



# Introdução a Filas (Conceitos Básicos)

< Estrutura de Dados Em C >



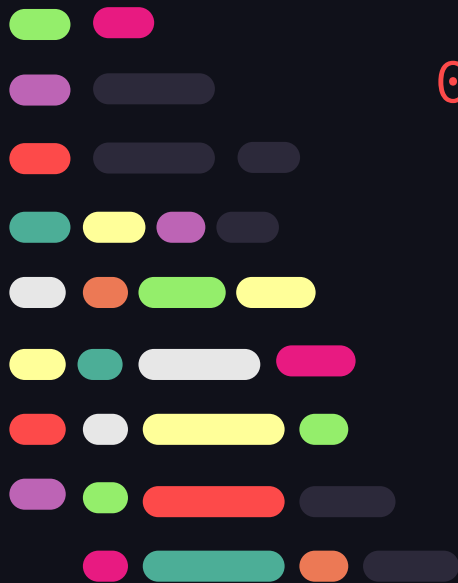
# Conteúdos Abordados

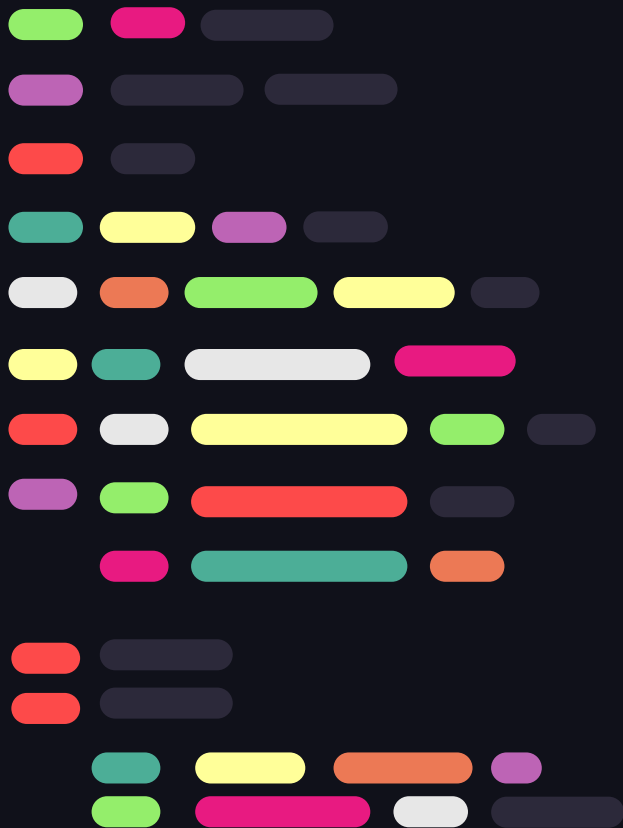
01 O que são Listas

02 Método de Busca

03 Método de Inserção

04 Método de Remoção





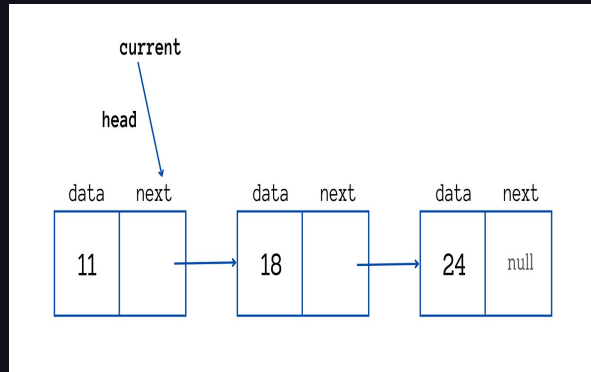
# O que são Listas?



< Como essa estrutura facilita a organização e implementação de operações eficientes como busca, inserção e remoção de elementos >



# Conceito



- Uma lista é como uma sequência ordenada de itens.
- Esses itens podem ser números, palavras, ou até mesmo outras listas;
- A ideia é organizar dados em uma ordem específica;
- Cada item na lista tem seu lugar definido, e podemos acessá-los facilmente.




# Conceito



- Existem dois tipos principais de listas:
- Listas Ligadas: cada elemento na lista aponta para o próximo, formando uma espécie de corrente.
- Listas Arrays: os elementos estão em posições fixas, como prateleiras numeradas.



# Para que servem?

- 
- Armazenamento de Dados;
  - Algoritmos de Busca e Ordenação;
  - Estruturas Dinâmicas;





# Estrutura

{

```
2  typedef struct nodo * ptr_nodo;
3  struct nodo {
4      elemento elem;
5      ptr_nodo prox;
6  };
7
8  typedef struct {
9      ptr_nodo lista;
10     int tamanho;
11 } lista_encadeada;
12
13 typedef lista_encadeada tipo_lista;
```



}

# Método de Buscar

```
int obter_elemento(lista_encadeada le, int i, elemento *e) {
    int j; ptr_nodo pnode;
    if ((i <= le.tamanho) && (i > 0)) {
        pnode = le.lista;
        for(j=2;j<=i;j++)
            pnode = pnode->prox;
        *e = pnode->elem;
        return 1;
    }
    else {
        *e = VL_NULO;
        return 0;
    }
}
```



# Método de Inserir

```
int incluir_elemento(lista_encadeada *le, int i, elemento e) {
    int j; ptr_nodo pnode_incluido, pnode_anterior;
    if ((i <= le->tamanho+1) && (i > 0)) {
        pnode_incluido = (ptr_nodo)malloc(sizeof(struct nodo));
        if (pnode_incluido == NULL)
            return 0; /* FALTA MEMORIA */
        else {
            pnode_incluido->elem = e;
            if (i == 1) {
                pnode_incluido->prox = le->lista;
                le->lista = pnode_incluido;
            }

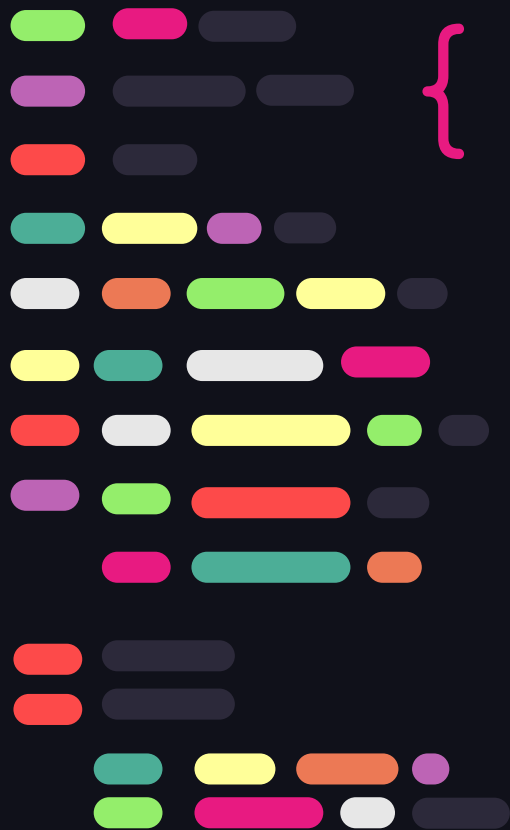
```

```
        else {
            pnode_anterior = le->lista;
            for (j=2; j<i; j++)
                pnode_anterior = pnode_anterior->prox;
            pnode_incluido->prox = pnode_anterior->prox;
            pnode_anterior->prox = pnode_incluido;
        }
        le->tamanho = le->tamanho + 1;
        return 1;
    }
    else
        return 0; /* POSIÇÃO INVÁLIDA */
}
```



# Método de Remover

```
int excluir_elemento(lista_encadeada *le, int i) {
    int j; ptr_nodo pnode_excluido, pnode_anterior;
    if ((i <= le->tamanho) && (i > 0)) {
        if (i == 1) {
            pnode_excluido = le->lista;
            le->lista = pnode_excluido->prox;
        }
        else {
            pnode_anterior = le->lista;
            for (j=2; j<i; j++)
                pnode_anterior = pnode_anterior->prox;
            pnode_excluido = pnode_anterior->prox;
            pnode_anterior->prox = pnode_excluido->prox;
        }
        free(pnode_excluido);
        le->tamanho = le->tamanho - 1;
        return 1;
    }
    else
        return 0; /* POSIÇÃO INVÁLIDA */
}
```



# Obrigada pela atenção!

< Dúvidas? >

Deixe nos comentários seus  
questionamentos ou considerações.

