Aula 05 - Estrutura de dados 1 – Uniube

Prof. Marcos Lopes

34 9 9878 0925

Busca em uma lista encadeada

Veja como é fácil verificar se um objeto x pertence a uma lista encadeada, ou seja, se é igual ao conteúdo de alguma célula da lista:

```
// Esta função recebe um inteiro x e uma lista encadeada le de inteiros e devolve
// o endereço de uma celula que contém x. Se tal celula não existe, devolve NULL.

celula* busca (int x, celula *le) {
   celula *p;
   p = le;
   while (p != NULL && p->conteudo != x) {
      p = p->prox;
   }
   return p;
}
```

Que beleza! Nada de variáveis booleanas de sinalização!

Além disso, a função tem comportamento correto mesmo que a lista esteja vazia.

Exercícios 1

a) Teste a função busca do slide anterior

b) Contrua uma função ‘busca_r’ que funciona igual a função busca, porem implementada usando recursividade!

c) Escreva uma função que verifique se uma lista encadeada que contém números inteiros está em ordem crescente.

Cabeça de lista

Às vezes convém tratar a primeira célula de uma lista encadeada como um mero marcador de início e ignorar o conteúdo da célula. Nesse caso, dizemos que a primeira célula é a cabeça (head cell) da lista encadeada.

Uma lista encadeada le com cabeça está vazia se e somente se le->prox == NULL. Para criar uma lista encadeada vazia com cabeça, basta dizer

```
celula *le;
le = malloc (sizeof (celula));
le->prox = NULL;
```

Para imprimir o conteúdo de uma lista encadeada le com cabeça, faça

```
void imprima (celula *le) {
  celula *p;
  for (p = le->prox; p != NULL; p = p->prox)
    printf ("%d\n", p->conteudo);
}
```

Exercícios 2

a) Escreva versões das funções busca e busca_r para listas encadeadas com cabeça.

b) Escreva uma função que verifique se uma lista encadeada com cabeça está em ordem crescente. (Suponha que as células contêm números inteiros.)

Inserção em uma lista encadeada

Considere o problema de inserir uma nova célula em uma lista encadeada. Suponha que quero inserir a nova célula entre a posição apontada por p e a posição seguinte. (É claro que isso só faz sentido se p é diferente de NULL.)

```
// Esta função insere uma nova celula em uma lista encadeada com cabeça.
// A nova celula tem conteudo x e é inserida entre a celula p e a celula seguinte.
// É necessário alocar memória pra nova célula
// (Supõe-se que p != NULL.)

void insere (int x, celula *p){
   celula *nova;
   nova = (celula*) malloc (sizeof (celula));
   nova->conteudo = x;
   nova->prox = p->prox;
   p->prox = nova;
}
```

Inserção em uma lista encadeada

Simples e rápido!

Basta mudar os valores de alguns ponteiros. Observe que a função comporta-se corretamente mesmo quando quero inserir no fim da lista, isto é, quando p->prox == NULL.

Se a lista tem cabeça, a função pode ser usada para inserir no início da lista: basta que p aponte para a célula-cabeça.

Mas no caso de lista sem cabeça a função não é capaz de inserir antes da primeira célula.

Exercícios 3

a) Desenhe um diagrama esquemático de um instantâneo das células de uma lista encadeada ‘lista’ (com cabeça), após todos os comandos:

```
celula *lista;
lista = (celula*) malloc(sizeof(celula));
lista->prox = NULL;

insere(10, lista);
insere(20, lista);
insere(30, lista);
```

b) Escreva uma função que insira uma nova célula em uma lista encadeada **sem cabeça.** (Será preciso tomar algumas decisões de projeto antes de começar a programar.)

c) Desenhe um diagrama esquemático similar ao da letra (a) considerando inserções em uma lista sem a célula da cabeça.