Listas generalizadas

SCC-202 – Algoritmos e Estruturas de Dados I

Lista generalizada

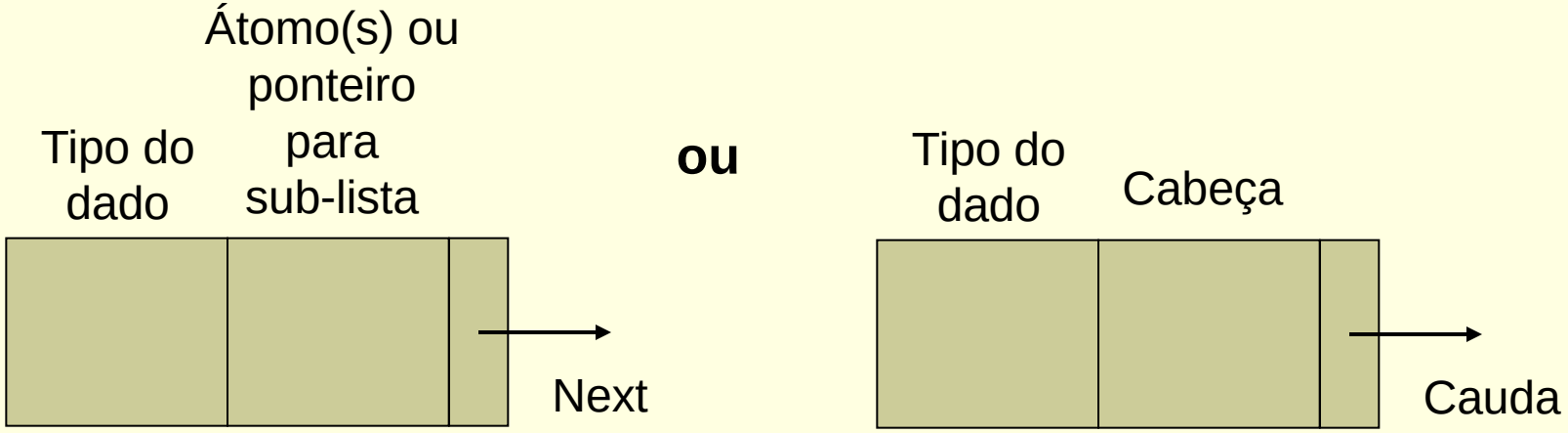
- Uma lista generalizada é aquela que pode ter como elemento ou um átomo ou uma outra lista (sub-lista)
  - Átomo: integer, real, char, string, etc.

- 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 α_1, α_1, ... α_n, em que α_i são átomos ou listas generalizadas. Os elementos α_i, com $1 \leq i \leq n$, que não são átomos são chamados sub-listas de A.

- Estrutura básica do bloco de memória



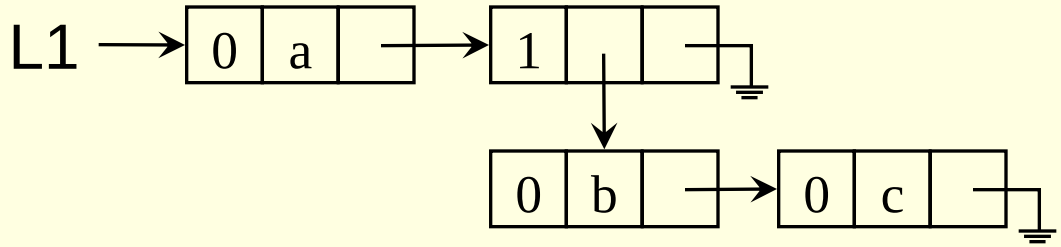
Lista generalizada

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

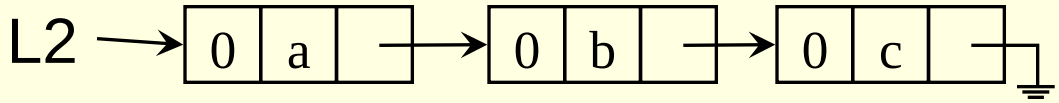
Lista generalizada

- Exemplos de representação

L1 = (a,(b,c))



L2 = (a,b,c)

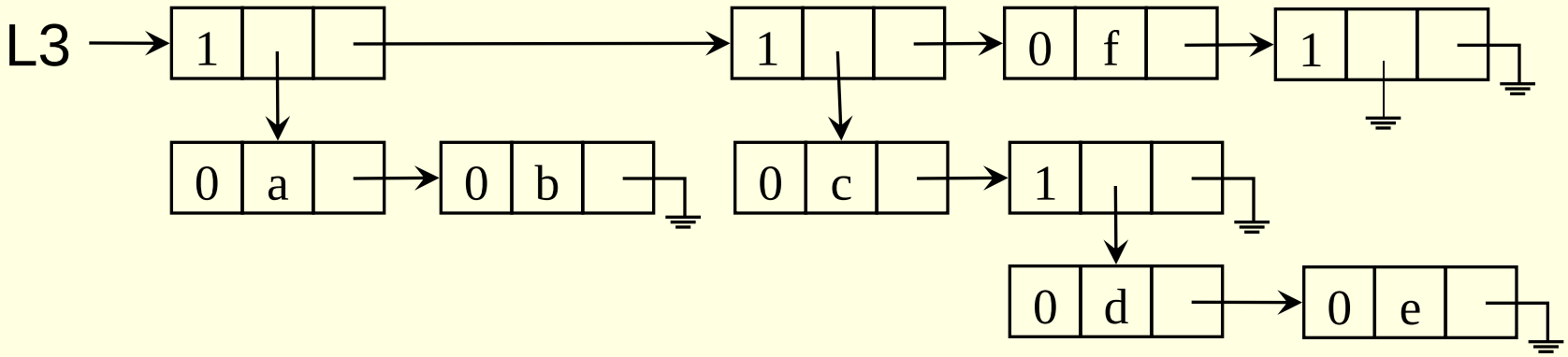


Cabeça(L2)? Cauda(L2)? Cabeça(Cauda(L2))?

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

Lista generalizada

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



Lista generalizada

- Declaração
  - Union

```
struct no {
        int tipo;
        union {
                int atomo;
                struct no *lista;
        } car;
        struct no *cdr;
};
typedef struct no Rec;
Rec *Lista;
```

Lista generalizada

- Exercícios
  - Implementar uma função recursiva para buscar um átomo x numa lista generalizada
    - (1) considere apenas a lista principal;
    - (2) considere que x pode estar em qualquer sublista.
  - Implementar uma sub-rotina para verificar se duas listas generalizadas são iguais
    - Tente fazer a sub-rotina recursiva

Algoritmos

- Uma função booleana recursiva para buscar um átomo x numa lista generalizada, L. Retorna também o endereço, se estiver lá.
  - (1) considere apenas a lista principal;

```
Função Busca (x, L):
Se L é vazia então retorna FALSE
Senão L = (l1, l2, ....ln) e
      se l1 é atomo então
            se l1 = x então retorna TRUE e x
      retorna Busca (x, (l2, l3,....ln))
```

Algoritmos

- Uma função booleana recursiva para buscar um átomo x numa lista generalizada, L. Retorna também o endereço, se estiver lá.
  - (2) considere que x pode estar em qualquer sublista.

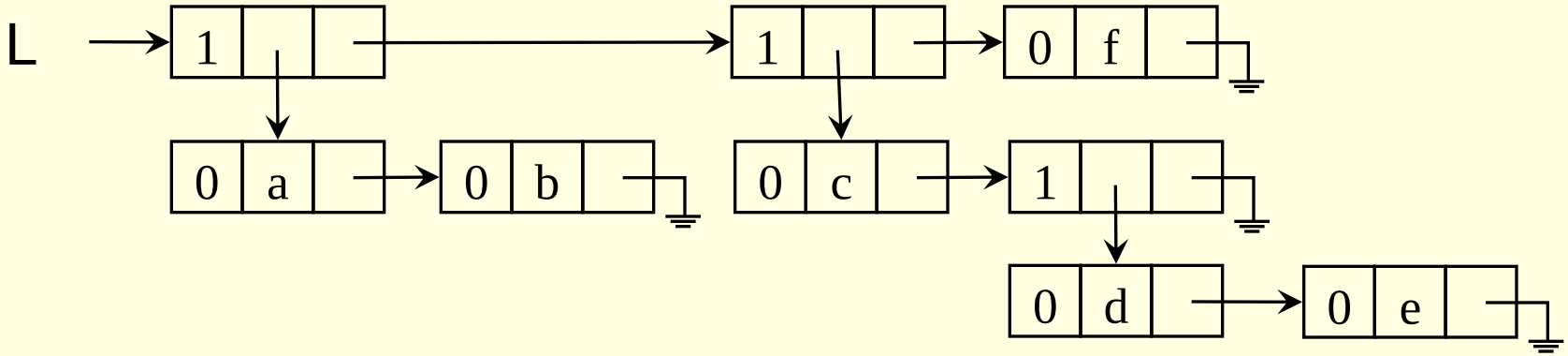
```
Função Busca (x, L):
Se L é vazia então retorna FALSE
Senão L = (l1, l2, ....ln) e
      se l1 é atomo então
            se l1 = x então retorna TRUE e x
            senão retorna Busca (x, (l2, l3,....ln))
      senão se Busca (x, l1) retorna TRUE e x
           senão retorna Busca (x, (l2, l3,....ln))
```

- Verificar se duas listas generalizadas, L1 e L2, são iguais
  - Tente fazer função booleana recursiva

```
Função Igual (K, L):
Se K e L são vazias então retorna TRUE;
Se K ou L é vazia então retorna FALSE;
/*ambas são não vazias: (k1,...kn) (l1,...lm)*/
Se k1 e l1 são átomos e são iguais
Então retorna Igual((k2,...kn), (l2,...lm))
Senão se k1 e l1 são sublistas
      então se Igual(k1, l1)
            então retorna Igual((k2,...kn), (l2,...lm))
      senão retorna FALSE
```

Listas e recursão

- Exercício extra
  - Implementar uma sub-rotina que determina a profundidade máxima de uma lista generalizada
    - Tente usar recursividade
  - Por exemplo, para o caso abaixo, a sub-rotina deveria retornar profundidade 3



Profundidade máxima de uma lista generalizada S

```
Função Profundidade ( S ):
Se S é atomo ou S = lista vazia então retorna 0
 senão{prof_atual = 0;
       para cada elemento elem de S:
            {prof :=  Profundidade(elem);
             se prof > prof_atual
                então prof_atual:= prof };
             retorna prof_atual + 1;
       }
```

Ex. S = (a, (b)) => Prof(S) = 2

A = (a, b, c) => Prof(A) = 1

B = () => Prof(B) = 0;

Uso de lista generalizada para representação de polinômios

- Considere polinômios em várias variáveis:

(1)$P(x,y,z) = 4x^2y^3z+3xy+5$

(2)$P(x,y,z) = x^{10}y^3z^2+2x^8y^2z^2+x^4y^4z+6x^3y^4z+2yz$

(3)$P(x,y) = 3x^2y$

(a) n° de termos: variável (1)=3 (2)=6 (3)=1

(b) n° de variáveis: variável (1) e (2)=3 (3)=2

(c) Nem todo termo é expresso em todas as variáveis.

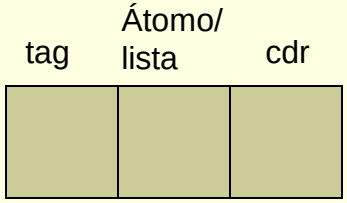
Uso de lista generalizada para representação de polinômios

Objetivos:

-representar de forma a otimizar o uso de memória.

-representação única para qualquer polinômio

Solução:

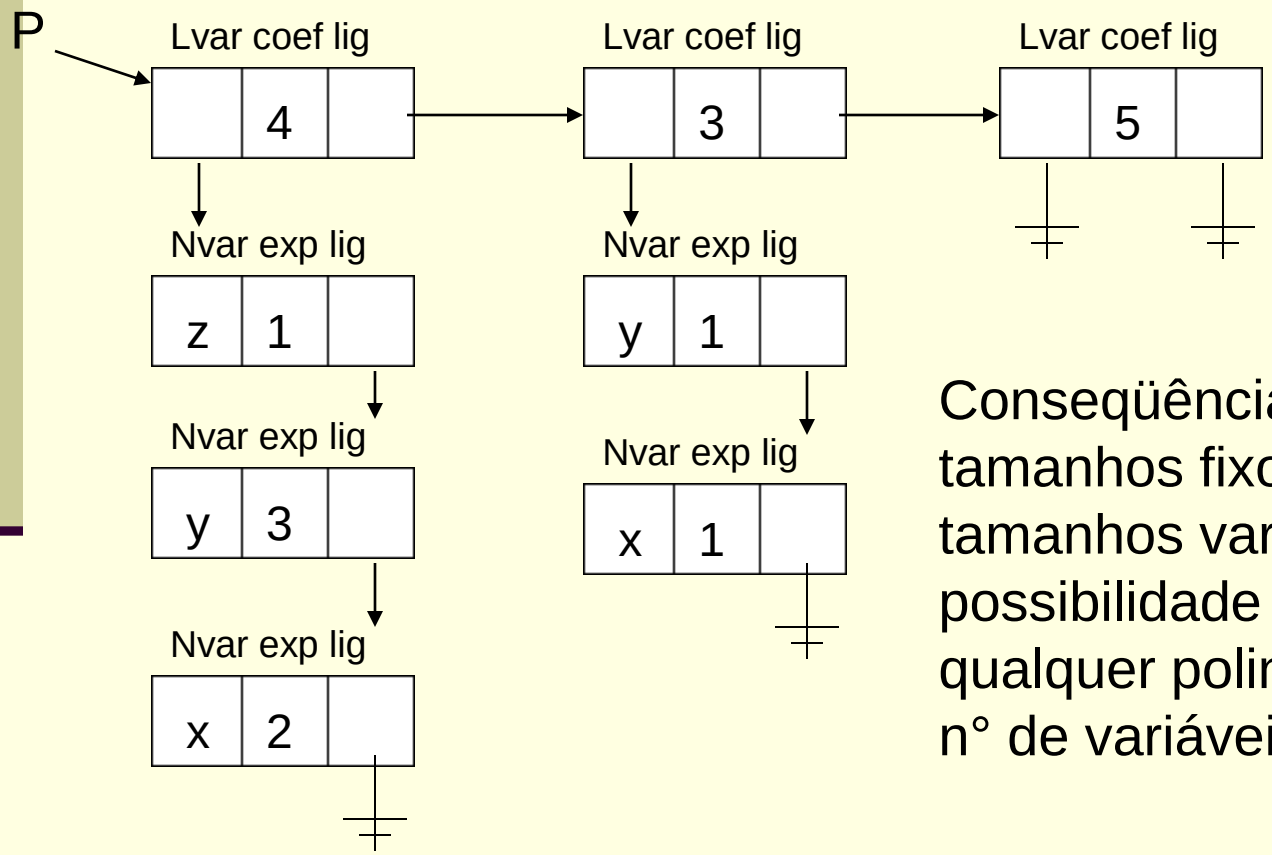


Onde: um polinômio P é uma lista generalizada em que cada elemento representa um termo.

Cada termo é composto por um coeficiente e uma lista dinâmica de variáveis (cada elemento da lista de variáveis tem o nome da variável e seu expoente).

Uso de lista generalizada para representação de polinômios

Ex: (1) $P(x,y,z) = 4x^2y^3z+3xy + 5$

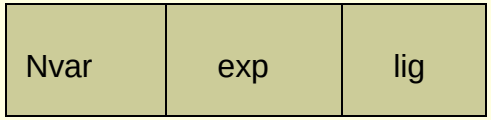


Conseqüência: registros de tamanhos fixos; listas de tamanhos variáveis; possibilidade de representar qualquer polinômio em qualquer n° de variáveis, qualquer grau

Uso de lista generalizada para representação de polinômios

Definição dos tipos:

```
Typedef struct tvar {
        char Nvar;
        int exp;
        struct tvar *lig;
  } tipovar;
Typedef struct ttermo {
        float coef;
        tipovar *Lvar;
        struct ttermo *lig;
  } tipotermo;
```



Complete a definição do tipo Polinomio

Exercícios

- Reflita sobre como seriam os algoritmos de operação de polinômios
- Quais seriam as funções presentes num TAD Polinomios?
  - Implemente-as