

Listas: nós de cabeçalho, listas não homogêneas, listas generalizadas

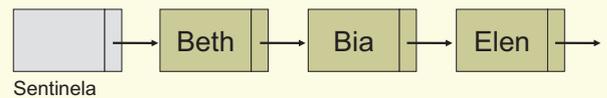
Baseado no material de Thiago A. S. Pardo

Algoritmos e Estruturas de Dados I

Debora Medeiros

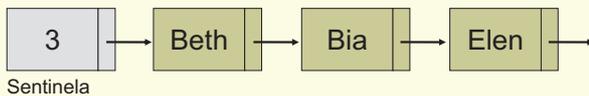
Lista com nó de cabeçalho

- Nó de cabeçalho
 - *Header*, sentinela, etc.
- Para que?



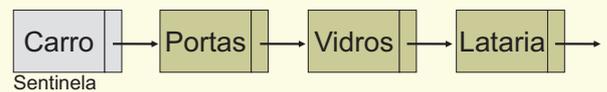
Lista com nó de cabeçalho

- Possibilidades de uso
 - **Informação global** sobre a lista que possa ser necessária na aplicação
 - Armazenar número de elementos da lista, para que não seja necessário atravessá-la contando seus elementos



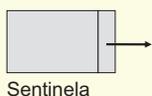
Lista com nó de cabeçalho

- Possibilidades de uso
 - **Informação global** sobre a lista que possa ser necessária na aplicação
 - Em uma fábrica, guarda-se as peças que compõem cada equipamento produzido, sendo este indicado pelo nó sentinela
 - Informações do voo correspondente a uma fila de passageiros



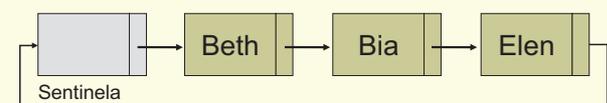
Lista com nó de cabeçalho

- Possibilidades de uso
 - **Informação global** sobre a lista que possa ser necessária na aplicação
 - Lista vazia contém somente o nó sentinela



Lista com nó de cabeçalho

- Possibilidades de uso
 - **Lista circular**
 - Não existe mais NULL no fim da lista, eliminando-se o risco de acessar uma posição inválida de memória

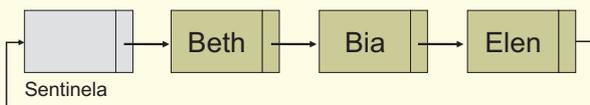


Lista com nó de cabeçalho

■ Possibilidades de uso

■ Lista circular

- Como saber qual é o último elemento da lista?

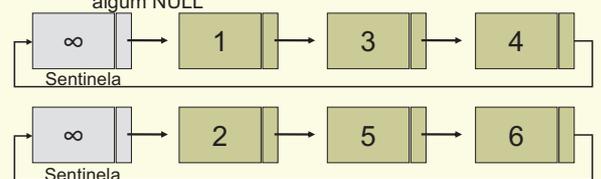


Lista com nó de cabeçalho

■ Possibilidades de uso

■ Auxílio para algumas tarefas

- Unir duas listas ordenadas pode ser mais simples
 - Coloca-se "infinito" nos sentinelas (no começo, caso a lista seja circular ou no fim da lista, caso contrário), forçando, assim, as listas a serem consumidas inteiramente, sem ter que se preocupar se se chegou a algum NULL



Lista com nó de cabeçalho

■ Possibilidades de uso

■ Informação global sobre a lista que possa ser necessária na aplicação

- Operação de busca de informação pode ser simplificada
 - No caso de lista circular, o elemento buscado pode ser colocado no nó de cabeçalho, assim, ele sempre será encontrado, evitando-se de ter que lidar com o NULL

Lista com nó de cabeçalho

■ Possibilidades de uso

■ Lista circular

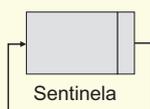
- Como representar a lista vazia?

Lista com nó de cabeçalho

■ Possibilidades de uso

■ Lista circular

- Como representar a lista vazia?

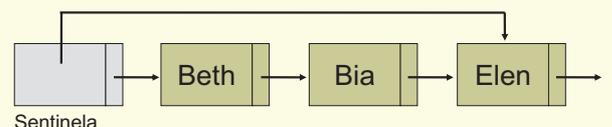


Lista com nó de cabeçalho

■ Possibilidades de uso

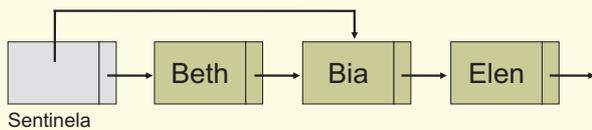
■ Informações para uso da lista como pilha, fila, etc.

- Exemplo: em vez de um ponteiro de fim da fila, o nó sentinela pode apontar o fim
 - O campo info do nó sentinela passa a ser um ponteiro
 - Acaba por indicar o início da fila também



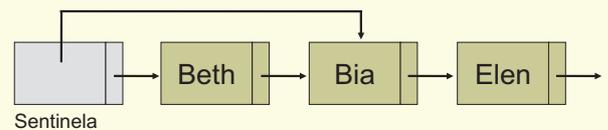
Lista com nó de cabeçalho

- Possibilidades de uso
 - Indica um **nó específico da lista**
 - Por exemplo, em buscas que são constantemente interrompidas
 - Verificação de pessoas em ordem alfabética: poupa o esforço de se recomençar ou a necessidade de ter uma variável auxiliar



Lista com nó de cabeçalho

- Possibilidades de uso
 - Nó sentinela com **ponteiro em seu campo info**
 - Vantagem: acesso possivelmente mais direto e imediato
 - Desvantagens? Quais?

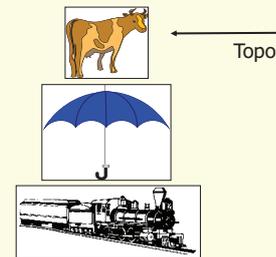


Lista não homogênea

- Lista **"genérica"**
- Possibilidade de usar uma mesma estrutura para armazenar informações diferentes
 - Inteiro, caracter, estrutura, etc.
- Não é necessário definir blocos de memória diferentes

Lista não homogênea

- Como inserir uma vaca, um guarda-chuva e um trem em uma mesma pilha?



Lista não homogênea

- **Solução 1**
 - Definem-se vários campos de informação
 - Usam-se somente os necessários

```
struct no {  
    char info1;  
    int info2;  
    struct no *prox;  
}
```

- Desvantagem: memória alocada desnecessariamente

Lista não homogênea

- **Solução 2**
 - Definem-se vários ponteiros
 - Aloca-se memória conforme necessidade

```
struct no {  
    char *info1;  
    int *info2;  
    struct no *prox;  
}
```

Lista não homogênea

Solução 3

- Definem-se um ponteiro genérico para qualquer tipo

```
struct no {
    void *info;
    struct no *prox;
}
```

Lista não homogênea

Solução 4

- Usa-se um registro/estrutura variante

```
struct no {
    union {
        int ival;
        float fval;
        char cval;
    } elemento;
    int tipo;
    struct no *prox;
}
```

Lista generalizada

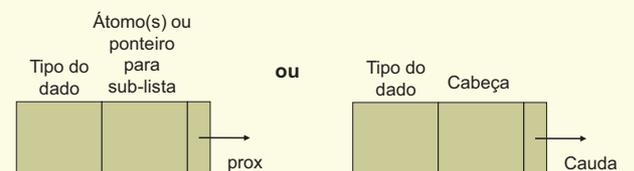
- Uma lista generalizada é aquela que pode ter como elemento ou um átomo ou uma outra lista (sub-lista)
 - Átomo: integer, real, char, string, etc.
- Cabeça e cauda**
 - Cabeça: primeiro elemento da lista (átomo ou lista)
 - Cauda: o resto (uma outra lista, mesmo que vazia)

Lista generalizada

Definição formal

- Uma lista generalizada A é uma seqüência finita de $n \geq 0$ elementos $\alpha_1, \alpha_2, \dots, \alpha_n$, em que α_i são átomos ou listas. Os elementos α_i , com $0 \leq i < n$, que não são átomos são chamados sub-listas de A.

Estrutura básica do bloco de memória

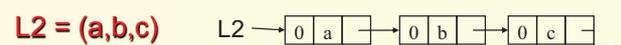
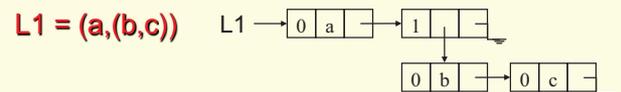


Lista generalizada

- Suponha que uma lista seja representada por elementos entre parênteses (no estilo da linguagem de programação LISP) ou entre colchetes (no estilo de PROLOG)
 - (a,b,c) ou [a,b,c]
 - (a,(b,c)) ou [a,[b,c]]
 - (a,(b),(c)) ou [a,[b],[c]]
 - (a,b,()) ou [a,b,[]]
- Tipo=0 indica átomo e tipo=1 indica sub-lista

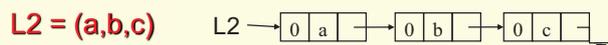
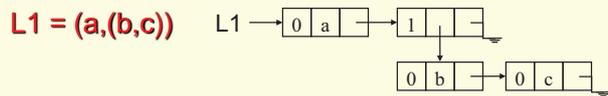
Lista generalizada

Exemplos de representação



Lista generalizada

Exemplos de representação



Cabeça(L2)? Cauda(L2)? Cabeça(Cauda(L2))?

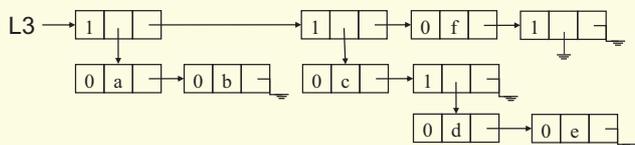
Cabeça(L1)? Cauda(L1)? Cabeça(Cauda(L1))?

Lista generalizada

Exercício: faça a representação da lista L3 ((a,b),(c,(d,e)),f,())

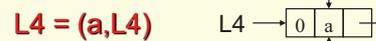
Lista generalizada

Exercício: faça a representação da lista L3 ((a,b),(c,(d,e)),f,())

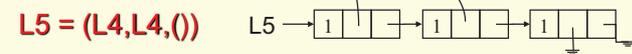


Lista generalizada

Listas Recursivas



Listas Compartilhadas



Lista generalizada

Declaração em C

Union

Lista generalizada

Declaração em C

Union

```
typedef struct bloco {
    union {
        char atomo;
        struct bloco *sublista;
    } info;
    int tipo;
    struct bloco *prox;
} no;
```

Lista generalizada e polinômios

- Considere os polinômios:

$$P1 = 4x^2y^3z + 3xy + 5$$

$$P2 = x^{10}y^3z^2 + 2x^8y^2z^2 + x^4y^4z + 6x^3y^4z + 2yz$$

$$P3 = 3x^2y$$

- (a) n° de termos: variável

- P1=3, P2=5, P3=1

- (a) n° de variáveis: variável

- P1=P2=3, P3=2

- (a) nem todo termo é expresso com todas as variáveis

Lista generalizada e polinômios

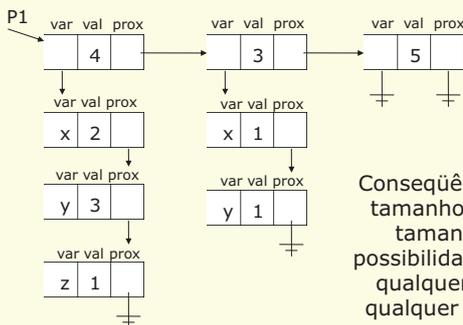
- Objetivos

- representar de forma a otimizar o uso de memória
- representação única para todo polinômio

- Solução: lista generalizada

Lista generalizada e polinômios

Ex: $P1 = 4x^2y^3z + 3xy + 5$



Conseqüência: registros de tamanhos fixos; listas de tamanhos variáveis; possibilidade de representar qualquer polinômio com qualquer n° de variáveis e qualquer grau

Lista generalizada

- Exercício

- Implementar uma sub-rotina para verificar se um átomo x está em uma lista generalizada
 - Apenas na lista principal (primeiro nível da lista)

Lista generalizada

- Exercício

- Implementar uma sub-rotina para verificar se um átomo x está em uma lista generalizada
 - Em qualquer parte dela

Lista generalizada

- Exercício

- Implementar uma sub-rotina para verificar se duas listas generalizadas são completamente iguais

Lista generalizada

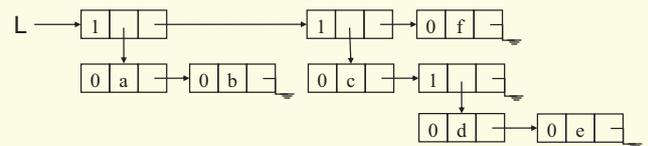
Exercício

- Implementar uma sub-rotina para verificar se duas listas generalizadas são estruturalmente iguais
 - O conteúdo em si não importa

Lista generalizada

Exercício

- Implementar uma sub-rotina que determina a profundidade máxima de uma lista generalizada
 - $A=(a,(b)) \rightarrow \text{prof}(A)=2$
 - $B=(a,b,c) \rightarrow \text{prof}(B)=1$
 - $C=() \rightarrow \text{prof}(C)=0$
 - Por exemplo, para o caso abaixo, a sub-rotina deveria retornar profundidade 3



Exercício

■ Implementar uma sub-rotina que:

- (a) receba um polinômio representado via lista generalizada e os valores das variáveis
- (b) percorra a lista generalizada e compute o resultado do polinômio
- (c) retorne o resultado para quem chamou a sub-rotina