FAESA CENTRO UNIVERSITÁRIO

Lista Dinâmica

Apresentação

As listas dinâmicas ou encadeadas são normalmente utilizadas para relacionar itens que precisam ser exibidos ou manipulados em tempo de execução. Com isso, é possível implementar métodos que permitem redimensionar as listas conforme os itens são inseridos ou removidos, ocupando exatamente a memória necessária para alocar os elementos que pertencem à listagem. Conhecer a implementação de listas dinâmicas e seus métodos de acesso e manipulação permite que o profissional de TI identifique e aplique soluções robustas e otimizadas para o desenvolvimento de sistemas.

Nesta Unidade de Aprendizagem, você vai estudar sobre listas dinâmicas e suas operações de manipulação e acesso com a linguagem C.

Bons estudos.

Ao final desta Unidade de Aprendizagem, você deve apresentar os seguintes aprendizados:

- Declarar uma lista dinâmica.
- Construir funções de manipulação de uma lista dinâmica.
- Desenvolver funções de acesso a uma lista dinâmica.

Desafio

Uma lista dinâmica é um conjunto de elementos ligados em sequência. Nesse conjunto, os elementos estão encadeados de modo que cada um deles aponte para o próximo, formando assim uma cadeia.

A grande vantagem de uma lista encadeada é a alocação/desalocação dinâmica de memória, onde a lista cresce ou diminui de acordo com a quantidade de elementos atuais, ocupando apenas a memória necessária.

Com base nisso, você deve declarar uma lista dinâmica e implementar os seguintes métodos na linguagem C:

- - Inicializar a lista.
- - Inserir elemento no início.
- - Inserir elemento no final.
- - Excluir elemento do início.
- - Excluir elemento do final.
- - Imprimir a lista.
- - Excluir toda a lista.

Use como base da lista os seguintes atributos: nome, idade e endereço.

Após implementar a lista e os métodos, faça algumas operações de inserção e exclusão para testar as funcionalidades do exercício.

Infográfico

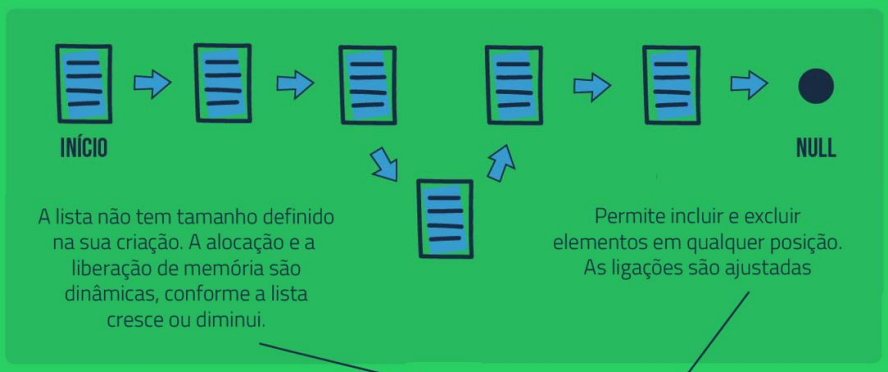
As listas dinâmicas são importantes para armazenar elementos que podem ser incluídos e excluídos em qualquer posição e a qualquer momento, sem que se conheça, necessariamente, o tamanho da lista no momento de sua criação. A principal vantagem de uma lista dinâmica está na capacidade de usar apenas a memória necessária para acomodar seus elementos, ou seja, aloca memória quando insere um novo elemento e libera quando realiza uma exclusão, sempre reajustando os apontamentos para manter a lista ligada e em sequência.

Veja, no Infográfico a seguir, o comportamento de uma lista dinâmica ao incluir e excluir elementos.

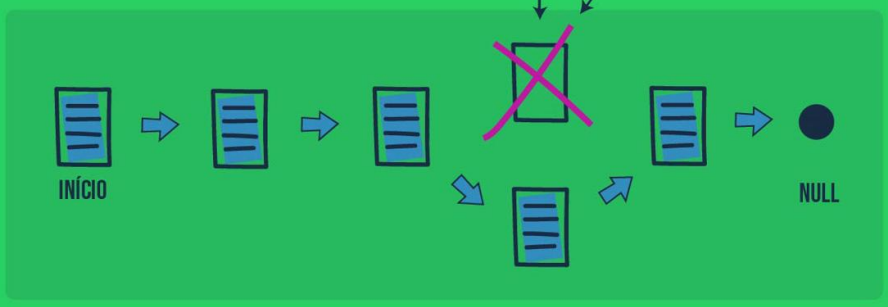
LISTA DINÂMICA E SEU COMPORTAMENTO AO INCLUIR E EXCLUIR ELEMENTOS

1 - LISTA COM ALGUNS ELEMENTOS:

2 - INCLUSÃO DE ELEMENTO NA LISTA:



3 - EXCLUSÃO DE ELEMENTO NA LISTA



Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

Conteúdo do Livro

Listas, de um modo geral, são utilizadas para representar dados em séries ou para apresentar itens de determinado domínio de aplicação, como a listagem de produtos em estoque de determinada loja. Conhecer as estruturadas baseadas em listas dinâmicas com encadeamento simples representa um passo na caminhada do conhecimento sobre manipulação de estruturas de dados, uma vez que elas servirão de base para outras disciplinas, como listas duplamente encadeadas, pilhas, filas e árvores binárias.

No capítulo Lista dinâmica, da obra *Estrutura de dados*, base teórica desta Unidade de Aprendizagem, você vai estudar sobre listas dinâmicas com encadeamento simples, bem como sobre a implementação dos seus métodos de manipulação e acesso.

Boa leitura!

ESTRUTURA DE DADOS

Maurício de Oliveira Saraiva

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS

Lista dinâmica

Objetivos de aprendizagem

Ao final deste texto, você deve apresentar os seguintes aprendizados:

- Definir uma lista dinâmica.
- Construir funções de manipulação de uma lista dinâmica.
- Desenvolver funções de acesso a uma lista dinâmica.

Introdução

As listas dinâmicas ou encadeadas são normalmente utilizadas para relacionar itens que precisam ser exibidos ou manipulados em tempo de execução. Com isso, é possível implementar métodos que permitem redimensionar as listas conforme os itens são inseridos ou removidos, ocupando exatamente a memória necessária para alocar os elementos que pertencem à lista. Conhecer a implementação de listas dinâmicas e seus métodos de acesso e manipulação permite que o profissional de TI (tecnologia da informação) identifique e aplique soluções robustas e otimizadas para o desenvolvimento de sistemas.

Neste capítulo, você vai estudar sobre listas dinâmicas e suas operações de manipulação e acesso com a linguagem *C*.

Lista dinâmica

Lista dinâmica, ou lista encadeada simples, é uma relação de elementos ligados, formando uma sequência de itens na qual cada elemento se liga ao elemento posterior. Os elementos dessa relação são formados por uma estrutura que pode conter variáveis de vários tipos de dados (Figura 1) (EDELWEISS; GALANTE, 2009).

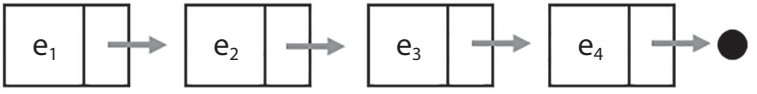


Figura 1. Apresentação de uma lista dinâmica.

De acordo com Celes, Cerqueira e Rangel (2004), uma lista dinâmica apresenta as seguintes características:

- Define um ponteiro para o início da lista. Esse ponteiro serve como ponto de referência da lista, apontando para o primeiro elemento, quando houver, ou *NULL* se a lista estiver vazia.
- Cada elemento da lista tem um atributo do tipo ponteiro da lista, o qual aponta para o próximo elemento.
- O último elemento da lista não aponta para *NULL*, isto é, para nenhum outro elemento, caracterizando o final da lista.
- Os elementos da lista podem ser variáveis de tipo primitivo ou estruturas compostas por outras variáveis, como registros.

Na linguagem *C*, a declaração de uma lista dinâmica pode ser implementada por meio de duas estruturas. A primeira especifica os detalhes do que será armazenado como um registro, isto é, os atributos que compõem esse conjunto de dados; e a segunda define propriamente a lista (LORENZI; MATTOS; CARVALHO, 2007).

Para exemplificar a declaração de uma lista dinâmica, tomaremos por base o cadastro de produtos de determinada loja. Esse cadastro será composto por quatro atributos: código de barras, descrição do produto, categoria e preço.

```
typedef struct {
    char codigoBarras[8];
    char descricao[20];
    int categoria;
    float preco;
} Produto;
```

A estrutura declarada nada mais é que a definição de um conjunto capaz de armazenar os dados dos produtos. No entanto, para declarar a lista, é preciso definir uma estrutura que armazene um dado de produto e um ponteiro que será utilizado para fazer a ligação dos elementos.

```
struct Nodo{
    Produto dado;
    struct Nodo *prox;
};
typedef struct Nodo nodo;
```

Nessa estrutura, a variável *dado* é do tipo *Produto*, isto é, armazena os atributos de produto, como código de barras, descrição e outros. Além disso, define a variável **prox*, que é um ponteiro para a mesma estrutura, o qual visa a armazenar o endereço do próximo elemento da lista, conforme ilustrado na Figura 2.

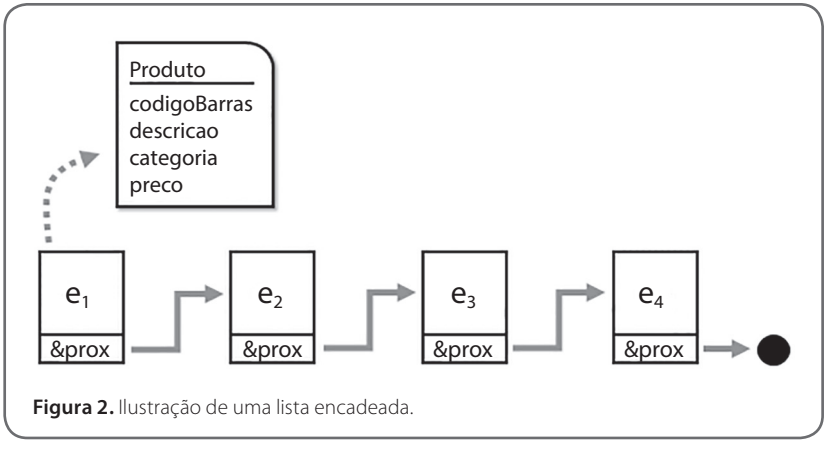


Figura 2. Ilustração de uma lista encadeada.

Por ser uma lista dinâmica, a alocação e a desalocação de memória ocorrem em tempo de execução. Assim, a lista ocupa apenas o espaço necessário para acomodar os elementos atuais, tornando-se uma ótima opção para os casos em que não se sabe exatamente o tamanho da lista.

Uma vez que as estruturas estejam definidas, é preciso declarar as variáveis que serão usadas para montar a lista. Para isso, vamos declarar uma variável do tipo *Produto* e uma variável que representa a lista, do tipo *Nodo*.

```
Produto produto;
nodo *L= (nodo *) malloc(sizeof(nodo));
```

É preciso alocar memória para cada elemento a ser incluído na lista. Na linguagem *C*, a instrução *malloc* realiza essa alocação com base no tamanho da estrutura *Nodo* que, além de armazenar os dados do *Produto*, armazena o endereço do próximo *Nodo*.

Construir funções de manipulação de uma lista dinâmica

As operações de manipulação de uma lista dinâmica efetuam as ações que modificam a lista, como incluir, excluir e alterar elementos. As operações de inclusão e exclusão podem ser realizadas em qualquer posição da lista, com tratamentos diferenciados para cada caso. Veja na sequência deste capítulo as principais operações de manipulação de listas dinâmicas.

Inicializar uma lista

Antes de inserir qualquer elemento em uma lista dinâmica, é necessário inicializá-la. Para isso, basta definir como *NULL* a variável que identifica o próximo elemento da lista.

```
void iniciar(nodo *L) {
    L->prox= NULL;
}
```

Incluir elemento no início da lista

Incluir um novo elemento no início da lista é uma tarefa simples. Para isso, é preciso ajustar o apontamento inicial da lista, que vai indicá-lo como primeiro elemento, definindo como seu próximo elemento aquele que ocupava essa posição anteriormente.

Antes de incluir um elemento no início da lista, é preciso alocar memória e atribuir os dados do produto para a estrutura. Na sequência, fazemos o novo elemento, que foi criado, apontar para o próximo do início da lista e o início da lista apontar para o novo elemento.

```
void inserirInicio(nodo *L, Produto *dado) {
     nodo *novo= (nodo *) malloc(sizeof(nodo));

     strcpy(novo->dado.codigoBarras, dado->codigoBarras);
     strcpy(novo->dado.descricao, dado->descricao);
     novo->dado.categoria= dado->categoria;
     novo->dado.preco= dado->preco;

     novo->prox= L->prox;
     L->prox= novo;
}
```

Incluir no meio da lista

A inclusão de um elemento em uma posição intermediária deve ser realizada pela varredura da lista até a posição desejada. Essa posição pode ser identificada, por exemplo, por algum atributo da estrutura, como a descrição do produto.

Assim que a posição for identificada, é preciso ajustar os apontamentos para manter a lista ligada e sequencial. Dessa forma, deve-se fazer o elemento anterior apontar para o novo elemento e esse para o elemento que era o posterior.

```
void inserirMeio(nodo *L, Produto *dado, char *produto) {
    nodo *novo= (nodo *) malloc(sizeof(nodo));

    strcpy(novo->dado.codigoBarras, dado->codigoBarras);
    strcpy(novo->dado.descricao, dado->descricao);
    novo->dado.categoria = dado->categoria;
    novo->dado.preco = dado->preco;

    if (estaVazia(L))
          L->prox= novo;
    else {
          nodo *tmp= NULL;
          nodo *no= L->prox;
          while(no != NULL) {
                if (strcmp(no->dado.descricao, produto) == 0) {
                      tmp= no->prox;
                      no->prox= novo;
                      novo->prox= tmp;
                }
                no= no->prox;
          }
    }
}
```

Incluir no final da lista

Inserir um elemento no final da lista também é uma tarefa simples. Basta percorrer a lista até a última posição e fazer com que o último elemento aponte para o novo e este para *NULL*, pois o último elemento deve, obrigatoriamente, apontar para *NULL* para indicar o final da lista.

```
void inserirFinal(nodo *L, Produto *dado) {
    nodo *novo= (nodo *) malloc(sizeof(nodo));

    strcpy(novo->dado.codigoBarras, dado->codigoBarras);
    strcpy(novo->dado.descricao, dado->descricao);
    novo->dado.categoria= dado->categoria;
    novo->dado.preco= dado->preco;

    if (estaVazia(L))
          L->prox= novo;
    else {
          nodo *no= L->prox;
          while(no->prox != NULL)
                no= no->prox;

          no->prox= novo;
    }
}
```

Excluir no início da lista

Para excluir o elemento do início da lista, é preciso fazer com que a lista inicie pelo elemento que era sucessor do elemento que foi excluído, além de liberar a memória desse elemento que deixou de pertencer à lista.

```
void excluirInicio(nodo *L) {
     nodo *noPrimeiro= L->prox;
     L->prox= noPrimeiro->prox;
     free(noPrimeiro);
}
```

Excluir no meio da lista

Para excluir um elemento que ocupa uma posição intermediária na lista dinâmica, é preciso percorrer a lista até encontrar a posição desejada, identificando o elemento anterior e o sucessor. A partir disso, basta fazer o anterior apontar para o sucessor, além de liberar a memória que estava alocada para o elemento que foi excluído.

```
void excluirMeio(nodo *L, char *produto) {
     nodo *noAnterior= L;
     nodo *noAtual= L->prox;
     while(noAtual->prox != NULL) {
           if (strcmp(noAtual->dado.descricao, produto) == 0) {
                 noAnterior->prox= noAtual->prox;
                 free(noAtual);
                 return;
           }
           noAnterior= noAtual;
           noAtual= noAtual->prox;
     }
}
```

Excluir no final da lista

Para excluir o elemento do final da lista, é necessário realizar uma varredura na lista até chegar à última posição e definir o apontamento do penúltimo elemento como *NULL*, bem como liberar a memória do elemento que ocupava a última posição.

```
void excluirFinal(nodo *L) {
     nodo *noAnterior= L;
     nodo *noAtual= L->prox;
     while(noAtual->prox != NULL) {
           noAnterior= noAtual;
           noAtual= noAtual->prox;
     }
     noAnterior->prox= NULL;
     free(noAtual);
}
```

Excluir toda a lista

Para excluir uma lista dinâmica inteira, é preciso percorrê-la e liberar a alocação de memória de todos os elementos, bem como indicar que o próximo elemento de cada item aponte para *NULL*, garantindo, assim, a destruição completa da lista.

```
void liberar(nodo *L) {
     nodo *proximo;
     nodo *atual;
     atual= L;
     while(atual->prox != NULL){
           proximo= atual->prox;
           atual->prox= NULL;
           free(atual);
           atual= proximo;
     }
}
```

Desenvolver funções de acesso a uma lista dinâmica

As operações de acesso a uma lista dinâmica realizam as ações de leitura e impressão sem aplicar qualquer modificação nos elementos. Essas operações são consideradas ações de apoio à manipulação de dados, pois atuam sobre os dados já inseridos na lista.

Podem ser criadas diversas operações de acesso a uma lista, sendo as principais: verificar se a lista está vazia ou cheia, identificar o primeiro e o último elementos, pesquisar um elemento em qualquer posição e imprimir a relação completa de elementos da lista.

Verificar se a lista está vazia

Uma lista dinâmica está vazia se ela aponta para *NULL*. Isso significa que nenhum elemento foi inserido ou que todos já foram excluídos. A função que identifica se uma lista está vazia está descrita a seguir.

```
int estaVazia(nodo *L) {
     if (L->prox == NULL)
           return 1;
     else
           return 0;
}
```

Acessar primeiro elemento da lista

Para acessar o primeiro elemento da lista, basta pegar o próximo elemento do endereço da lista. Verificar se a lista está vazia é cabível para apresentar a mensagem correta, no caso de a lista não conter elementos.

```
void primeiro(nodo *L) {
     if (estaVazia(L)) {
           printf("Lista vazia!\n\n");
           return;
     }

     nodo *no= L->prox;
     printf("%-8s %s \t%s %s \n",
            "Codigo", "Descricao", "Valor", "Categoria");
     printf("%-8s %s \t%-6.2f %d \n", no->dado.codigoBarras,
        no->dado.descricao, no->dado.preco, no->dado.categoria);
     printf("\n\n");
}
```

Acessar último elemento da lista

Não é possível acessar diretamente o último elemento de uma lista dinâmica, uma vez que o endereço de cada elemento só é conhecido pelo seu sucessor. Com base nisso, é preciso percorrer toda a lista a partir do primeiro elemento.

```
void ultimo(nodo *L) {
     if (estaVazia(L)) {
           printf("Lista vazia!\n\n");
           return;
     }

     nodo *no= L->prox;
     printf("%-8s %s \t%s %s \n",
            "Codigo", "Descricao", "Valor", "Categoria");
     while(no != NULL){
           if (no->prox == NULL) {
                 printf("%-8s %s \t%-6.2f %d \n",
                   no->dado.codigoBarras, no->dado.descricao,
                   no->dado.preco, no->dado.categoria);
           }
           no= no->prox;
     }
     printf("\n\n");
}
```

Pesquisar elemento na lista

Sabendo que uma lista dinâmica não permite acessar diretamente os elementos, é preciso realizar uma varredura na lista a partir da primeira posição. Dessa forma, compara-se cada elemento da lista com o dado pesquisado, buscando localizar a informação solicitada.

```
void pesquisar(nodo *L, char *descricao) {
    if (estaVazia(L)) {
          printf("Lista vazia!\n\n");
          return;
    }

    nodo *no= L->prox;
    printf("%-8s %s \t%s %s \n",
           "Codigo", "Descricao", "Valor", "Categoria");
    while(no != NULL){
          if (strcmp(no->dado.descricao, descricao) == 0) {
                printf("%-8s %s \t%-6.2f %d \n",
                   no->dado.codigoBarras, no->dado.descricao,
                   no->dado.preco, no->dado.categoria);
          }
          no= no->prox;
    }
    printf("\n\n");
}
```

Imprimir toda a lista

Para imprimir a relação completa de elementos de uma lista dinâmica, é necessário percorrê-la a partir da primeira posição, até que o último elemento aponte para *NULL*.

```
void imprimir(nodo *L) {
    if (estaVazia(L)) {
          printf("Lista vazia!\n\n");
          return;
    }

    nodo *no= L->prox;
    printf("%-8s %s \t%s %s \n",
           "Codigo", "Descricao", "Valor", "Categoria");
    while(no != NULL){
          printf("%-8s %s \t%-6.2f %d \n",
             no->dado.codigoBarras, no->dado.descricao,
             no->dado.preco, no->dado.categoria);
          no= no->prox;
    }
    printf("\n\n");
}
```

Referências

CELES, W.; CERQUEIRA, R.; RANGEL, J. L. *Introdução a estrutura de dados*: com técnicas de programação em C. Rio de Janeiro: Campus, 2004. 294 p.EDELWEISS, N.; GALANTE, R. *Estrutura de dados*. Porto Alegre: Bookman, 2009. 262 p. (Série Livros Didáticos Informática UFRGS, 18).

LORENZI, F.; MATTOS, P.; CARVALHO, T. *Estruturas de dados*. São Paulo: Cengage Learning, 2007. 200 p.

Leituras recomendadas

DEITEL, P.; DEITEL, H. *C*: como programar. 6. ed. São Paulo: Pearson Education, 2011. 818 p.

FORBELLONE, A. L. V.; EBERSPÁCHER, H. F. *Lógica de programação*: a construção de algoritmos e estruturas de dados. 3 ed. São Paulo, Pearson, 2005. 232 p.

Encerra aqui o trecho do livro disponibilizado para esta Unidade de Aprendizagem. Na Biblioteca Virtual da Instituição, você encontra a obra na íntegra.

Conteúdo:

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS

Dica do Professor

Listas dinâmicas com encadeamento simples são estruturas que realizam a alocação de memória de modo eficiente, isto é, alocam memória quando precisam incluir um elemento e liberam quando realizam a exclusão. Existem várias formas de incluir um elemento em uma lista encadeada simples, sendo a inclusão no final da lista a mais comum, pois normalmente os elementos são inseridos nessa etapa, mantendo a ordem original de inserção.

Assista à Dica do Professor para conhecer a implementação do método em linguagem C, que insere elementos no final de uma lista dinâmica com encadeamento simples.

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

Exercícios

1) **Em relação às listas dinâmicas com encadeamento simples, leia as alternativas a seguir e indique a correta:**

A) Alocam a memória necessária para armazenar todos os elementos da lista na sua declaração.

B) Cada elemento aponta para o endereço dos elementos anterior e posterior.

C) Podem ser percorridas em qualquer sentido.

D) Alocam apenas a memória necessária para armazenar os elementos atuais da lista .

E) Podem acessar diretamente a posição de um elemento da lista.

2) **Baseando-se no conceito de lista dinâmica encadeada simples, marque a alternativa correta:**

A) O último elemento da lista aponta para o primeiro, formando o elo de ligação da lista.

B) Cada elemento possui um atributo do tipo ponteiro da lista, usado para referenciar o próximo elemento.

C) Os elementos da lista devem ser estruturas compostas, como registros.

D) A inclusão de elementos deve ocorrer em uma posição que esteja vazia.

E) O número máximo de elementos a serem incluídos é definido no momento de sua criação.

3) **Analise o seguinte método e marque a alternativa que representa o seu significado:**

```
struct Nodo{
int valor;
struct Nodo *p;
};
typedef struct Nodo node;
void metodo(node *lista, valor) {
node  *n= (node *) malloc(sizeof(node));
n->valor= valor;
n->p= lista->p;
lista->p= n;
```

```
}
```

A) Adiciona um elemento no início da lista.

B) Adiciona um elemento no meio da lista.

C) Adiciona um elemento no final da lista.

D) Remove o primeiro elemento da lista.

E) Remove o último elemento da lista.

4) Analise o seguinte método e selecione a opção que representa o seu objetivo:

```
struct Nodo{
int valor;
struct Nodo *p;
};
typedef struct Nodo node;
void metodo(node *lista) {
node *pr, *at;
at= lista;
while(at->p != NULL){
pr= at->p;
at->p= NULL;
free(at);
at= pr;
}
}
```

A) Remove o primeiro elemento da lista.

B) Remove o último elemento da lista.

C) Substitui o primeiro elemento da lista.

D) Substitui o último elemento da lista.

E) Exclui todos os elementos da lista.

5) Analise o seguinte método e marque a alternativa que representa o seu significado:

```
struct Nodo{

int valor;
struct Nodo *p;
};
typedef struct Nodo node;
int metodo(nodo *lista) {
node *no= lista->p;
return no->valor;
}
```

A) Retorna o primeiro elemento da lista.

B) Retorna o último elemento da lista.

C) Exclui o primeiro elemento da lista e retorna o seu valor anterior.

D) Exclui o último elemento da lista e retorna o seu valor anterior.

E) Retorna todos os elementos da lista.

Na prática

Listas dinâmicas, de um modo geral, são muito utilizadas nos sistemas computacionais. Com uma lista dinâmica é possível, por exemplo, armazenar uma relação de produtos em estoque de determinada loja. No entanto, pode ser necessário que os dados de uma lista sejam inseridos em ordem classificada, visando facilitar as pesquisas.

Veja como:

Conteúdo interativo disponível na plataforma de ensino!

Saiba mais

Para ampliar o seu conhecimento a respeito desse assunto, veja abaixo as sugestões do professor:

Aula 11 - ED - Lista dinâmica encadeada parte 2 - Implementação.

Assista ao vídeo a seguir para conhecer mais sobre a implementação de uma lista dinâmica.

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

Aula 14 - ED1 - Remoção na lista dinâmica.

Assista ao vídeo a seguir para conhecer mais sobre exclusão em uma lista dinâmica.

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

Aula 13 - ED1 - Inserção na lista dinâmica.

Assista ao vídeo a seguir para saber mais sobre a inserção em uma lista dinâmica.

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.