso cria um alias tipo_fila para o tipo fila. Essencialmente, agora tipo_fila pode ser usado como um nome alternativo para struct fila.
Resumindo, o código define uma estrutura de dados de fila, onde cada elemento na fila é representado por um nó (struct nodo). A fila em si tem ponteiros para o início e fim, bem como uma variável para rastrear o tamanho. O uso de typedef simplifica a declaração de ponteiros para a fila e seus nós, e tipo_fila é outro nome para o tipo fila.

- 2.2 TAMANHO DA FILA:

```
int tamanho(fila f) {
  return f.tamanho;
}
```
int tamanho(fila f) {:

A função tamanho recebe como parâmetro uma fila (f), que é uma instância da estrutura fila definida anteriormente.
return f.tamanho;
A função retorna o valor armazenado no campo tamanho da fila f. Isso significa que ela retorna ao tamanho atual da fila.
Essencialmente, essa função serve como uma maneira de obter o tamanho de uma fila específica, pois ela retorna o valor armazenado no campo tamanho da fila que foi passada como argumento. Isso pode ser útil ao trabalhar com filas para monitorar e gerenciar seu tamanho durante operações de enfileiramento e desenfileiramento.

- 2.3 CRIAR FILA:

```
void criarFila(fila *f) {
        f->tamanho = 0;
        f->inicio  = NULL;
        f->fim     = NULL;
```

```
}
```

O método criarFila recebe um ponteiro para uma fila e atribui o valor 0 para seu tamanho e null para os ponteiros para o início da fila e o fim da fila.

- 2.4 FILA VAZIA:

```
int filaVazia(fila f) {
    return (f.inicio==NULL);
}
```

O método filaVazia verifica se a fila está vazia por meio de uma comparação, caso o ponteiro que aponta para o início da fila for igual a NULL significa que a fila é vazia.

- 2.5 ENTRAR ELEMENTO:

```
int entrarElemento(fila *f, elemento e) {
        ptrNodo pnodo;
        pnodo = (ptrNodo) malloc(sizeof(struct nodo));
        if (pnodo == NULL)
                return 0;
        else {
                pnodo->elem = e;
                pnodo->prox = NULL;

                if (f->fim != NULL)
                        f->fim->prox = pnodo;
                f->fim = pnodo;

                if (f->fim != NULL)
                        f->fim->prox = pnodo
                f->tamanho++;
                return 1;
        }
}
```

O método entrarElemento é responsável por enfileirar os elementos, neste método passamos um ponteiro para fila e um elemento qualquer. Primeiro, criamos um novo nó e alocamos memória para ele ptrNodo pnodo; pnodo = (ptrNodo) malloc(sizeof(struct nodo)); . Em sequência, se a alocação for bem

sucedida, passamos o elemento para dentro desse nó pnodo->elem = e; pnodo->prox = NULL; . Caso o fim da fila seja diferente de NULL, o nodo recém criado passa a ser o novo fim da fila: if (f->fim != NULL) f->fim->prox = pnodo . Caso a fila esteja vazia, o nó recém criado passa a ser o início da fila: if (f->fim != NULL) f->fim->prox = pnodo.

- 2.6 SAIR ELEMENTO:

```
int sairElemento(fila *f, elemento *e) {
        ptrNodo pnodo;
        if (filaVazia(*f))
                return 0;
        else {
                pnodo = f->inicio;
                f->inicio = f->inicio->prox;
                *e = pnodo->elem;
                free(pnodo);
                if (f->inicio == NULL)
                        f->fim = NULL;
                f->tamanho--;
                return 1;
        }
}
```

O método sairElemento é responsável por remover o elemento que está no início da fila. Nesse método, encontra-se um nó ptrNodo pnodo;. Primeiramente, é feita uma verificação para saber se a fila está vazia, caso não seja uma fila vazia passamos o início da fila para o nó pnodo: pnodo = f->inicio; , o início da fila passa a apontar para o nó próximo do início da fila: f->inicio = f->inicio->prox; . Por fim, o elemento recebe o elemento armazenado em pnodo: *e = pnodo->elem;.

EXERCÍCIOS:

Agora com base na vídeo aula e no material de suporte responda os exercícios abaixo:

1. Escreva uma função para esvaziar completamente uma fila dinâmica.

2. Crie uma função para imprimir todos os elementos de uma fila dinâmica.

3. Implemente uma função para inverter a ordem dos elementos em uma fila dinâmica.

4. Desenvolva uma função para copiar todos os elementos de uma fila dinâmica para outra fila dinâmica.

5. Escreva uma função para encontrar o maior elemento em uma fila dinâmica.

6. Implemente uma função para remover todos os elementos duplicados de uma fila dinâmica.

7. Crie uma função para concatenar duas filas dinâmicas.