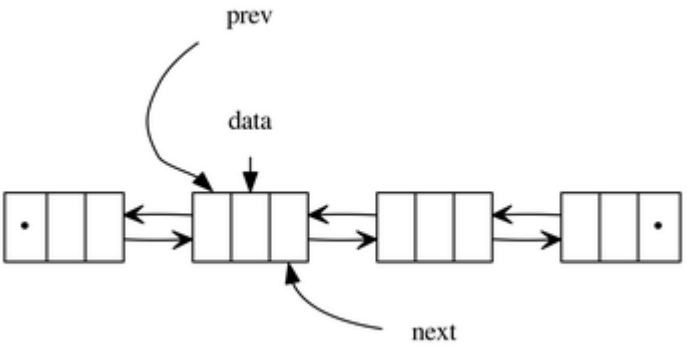
Listas encadeadas em C

Material de suporte para a vídeo aula da estrutura de dados lista.

1 - O que são Listas ?

- Uma lista encadeada é uma estrutura de dados na qual os elementos são armazenados de forma sequencial na memória do computador, e cada elemento aponta para o próximo na sequência. Cada elemento em uma lista encadeada é chamado de "nó" e contém dois campos principais: um campo de dados para armazenar a informação propriamente dita e um ponteiro (ou referência) para o próximo nó na sequência. Existem diversas variações de listas, tais como: lista duplamente encadeada, lista circular etc.



2_ Estrutura da Lista Dinâmica:

- 2.1 Nó

```
typedef struct No {
        int dado;
        struct No* proximo;
} No;
```

A estrutura básica do nó de uma lista é composta por um campo que armazena o elemento, neste caso um valor inteiro (int dado) e um ponteiro para um nó, denominado de próximo.

- 2.2 Criar nó

```
No* criarNo(int valor) {
        No* novoNo = (No*)malloc(sizeof(No));
        novoNo->dado = valor;
        novoNo->proximo = NULL;
        return novoNo;
}
```

O método de criarNo, que recebe um valor contendo um elemento, é usado para instanciar um novo nó.

- 2.3 Inserir no início

```
void inserirNoInicio(No** cabeca, int valor) {
          No* novoNo = criarNo(valor);
          novoNo->proximo = *cabeca;
        *cabeca = novoNo;
}
```

O método inserirNoInicio é responsável por inserir os elementos no início da lista. Instanciamos um novo nó `No* novoNo = criarNo(valor);` . Após isto, o ponteiro para o próximo do novo nó passar a apontar para o início atual da lista: `novoNo->proximo = *cabeca;` e o início da lista passa a apontar para o novo nó : `*cabeca = novoNo;`.

- 2.4 Buscar Elemento

```
No* buscarElemento(No* cabeca, int valor) {
   No* atual = cabeca;

   while (atual != NULL) {
      if (atual->dado == valor) {
         return atual;
      }
      atual = atual->proximo;
   }

   return NULL;  // Elemento não encontrado
}
```

O método buscarElemento é responsável por realizar uma busca linear na lista de modo a percorrer cada nó até que se encontre o elemento buscado. Para percorrer a lista, criamos um nó que aponta para o início da lista `No* atual = cabeca;` , dessa forma é possível percorrer a lista inteira. Depois é feito um laço de repetição que percorre a lista, caso o valor de algum nó coincida com o elemento buscado, o valor é retornado, caso contrário é retornado NULL.

- 2.5 Modificar Elemento:

```
// Função para alterar o valor de um elemento na lista
void modificarElemento(No* cabeca, int valorAntigo, int valorNovo) {
    No* noParaModificar = buscarElemento(cabeca, valorAntigo);

    if (noParaModificar != NULL) {
        noParaModificar->dado = valorNovo;
    } else {
        printf("Elemento não encontrado para modificação.\n");
    }
}
```

O método de modificar elemento consiste em buscar o nó que contém o valor a ser modificado, após isso, o valor novo passado na função é atribuído ao nó que será modificado.

- 2.6 Remover Elemento:

```
// Função para remover um elemento da lista
void removerElemento(No** cabeca, int valor) {
    No* atual = *cabeca;
    No* anterior = NULL;

    while (atual != NULL && atual->dado != valor) {
        anterior = atual;
        atual = atual->proximo;
    }
```

```
    if (atual == NULL) {
        printf("Elemento não encontrado.\n");
        return;
    }

    if (anterior == NULL) {
        *cabeca = atual->proximo;
    } else {
        anterior->proximo = atual->proximo;
    }

    free(atual);
}
```

Este método consiste em percorrer a lista até que se encontre o nó com o valor desejado. Caso o nó atual seja igual a NULL, então o elemento não está presente na lista. Verifica se o nó a ser removido é o primeiro nó da lista (verificando se anterior é NULL).
Se o nó atual for o primeiro nó, atualiza o ponteiro da cabeça (*cabeca) para apontar para o próximo nó da lista. Se não for o primeiro nó, atualiza o ponteiro do nó anterior (anterior->proximo) para "pular" o nó a ser removido.

EXERCÍCIOS:

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

1 - Uma lista ligada é uma estrutura de dados na qual os objetos estão organizados em ordem linear. Entretanto, diferentemente de um arranjo, no qual a ordem linear é determinada pelos índices do arranjo, a ordem em uma lista ligada é determinada por um ponteiro em cada objeto.
Em relação à tabela de espalhamento, segundo Cormen (2012), analise os itens a seguir:

I. Uma lista pode ter uma entre várias formas; ela pode ser simplesmente ligada ou duplamente ligada, pode ser ordenada ou não e pode ser circular ou não.
II. Se uma lista é simplesmente ligada, omitimos o ponteiro anterior em cada elemento.
III. Se a lista é não ordenada, os elementos podem aparecer em qualquer ordem.

Está **CORRETO** o que se afirma em:

A Nenhum dos itens é verdadeiro.

B I e II, apenas.

C II e III, apenas.

D I e III, apenas.

E I, II e III.

2 - Uma _____ é uma sequência finita de elementos ligados entre si, onde uma célula da dela, aponta para a próxima célula sequencialmente. Elas são úteis para representar conjuntos dinâmicos de dados. Assinale a alternativa que preencha corretamente a lacuna acima.

A árvore.

B lista.

C pilha.

D fila.

E tabela espelhada