Aula 04 - Estrutura de dados 1 – Uniube

Prof. Marcos Lopes

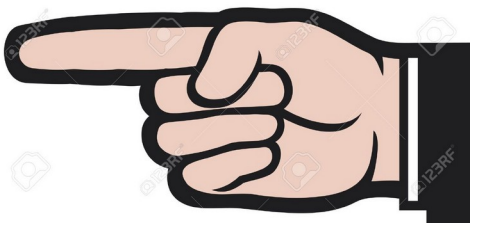
34 9 9878 0925

Listas encadeadas:

Uma lista encadeada é uma representação de uma sequência de objetos, todos do mesmo tipo, na memória RAM (= random access memory) do computador.

Cada elemento da sequência é armazenado em uma célula da lista: o primeiro elemento na primeira célula, o segundo na segunda, e assim por diante.

Sumário:

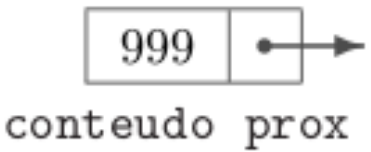


Estrutura de uma lista encadeada:

Uma lista encadeada (= linked list = lista ligada) é uma sequência de células; cada célula contém um objeto (todos os objetos são do mesmo tipo) e o endereço da célula seguinte.

Vamos supor que os objetos armazenados nas células são do tipo int. Cada célula é um registro que pode ser definido assim:

```
struct reg {
      int  conteudo;
      struct reg *prox;
};
```



É conveniente tratar as células como um novo tipo-de-dados e atribuir um nome a esse novo tipo:

```
typedef struct reg celula;  // célula
```

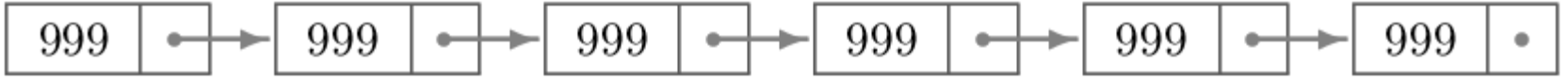
Uma célula c e um ponteiro p para uma célula podem ser declarados assim:

```
celula  c;
celula *p;
```

Se c é uma célula então c.conteudo é o conteúdo, ou carga útil, da célula e c.prox é o endereço da próxima célula.

Se p é o endereço de uma célula, então p->conteudo é o conteúdo da célula e p->prox é o endereço da próxima célula.

Se p é o endereço da última célula da lista então p->prox vale NULL .



Exercícios 1

a) Verifique que a declaração de celula pode também ser escrita assim:

```
typedef struct celula celula;
struct celula {
   int     conteudo;
   celula *prox;
};
```

b) Compile e execute o seguinte programa:

```
int main (void) {

   celula cel;
   printf ("sizeof (cel) = %d\n", sizeof (cel));
   return EXIT_SUCCESS;
}
```

Endereço de uma lista encadeada

O endereço de uma lista encadeada é o endereço de sua primeira célula. Se ‘le’ é o endereço de uma lista encadeada, convém dizer simplesmente que

le é uma lista encadeada.

A lista está vazia (ou seja, não tem célula alguma) se e somente se le == NULL.

Listas são elementos eminentemente recursivos.

Para tornar isso evidente, basta fazer a seguinte observação: se le é uma lista não vazia então le->prox também é uma lista.

Muitos algoritmos sobre listas encadeadas ficam mais simples quando escritos em estilo recursivo.

Exemplo. A seguinte função recursiva imprime o conteúdo de uma lista encadeada le:

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

E aqui está a versão iterativa da mesma função:

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

Exercícios 2

a) Escreva um programa pra testar a função ‘imprime’ do slide anterior. Adicione algumas celulas na lista, declarando cada celula.

b) Escreva uma função que conte o número de células de uma lista encadeada. Faça duas versões: uma iterativa e uma recursiva.