Listas Dinâmicas

Prof. Edson Orivaldo Lessa Junior

Listas Dinâmicas

- Estrutura de dados em que os elementos são encadeados por meio de ponteiros, permitindo a inserção e remoção de elementos de forma flexível durante a execução do programa.
- Listas dinâmicas não possuem um tamanho fixo pré-determinado, o que as torna mais versáteis e eficientes em termos de gerenciamento de memória.

Características das Listas Dinâmicas

- Alocação Dinâmica de Memória:
  - Os elementos são alocados dinamicamente na memória conforme necessário, permitindo a expansão ou redução da lista de forma eficiente.
- Encadeamento de Elementos:
  - Cada elemento da lista possui um ponteiro que aponta para o próximo elemento, permitindo a navegação sequencial pela lista.
- Flexibilidade:
  - É possível inserir, remover e modificar elementos em qualquer posição da lista, sem a necessidade de realocar todos os elementos, como ocorre em arrays.

Características das Listas Dinâmicas

- Eficiência na Inserção e Remoção:
  - As operações de inserção e remoção de elementos em listas dinâmicas são mais eficientes do que em estruturas estáticas, especialmente em listas de grande tamanho.
- Versatilidade:
  - Existem diferentes variantes de listas dinâmicas, como listas duplamente encadeadas e listas circulares, que oferecem funcionalidades específicas, como percorrer a lista em ambas as direções ou criar estruturas cíclicas.
- Gerenciamento de Memória:
  - As listas dinâmicas permitem um melhor aproveitamento da memória, pois os elementos são alocados conforme a necessidade, evitando desperdício de espaço.

Nó em uma Lista Dinâmica

- Elemento que contém informações e referências a outros elementos, permitindo a construção de estruturas mais complexas e organizadas.
- Informação:
  - Um nó pode armazenar dados, como valores numéricos, strings, objetos, ou qualquer outra informação relevante para a aplicação em questão.
- Referência:
  - Além dos dados, um nó geralmente contém referências (ponteiros) que apontam para outros nós na estrutura de dados. Essas referências são essenciais para estabelecer conexões e relações entre os elementos.
- Permitem a organização e armazenamento de informações de forma eficiente e estruturada.

Componentes de um Nó em uma Lista Dinâmica

- Cada nó contém dois principais componentes: o dado a ser armazenado e um ponteiro que aponta para o próximo nó na sequência.
- Essa estrutura básica permite a ligação entre os elementos da lista, possibilitando a navegação e manipulação dos dados de forma encadeada.

Componentes de um Nó em uma Lista Dinâmica

- Dado: O dado é a informação que o nó armazena. Pode ser de qualquer tipo de dado, como um número, uma string, um objeto, entre outros, dependendo da aplicação da lista.
- Ponteiro para o Próximo Nó: O ponteiro é uma referência que aponta para o próximo nó na sequência da lista. Ele é responsável por estabelecer a conexão entre os nós, permitindo percorrer a lista de forma sequencial.

Estrutura de um Nó em C

```
struct Node {
    int data; // Dado armazenado no nó
    struct Node* next; // Ponteiro para o próximo nó
};
```

Exemplo de um Nó em uma Lista Dinâmica

- Dado: 100
- Ponteiro para o Próximo Nó: Endereço de memória do próximo nó na lista
- Cada nó contém um dado (no exemplo, o número 100) e um ponteiro que aponta para o próximo nó na sequência.
- Ao percorrer a lista, o programa utiliza os ponteiros para acessar os dados e os próximos nós, permitindo acesso dos elementos de forma encadeada.

Listas dinâmicas vs Vetores.

- Tamanho Flexível:
  - Listas dinâmicas: Tamanho pode ser facilmente ajustado durante a execução do programa, permitindo a inserção e remoção de elementos de forma dinâmica.
  - Vetores: Têm um tamanho fixo que precisa ser definido previamente.
- Inserção e Remoção Eficientes:
  - Listas dinâmicas: A inserção e remoção de elementos em qualquer posição da lista são operações eficientes. Não exigem realocações de memória em massa como nos vetores.
  - Vetores: A inserção e remoção no meio da estrutura podem ser custosas devido à necessidade de deslocar os elementos subsequentes.
- Uso Eficiente de Memória:
  - Listas dinâmicas: a memória é alocada conforme necessário para cada novo nó, o que evita desperdício de espaço.
  - Vetores: Alocação uma quantidade fixa de memória, mesmo que nem todos os elementos sejam utilizados.

Listas dinâmicas vs Vetores.

- Facilidade de Implementação de Estruturas Complexas:
  - Listas dinâmicas: Facilitam a implementação de estruturas de dados mais complexas, listas encadeadas, árvores, entre outras, devido à sua natureza flexível e encadeada.
- Manipulação de Grandes Conjuntos de Dados:
  - Listas dinâmicas: Para conjuntos de dados grandes e variáveis, são mais adequadas, pois permitem a gestão eficiente de elementos sem a necessidade de realocações constantes de memória.
- Facilidade de Reordenamento:
  - Listas dinâmicas: É mais simples reordenar os elementos, pois basta ajustar os ponteiros entre os nós.
  - Vetores: Reordenar os elementos pode exigir a realocação de todos os dados.
- Suporte a Estruturas Encadeadas Complexas:
  - Listas dinâmicas: permitem a criação de estruturas encadeadas complexas, como listas duplamente encadeadas, listas circulares, entre outras, o que pode ser mais difícil de implementar com vetores.

Componentes de um Nó em uma Lista Dinâmica

- Campo de Dados: É onde a informação a ser armazenada na lista é guardada. Pode ser de qualquer tipo de dado, como números, strings, objetos, entre outros, dependendo da aplicação da lista.
- Ponteiro para o Próximo Nó: Este ponteiro aponta para o próximo nó na sequência da lista. É o que estabelece a ligação entre os nós, permitindo percorrer a lista de forma encadeada.

Funcionamento de um Nó em uma Lista Dinâmica

- Cada nó contém um campo de dados para armazenar a informação e um ponteiro para o próximo nó na lista.
- O último nó da lista tem seu ponteiro apontando para NULL, indicando o final da sequência.
- Através dos ponteiros, é possível percorrer a lista, acessar os dados armazenados em cada nó e realizar operações como inserção, remoção e busca de elementos.

Benefícios de Utilizar Nós em uma Lista Dinâmica

- Flexibilidade na inserção e remoção de elementos em qualquer posição da lista.
- Eficiência na manipulação de grandes conjuntos de dados, sem a necessidade de realocação completa da estrutura.
- Possibilidade de implementar diferentes variantes de listas dinâmicas, como listas duplamente encadeadas ou circulares, para necessidades específicas.

Implementando uma Lista Dinâmica

Estrutura de um nó em uma lista dinâmica

- Em uma lista dinâmica, a estrutura de um nó é fundamental para armazenar os dados e manter a referência para o próximo nó na sequência.
- A estrutura de um nó em uma lista dinâmica geralmente contém dois elementos principais: o campo de dados e o ponteiro para o próximo nó.

Estrutura de um nó em uma lista dinâmica

```
struct No {
    TipoDado dado;
// Campo de dados que armazena a informação relevante
    struct No* proximo;
// Ponteiro para o próximo nó na lista
};
```

Estrutura de um nó em uma lista dinâmica

- TipoDado:
  - Representa o tipo de dado que será armazenado no nó da lista dinâmica.
  - Pode ser um tipo primitivo (como int, float, char) ou um tipo personalizado (como uma estrutura ou classe).
- dado:
  - Campo de dados que armazena a informação específica associada a esse nó na lista.
- struct No* proximo:
  - Ponteiro que aponta para o próximo nó na sequência da lista.
  - Esse ponteiro é responsável por encadear os nós e percorrer a lista de forma sequencial.

Estrutura de um nó em uma lista dinâmica

- Exemplo de Utilização:

```
struct No {
    int valor;        // Campo de dados que armazena um valor inteiro
    struct No* proximo;  // Ponteiro para o próximo nó na lista
};
```

- Criação de um Nó:
  - Para criar um novo nó em uma lista dinâmica, é necessário alocar dinamicamente memória para o nó e atribuir valores aos campos de dados e ao ponteiro para o próximo nó.
- Exemplo de Criação de um Nó:
  - struct No* novoNo = (struct No*)malloc(sizeof(struct No));
  - novoNo->valor = 42;   // Atribui o valor 42 ao campo de dados do nó
  - novoNo->proximo = NULL; // Define o ponteiro para o próximo nó como

Exemplo

- https://github.com/edson-lessa-jr/unicesumar-aula5-estrutura-de-dados/

BONS ESTUDOS