Listas encadeadas

Prof. Henrique Y. Shishido

Universidade Tecnológica Federal do Paraná

shishido@utfpr.edu.br

Crédito
Aluna: Renata Carina Soares (Estudante de eng. comp.)
Orientação: Prof. Dr. Danilo Sanches

Tópicos

Motivação

1. Lista Simplesmente Encadeada
2. Lista Duplamente Encadeada
3. Listas Circulares
4. Implementações Recursivas
5. Listas de Tipos Estruturados

5. Lista de Tipos Estruturados

5. Lista de tipos estruturados

- A **informação** associada a cada **nó** de uma lista encadeada pode ser mais **complexa**, sem alterar o encadeamento dos elementos.

- As funções apresentadas para manipular listas de inteiros podem ser adaptadas para tratar de outros tipos.

5. Lista de tipos estruturados

- O **campo da informação** pode ser representado por um ponteiro para uma estrutura, em lugar da estrutura em si.

- Independente da informação armazenada na lista, a **estrutura** do **nó** é sempre composta por:
  - Um ponteiro para a informação e
  - Um ponteiro para o próximo nó da lista.

Lista de retângulos

```
struct retangulo{
        float b;
        float h;
};
typedef struct retangulo Retangulo;

struct lista{
        Retangulo info;
        struct lista* prox;
};
```

- O campo da informação representado por um ponteiro para uma estrutura, em lugar da estrutura em si

Função auxiliar para alocar um nó

```
static ListaRec*aloca(float b, float h){

      Retangulo* r = (Retangulo*)malloc(sizeoff(Retangulo));
      ListaRec* p = (ListaRec)malloc(sizeoff(ListaRec));
      r -> b = b;
      r -> h = h;
      p -> info = r;
      p -> prox = NULL;
      return p;
}
```

- Para alocar um nó, são necessárias **duas alocações dinâmicas**; uma para criar a estrutura do retângulo e outra para criar a estrutura do nó.

- O valor da base associado a um nó p seria acessado por: **p->info->b.**

Listas heterogêneas

A representação da informação por um ponteiro permite construir listas heterogêneas, isto é, listas em que as **informãções** armazenadas **diferem** de nó para nó.

Exemplo:

- Listas de retângulos, triângulos ou círculos.
- Áreas desses objetos são dadas por:

$$r = b * h \qquad t = \frac{b*h}{2} \qquad c = \pi r^2$$

5. Lista de tipos estruturados

```
struct retangulo{
        float b;
        float h;
};
typedef struct retangulo Retangulo;

struct triangulo {
        float b;
        float h;
};
typedef struct triangulo Triangulo;

struct circulo{
        float r;
};
typedef struct circulo Circulo;
```

Listas homogêneas

Todos os **nós** contêm os mesmos campos:

- Um ponteiro para o **próximo** nó da lista.
- Um ponteiro para a estrutura que contém a **informação**.
  - Deve ser do tipo genérico (ou seja, do tipo void*) pois pode apontar para um retângulo, um triângulo ou um círculo.
- Um identificador indicando qual **objeto** o nó armazena.
  - Consultando esse identificador, o ponteiro genérico pode ser convertido no ponteiro especifíco para o objeto e os campos do objeto podem ser acessados.

5. Lista de tipos estruturados

```
/*Definição dos tipos de objetos*/
#define RET 0
#define TRI 1
#define CIR 2

/*Definição do nó da estrutura*/
struct listahet {
        int tipo;
        void *info;
        struct listahet* prox;
};
typedef struct listahet ListaHet;
```

Função para a criação de um nó da lista

```
ListaHet* cria_ret (float b, float h){

    Retangulo* r;
    ListaHet* p;

    /* aloca retângulo*/
    r = (Retangulo*)malloc(sizeoff(Retangulo));
    r -> b = b;
    r -> h = h;

    /* aloca nó*/
    p = (ListaHet*)malloc(sizeoff(ListaHet));
    p -> tipo = RET;
    p -> info = r;
    p -> prox = NULL;

    return p;
}
```

A função para a criação ed um nó possui três variações, uma cada tipo de objeto

Função para calcular a maior área

- Retorna a **maior área** entre os elementos da lista.
- Para **cada nó**, de acordo com o tipo de objeto que armazena, chama uma função específica para o **cálculo da área**.

5. Lista de tipos estruturados

```
/* função para cálculo da área de um retângulo*/
static float ret_area (Retangulo* r){
        return r->b *  r->h;
}

/* função para cálculo da área de um triângulo*/
static float tri_area (Triangulo* t){
        return (t->b *  t->h) / 2;
}

/* função para cálculo da área de um círculo*/
static float cir_area (Circulo* c){
        return PI * c->r * c->r;
}
```

5. Lista de tipos estruturados

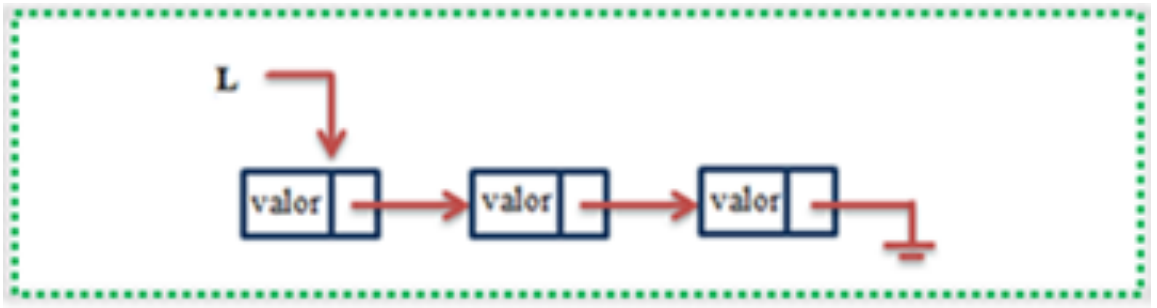
```
static float area (ListaHet* p){
    float a;
    switch(p->tipo){
        case RET;
            a = ret_area(p->info);
        break;

        case TRI;
            a = tri_area(p->info);
        break;

        case CIR;
            a = cir_area(p->info);
        break;
    }
    return a;
}
```

A conversão de ponteiro genérico para ponteiro específico ocorre quando uma das funções de cálculo da área é chamada:

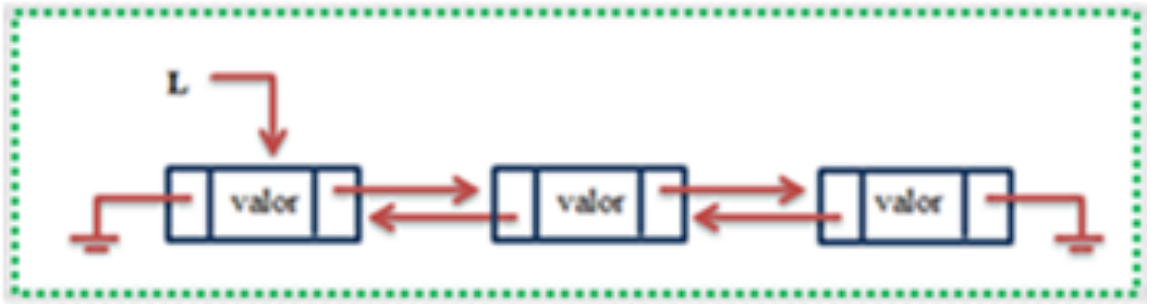
Passa-se um ponteiro genérico, que é atribuído a um ponteiro específico, por meio da conversão implícita de tipo.

Resumo

Lista Simplesmente Encadeada



Lista Duplamente Encadeada



Lista Circular