1.A estrutura

Uma lista encadeada é uma sequência de nós onde cada contém duas partes:

- **Data:** valores armazenados neste nó
- **Ponteiro:** um ponteiro que aponta para o nó subsequente

Diferente de arrays, listas encadeadas não armazenam elementos de forma contínua na memória e sim, cada nó aponta para o próximo nó formando uma estrutura de nós em cadeia para se acessar cada elemento (ou nó). Embora, é possível fazer com que o armazenamento seja sequencial na memória.

Na linguagem C, listas encadeadas são representadas como ponteiros para o primeiro nó da lista e por esta razão, o primeiro nó é chamado de cabeça (head) da lista encadeada. Cada nó subsequente da lista encadeada é representado por uma *struct* que contém aquele(s) tipo(s) de dado(s) da estrutura e um ponteiro para o mesmo tipo.

```
struct Lista
{
   int data;
   Lista *próximo_nó;
}
```

Operação	Tipo de operação	Descrição	Complexidade Tempo/Espaço
Inserção	Começo	Novo nó no começo da lista	O(1) / O(1)
	Fim	Novo nó no fim da lista	O(N) / O(1)
	Específico	Novo nó após elemento específico	O(N) / O(1)
Remoção			
	Começo	Remove primeiro nó da lista	O(1) / O(1)
	Fim	Remove último nó da lista	O(N) / O(1)
	Específico	Remove nó específico	O(N) / O(1)
Busca		Percorre a lista do início ao fim	O(N) / O(1)

2.Aplicações

Suas aplicações podem ser feitas em:

- Alocação dinâmica de memória eficiente em sistemas e aplicações
- Implementação de estruturas de dados diferenciadas como filas e pilhas

- Representação e manipulação polinomial, com cada nó armazenando termos
- Utilizado em gerenciamento de arquivos de sistemas em sistemas operacionais

3.Vantagens

Algumas das vantagens de implementação

- Podem aumentar ou diminuir em tamanho dinamicamente, conforme a memória é alocada quando preciso
- Inserção e remoção são eficientes e não requerem a troca de elementos, diferentemente de arrays
- Maior eficiência no uso da memória, reduzindo espaço desperdiçado
- Podem ser utilizadas como blocos de memória não contínuos, melhorando aplicações onde a memória é fragmentada

4.Desvantagens

- Cada nó requer uma memória extra alocada para armazenamento do nó
- Não permitem acesso direto aos elementos da lista, fazendo com que seja necessário percorrer a lista até o elemento desejado
- Em busca, são mais lentos do que arrays
- Não suportam a busca inversa