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

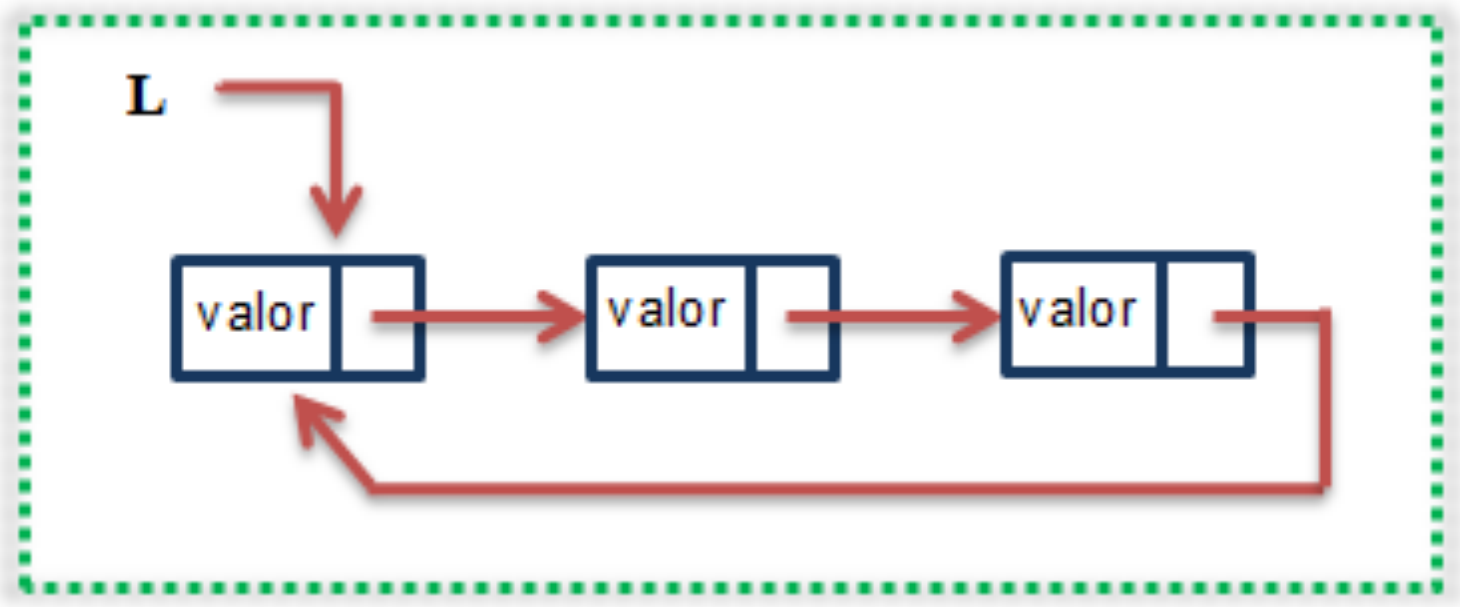
Motivação

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

3. Lista Circular

3. Lista circular (source: lst_circ_encadeada.c)

- O **último** elemento tem como próximo o **primeiro** nó da lista, formando um ciclo.
- A lista pode ser representada por um ponteiro para um elemento inicial qualquer da lista.



Função para imprimir os valores contidos na lista

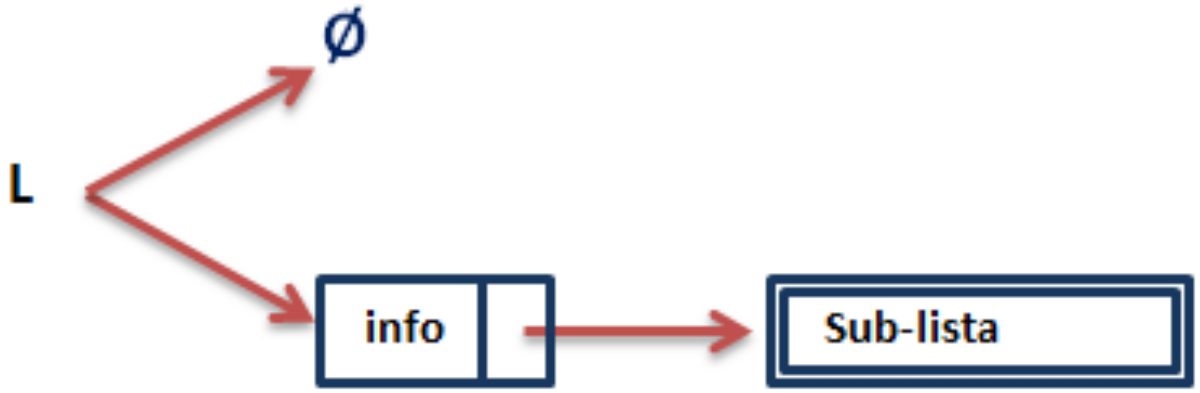
- Visita todos os nós a partir do ponteiro do elemento inicial até alcançar, novamente, esse elemento.
- Se a lista está vazia, o ponteiro para um nó inicial é NULL.

```
void imprimir(ListaCirc* L){

        ListaCirc* p = L;           /*faz p apontar para o nó inicial*/

   /*testa se a lista não é vazia e então percorre com do-while*/
        if(p) do{
                /*imprime a informação do nó*/
                printf("%d\n", p->info);
                p = p -> prox;          /*avança para o próximo nó*/
        }while(p != L);
}
```

4. Implementações Recursivas

Definição recursiva de lista

- Uma **lista** é:
  - uma lista vazia; *ou*
  - um elemento seguido de uma (sub-)lista.



Função para imprimir os valores contidos na lista

- Se *a **lista for vazia***, não imprima nada. Caso contrário:
  - Imprima a informação associada ao primeiro nó, dada por:

    L -> info

  - Imprima a sub-lista, dada por:

    L -> prox

    Chamando recursivamente a função.

Função para imprimir os valores contidos na lista

```
void imprimir(ListaRec* L){

        if(! lista_vazia(L)){

                /*imprime o primeiro nó*/
                printf("info: %d\n", L->info);
                /*imprime a sub-lista*/
                imprimir(L->prox);
        }
}
```

Função para retirar um elemento da lista

- Retire o elemento, se ele for o **primeiro** da lista (ou da sub-lista).
- Caso contrário, chame a função **recursivamente** para retirar o elemento da sub-lista.

4. Implementação recursiva (source: lst_simpl_encadeada_resuriva.c)

- É necessário re-atribuir o valor de **L->prox** na chamada recursiva, já que a função pode alterar o valor da sub-lista.

```
ListaRec* retirar_rec(ListaRec* L, int valor){

        if(! lista_vazia(L)){
        /*verifica se o elemento a ser retirado é o primeiro*/
                if(L->info == valor){
                        /*temporário para poder liberar*/
                        ListaRec* t = L;
                        L = L -> prox;
                        free(t);
                }else{  /*retirar da sub-lista*/
                        L -> prox = retirar_rec(L -> prox, valor);
                }
        }
        return L;
}
```