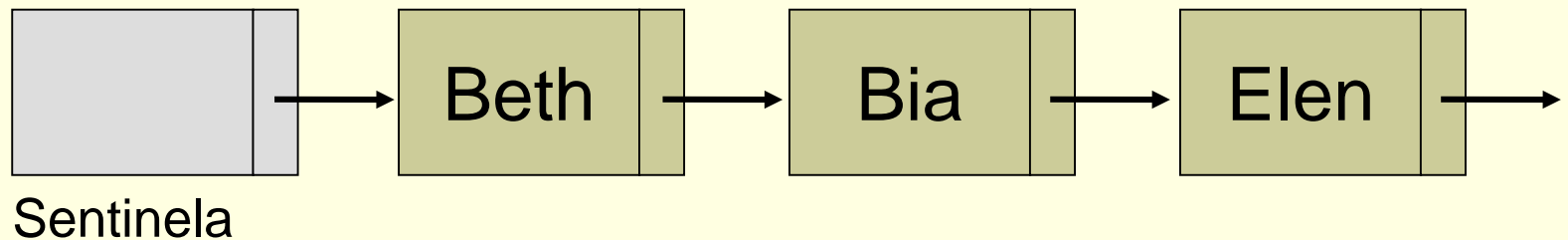


Listas: nós de cabeçalho, listas não homogêneas, listas generalizadas

SCC-202 – Algoritmos e Estruturas de
Dados I

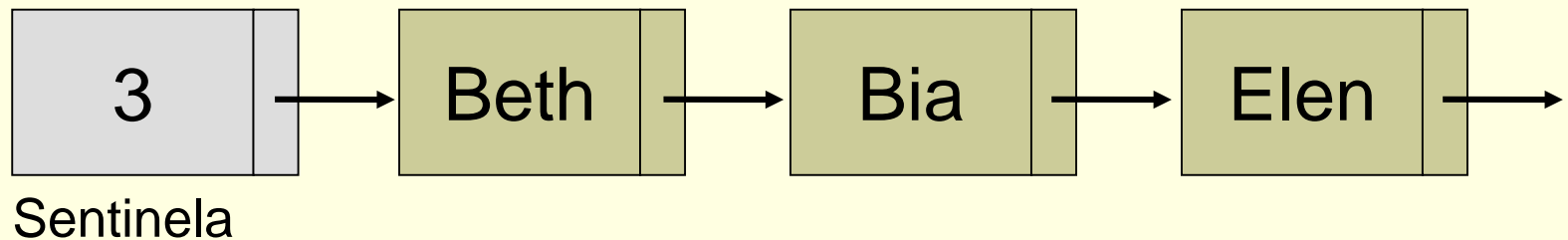
Lista com nó de cabeçalho

- Nó de cabeçalho
 - *Header*, sentinela, etc.
 - Pode estar em qualquer ponto da lista
- Para que?



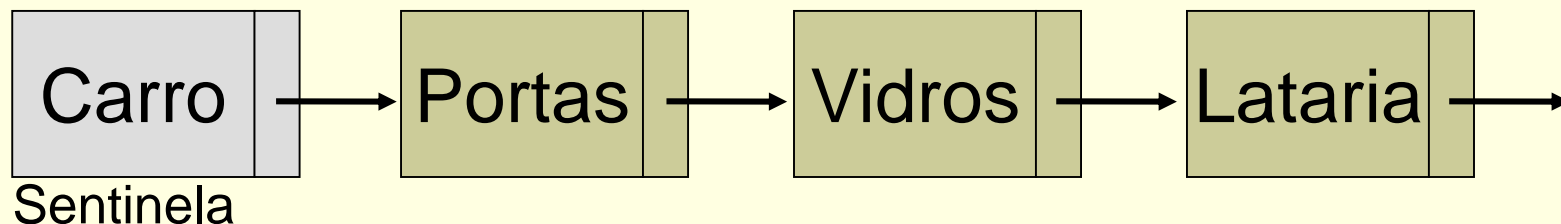
Lista com nó de cabeçalho

- 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



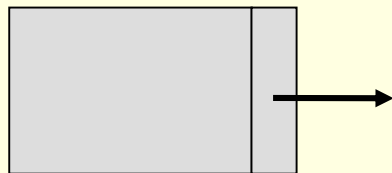
Lista com nó de cabeçalho

- Possibilidades de uso
 - **Informação global** sobre a lista que possa ser necessária na aplicação
 - Em uma fábrica, guardam-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

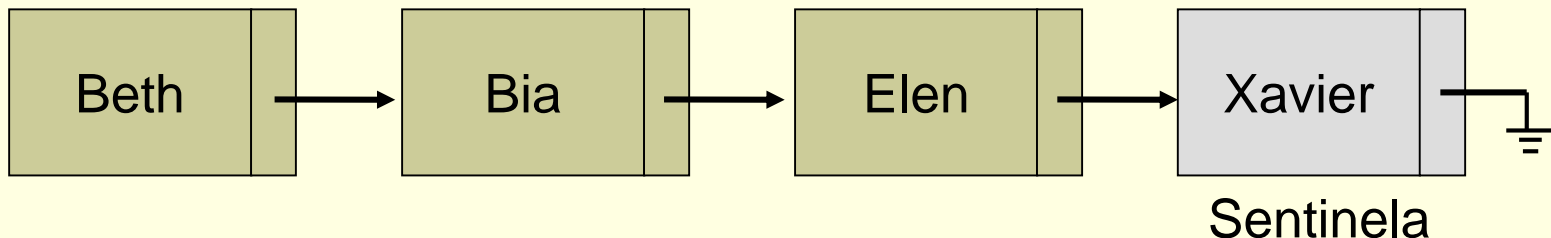


Sentinela

Lista com nó de cabeçalho

- Possibilidades de uso
 - **Auxílio para algumas tarefas**
 - Operação de busca de informação pode ser simplificada
 - Qual a vantagem?

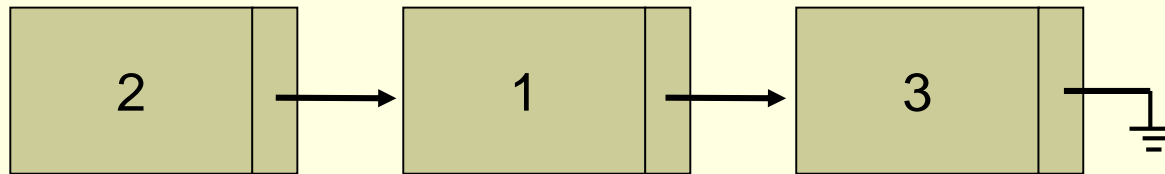
Busca por Xavier



Lista com nó de cabeçalho

■ Sem nó sentinela

Busca por 5

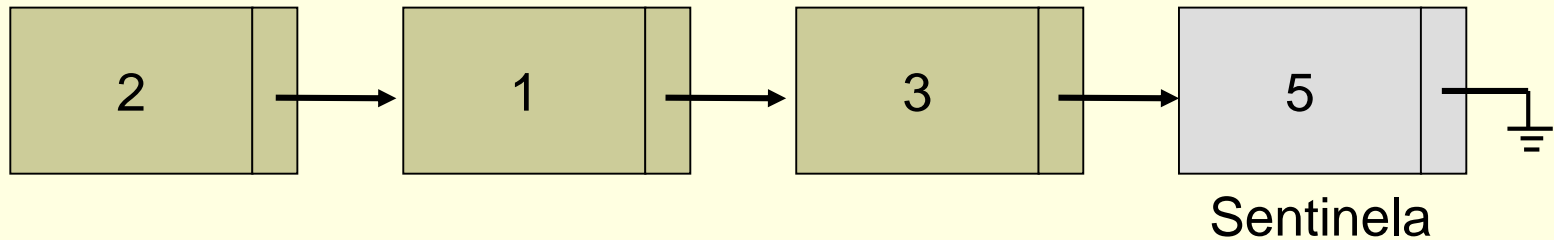


```
int busca(Lista *L, elem *x) {  
    int achou=0;  
    no *p=Lista->ini;  
    while ((p!=NULL) && (!achou)) {  
        if (p->info==*x)  
            achou=1;  
        else p=p->prox;  
    }  
    return(achou);  
}
```

Lista com nó de cabeçalho

■ Com nó sentinela

Busca por 5



```
int busca(Lista *L, elem *x) {  
    int achou=0;  
    no *p=Lista->ini;  
    while ((p!=NULL) && (!achou)) { //poupamos uma comparação com NULL  
        if (p->info==*x)  
            achou=1;  
        else p=p->prox;  
    }  
    if (p->prox==NULL) //se encontramos o nó sentinela...  
        return(0);  
    else return(1);  
}
```

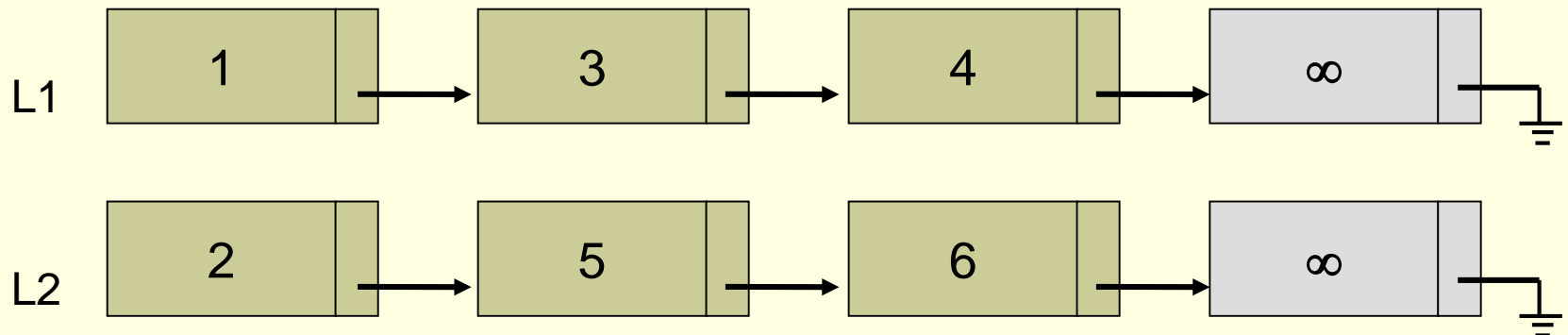

Lista com nó de cabeçalho

- Possibilidades de uso

- Auxílio para algumas tarefas

- Unir duas listas ordenadas pode ser mais simples
 - Coloca-se “infinito” nos sentinelas, forçando, assim, as listas a serem consumidas inteiramente, sem ter que se preocupar se se chegou a algum NULL

Como ocorre a união das duas listas abaixo?



Lista com nó de cabeçalho

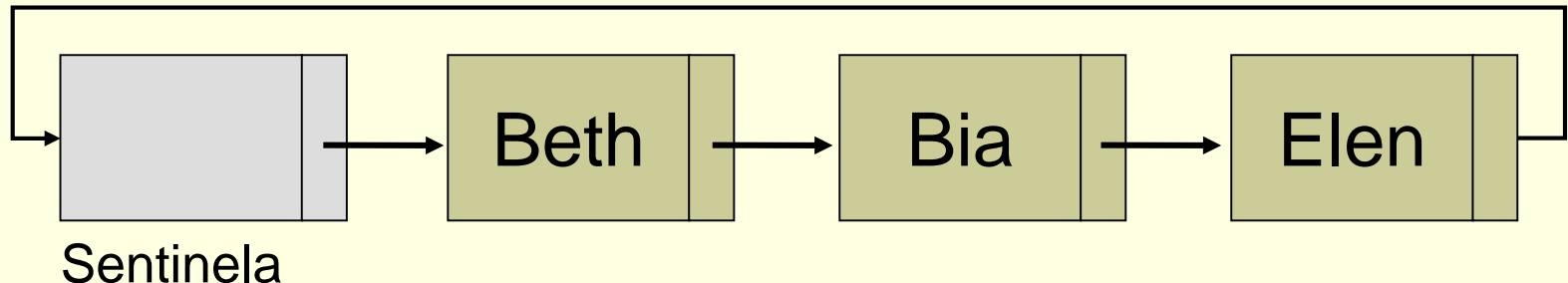
- Possibilidades de uso
 - **Auxílio para algumas tarefas**
 - Operações de inserção e remoção podem se tornar mais caras
 - Por que?

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

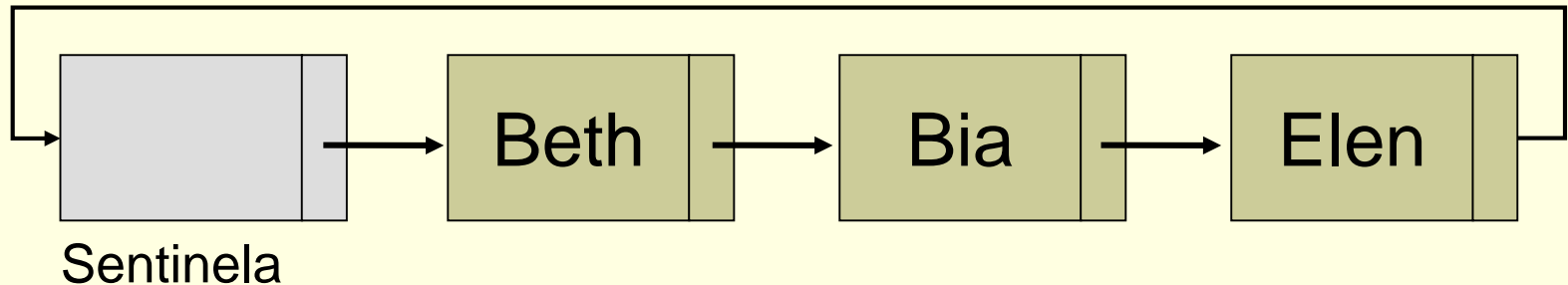


Lista com nó de cabeçalho

- Possibilidades de uso

- Lista circular

- Como saber qual é o último elemento da lista?

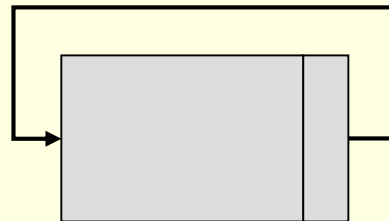


Lista com nó de cabeçalho

- Possibilidades de uso
 - Lista circular
 - Como representar a lista vazia?

Lista com nó de cabeçalho

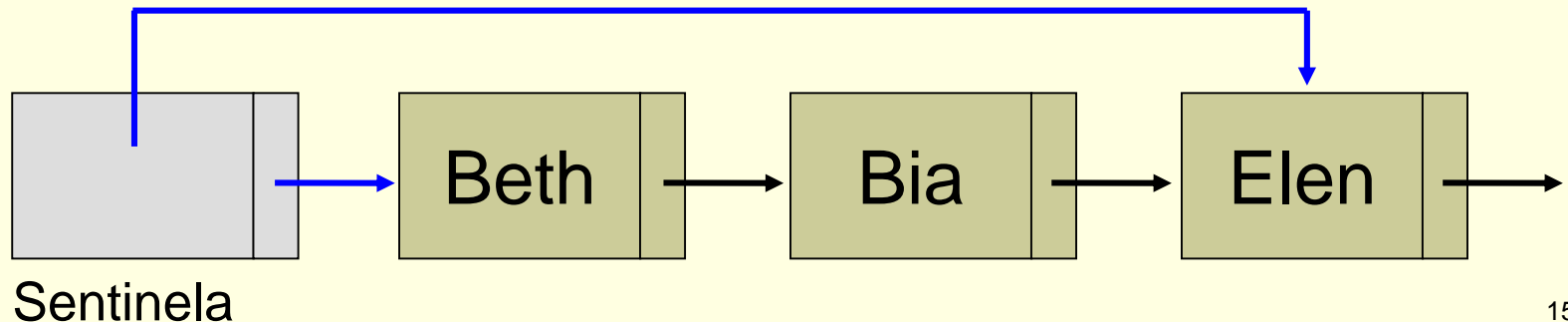
- Possibilidades de uso
 - **Lista circular**
 - Como representar a lista vazia?



Sentinela

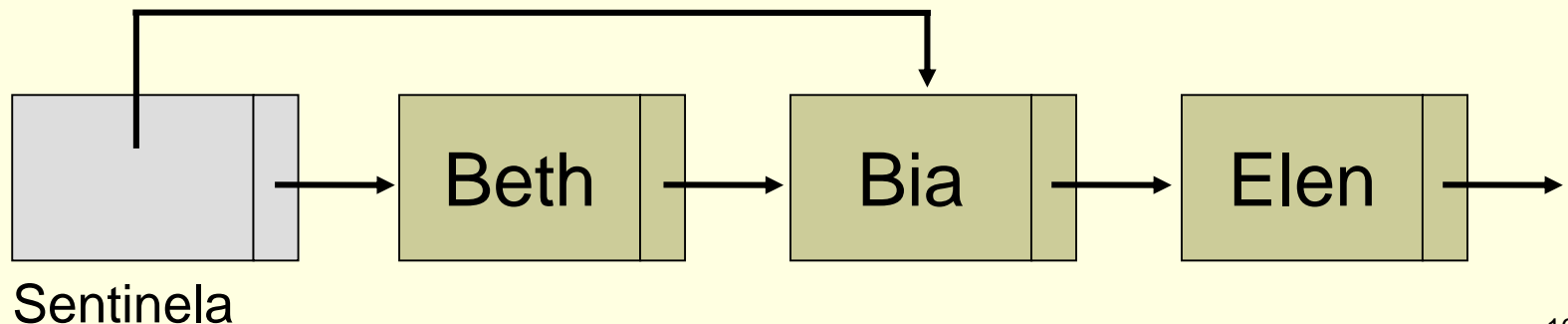
Lista com nó de cabeçalho

- 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



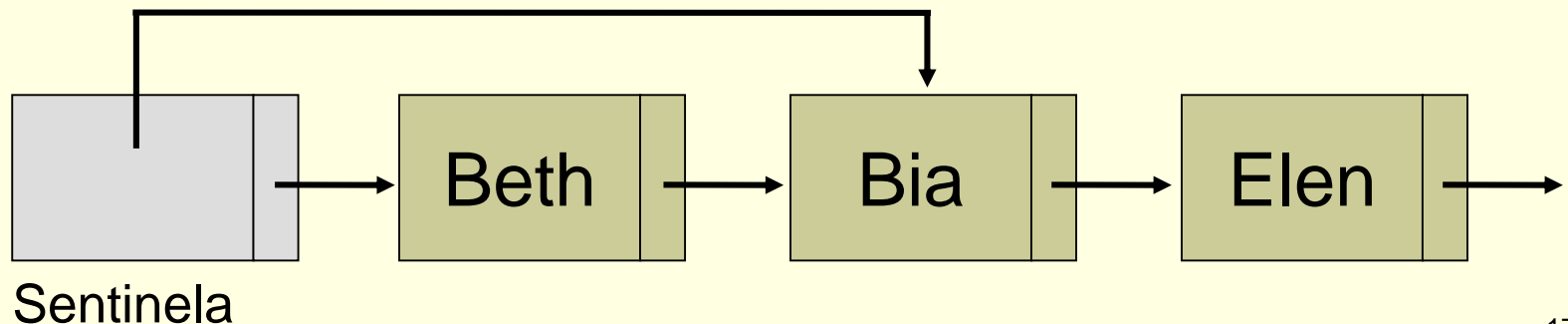
Lista com nó de cabeçalho

- 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 recomeçar ou a necessidade de ter uma variável auxiliar



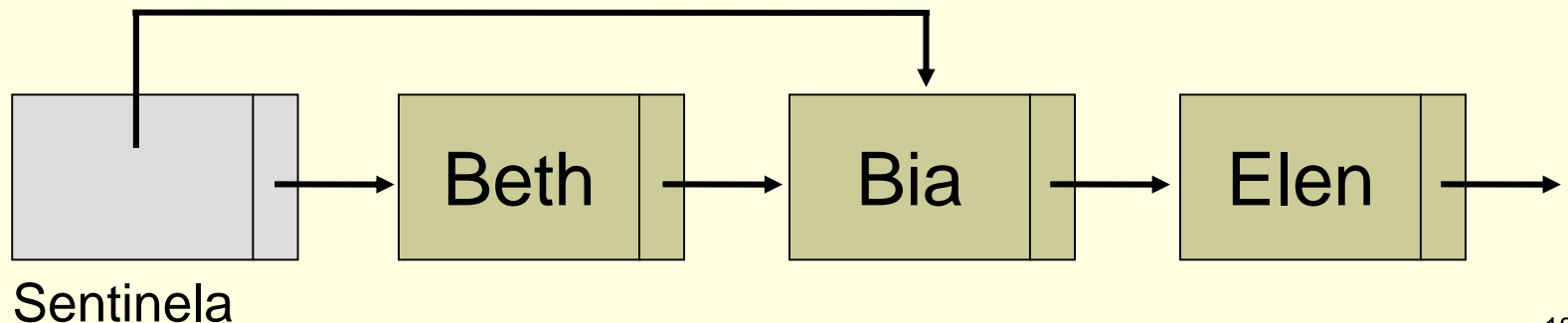
Lista com nó de cabeçalho

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



Lista com nó de cabeçalho

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



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

■ 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

- Define-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_usado;  
    struct no *prox;  
}
```

Lista generalizada

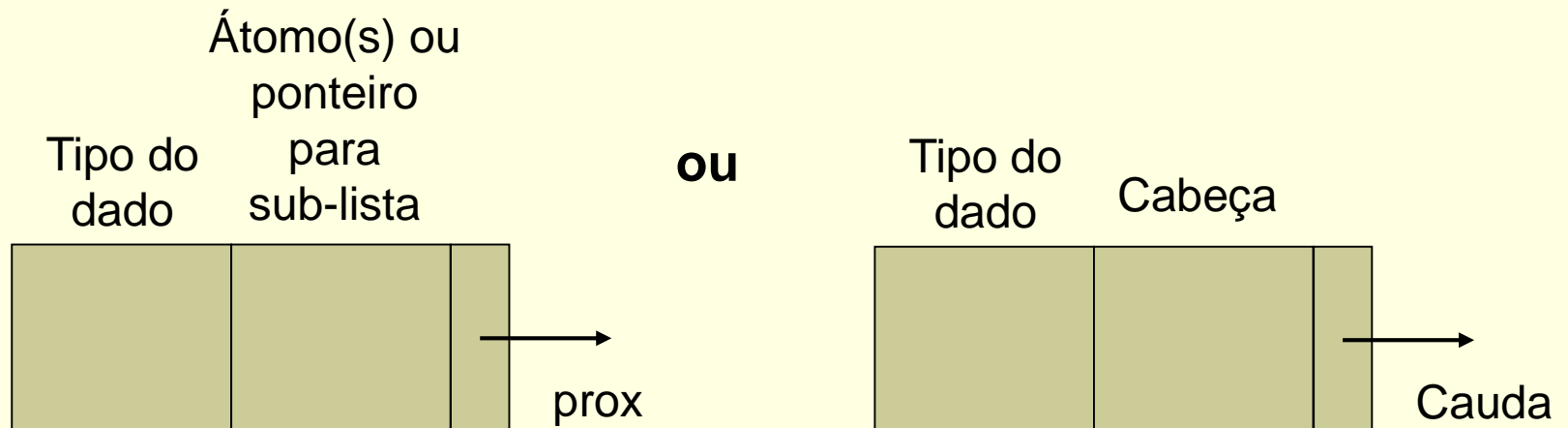
- Uma **lista generalizada** é aquela que pode ter como elemento ou um átomo ou uma outra lista (sub-lista)
 - Átomo: integer, real, char, string, etc.
- **Cabeça e cauda**
 - Cabeça: primeiro elemento da lista (átomo ou lista)
 - Cauda: o resto (uma outra lista, mesmo que vazia)

Lista generalizada

■ Definição formal

- Uma lista generalizada A é uma sequência finita de $n \geq 0$ elementos $\alpha_0, \alpha_1, \dots, \alpha_n$, em que α_i são átomos ou listas. Os elementos α_i , com $0 \leq i \leq n$, que não são átomos são chamados sublistas de A .

■ Estrutura básica do bloco de memória



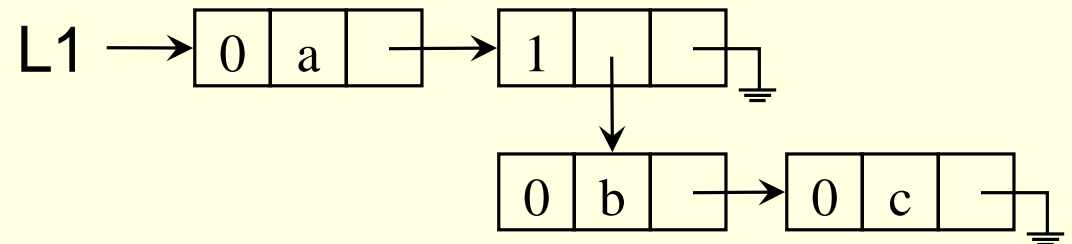
Lista generalizada

- 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

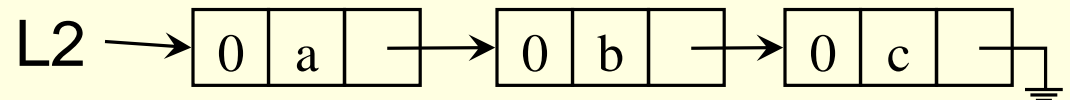
Lista generalizada

■ Exemplos de representação

L1 = (a,(b,c))



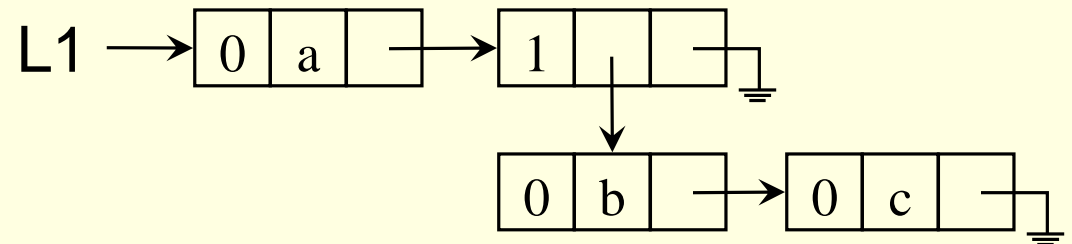
L2 = (a,b,c)



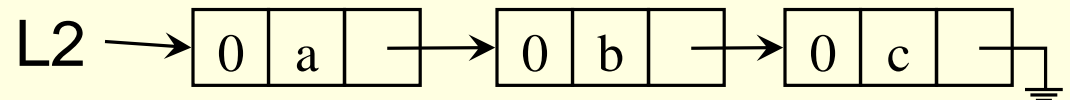
Lista generalizada

■ Exemplos de representação

L1 = (a,(b,c))



L2 = (a,b,c)



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

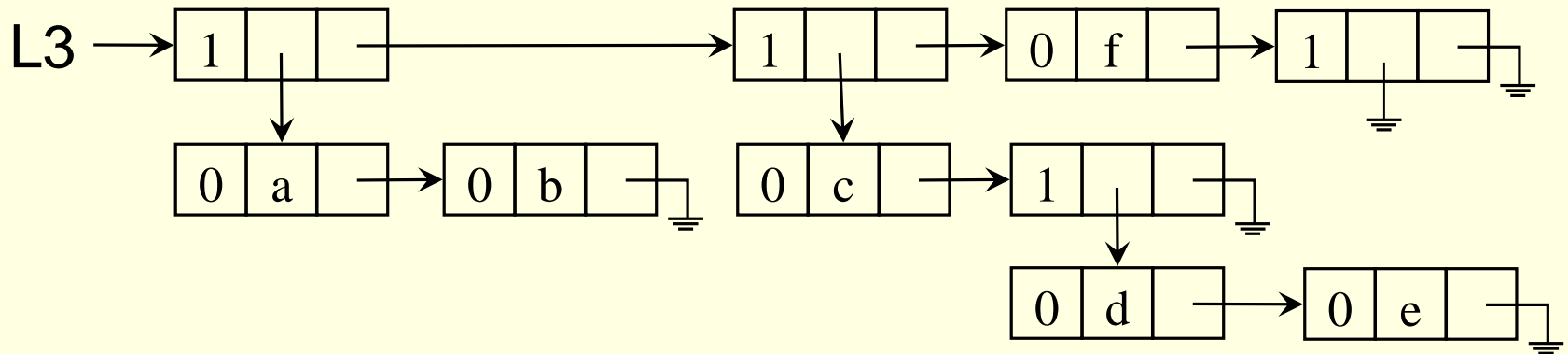
Cabeça(L1)? Cauda(L1)? Cabeça(Cauda(L1))?

Lista generalizada

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

Lista generalizada

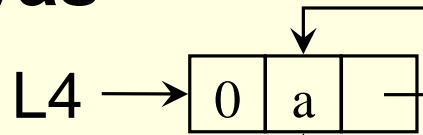
- Exercício: faça a representação da lista L3
((a,b),(c,(d,e)),f,())



Lista generalizada

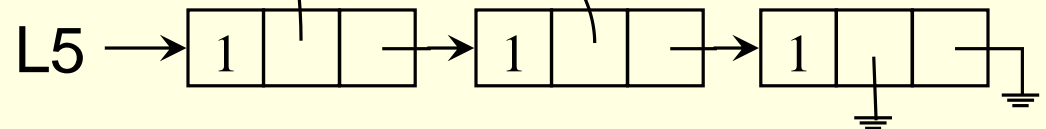
Listas Recursivas

$L4 = (a, L4)$



Listas Compartilhadas

$L5 = (L4, L4, ())$



Lista generalizada

- Declaração em C
 - Union

Lista generalizada

- Declaração em C

- Union

```
typedef char elem;
```

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

```
typedef struct {  
    no *inicio;  
} ListaGen;
```

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 um átomo x está em uma lista generalizada
 - Em qualquer parte dela
 - Faça a árvore de ativação dessa sub-rotina para a busca pelo elemento 'd' na lista generalizada (f,(a,b),(c,(d,e)),())

Lista generalizada

■ Exercício

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

Lista generalizada

■ Exercício

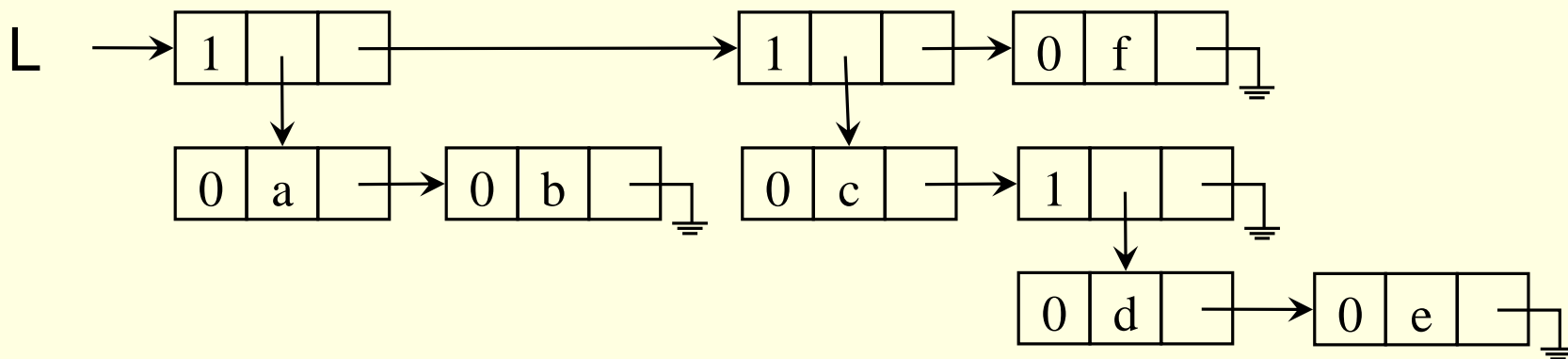
- Implementar uma sub-rotina para verificar se duas listas generalizadas são estruturalmente iguais
 - O conteúdo em si não importa

Exercício para entregar

■ 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



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

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

(b) n° de variáveis: variável

- $P1=P2=3, P3=2$

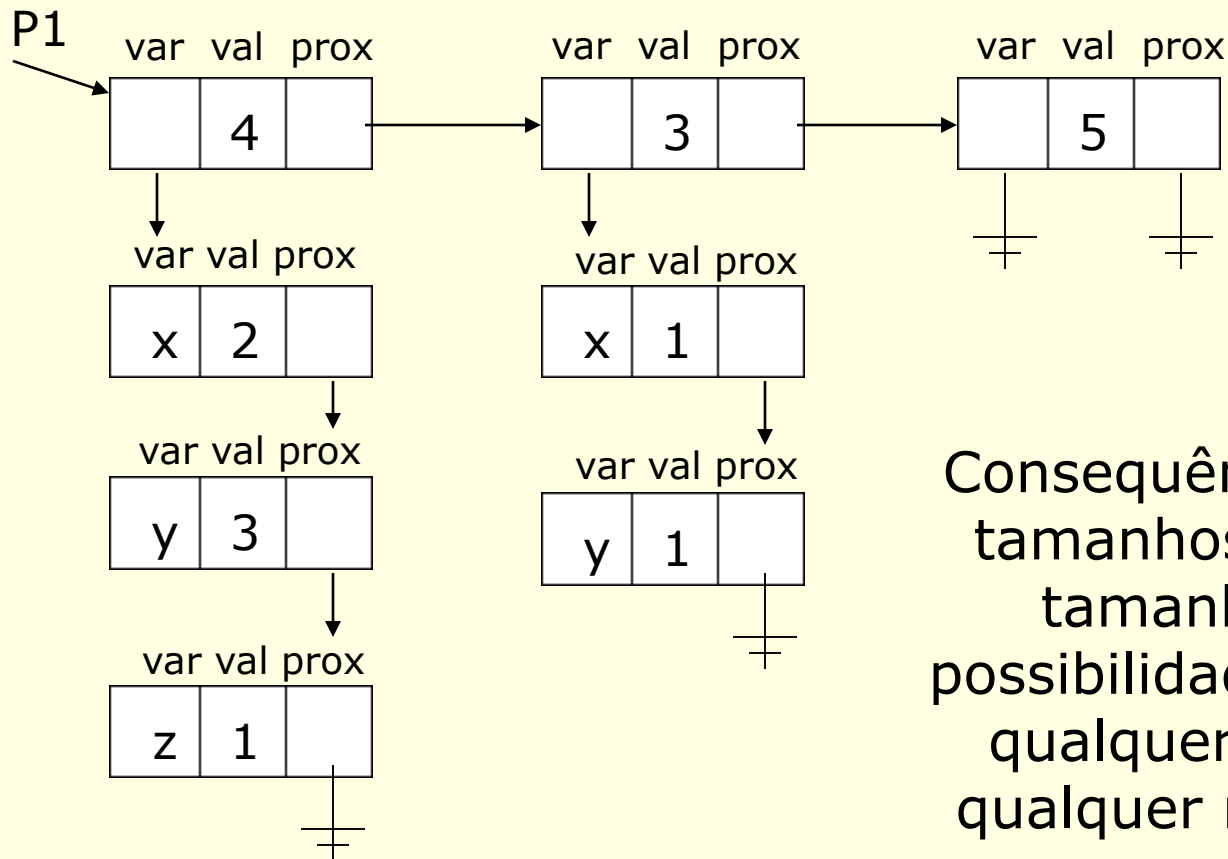
(c) nem todo termo é expresso com todas as variáveis

Lista generalizada e polinômios

- Objetivos
 - representar de forma organizada e robusta
 - representação única para todo polinômio
- Solução: lista generalizada

Lista generalizada e polinômios

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



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

Exercício para casa

- 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