

Estruturas de Dados

# Listas Encadeadas – Exercícios

PONTIFÍCIA UNIVERSIDADE CATÓLICA  
DO RIO DE JANEIRO



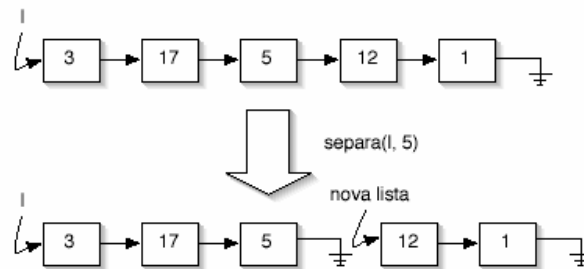
# Referências

Waldemar Celes, Renato Cerqueira, José Lucas Rangel,  
*Introdução a Estruturas de Dados*, Editora Campus  
(2004)

Exercícios Parte II (pág. 214)

# Exercícios Parte II

2.6. Considerando listas de valores inteiros, implemente uma função que receba como parâmetro uma lista encadeada e um valor inteiro n e divida a lista em duas, de tal forma que a segunda lista comece no primeiro nó logo após a primeira ocorrência de n na lista original. A figura a seguir ilustra essa separação:



Essa função deve obedecer ao protótipo:

```
Lista* separa (Lista* l, int n);
```

A função deve retornar um ponteiro para a segunda sub-divisão da lista original, enquanto l deve continuar apontando para o primeiro elemento da primeira sub-divisão da lista.

# Exercícios Parte II

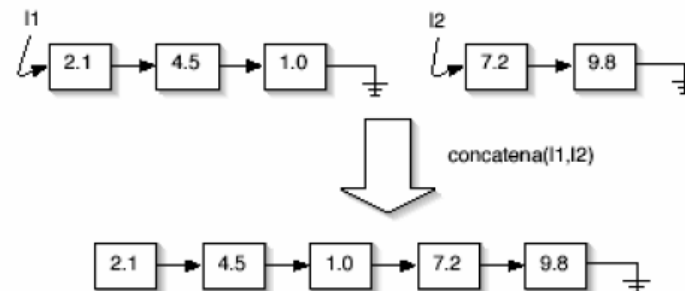
```
/* função separa */
Lista* separa (Lista* l, int n)
{
    Lista* p; /* variável auxiliar para percorrer a lista */
    Lista* q; /* variável auxiliar para nova lista */
    for (p = l; p != NULL ; p = p->prox)
        if (p->info == n)
            {q = p->prox; p->prox = NULL; return q;}
    return NULL;
}
```

# 2004.1 – P2 – Questão 1

Considere estruturas de listas encadeadas que armazenam valores reais. O tipo que representa um nó da lista é dado por:

```
struct lista {  
    float info;  
    struct lista* prox;  
};  
typedef struct lista Lista;
```

Implemente uma função que, dadas duas listas encadeadas *l1* e *l2*, concatene a lista *l2* no final da lista *l1*, conforme ilustra a figura abaixo.



A função deve retornar a lista resultante da concatenação, obedecendo ao protótipo:

```
Lista* concatena (Lista* l1, Lista* l2);
```

Observe que *l1* e/ou *l2* podem ser listas vazias.

# 2004.1 – P2 – Questão 1

```
/* concatena modificando l1 */
Lista* concatena (Lista* l1, Lista* l2)
{
    Lista* p; /* variável auxiliar para percorrer a lista */
    Lista* q; /* variável auxiliar para criar a nova lista */
    if (l1 == NULL) return l2;
    p = l1;
    do
    { q = p;
      p = p->prox;
    } while(p != NULL);
    q->prox = l2;
    return l1;
}
```

## 2004.1 – P2 – Questão 2

Considere estruturas de listas encadeadas que armazenam valores inteiros. O tipo que representa um nó da lista é dado por:

```
struct lista {
    int info;
    struct lista* prox;
};
typedef struct lista Lista;
```

Implemente uma função que receba um vetor de valores inteiros com  $n$  elementos e construa uma lista encadeada armazenando os elementos do vetor nos nós da lista. Assim, se for recebido o vetor  $v[5] = \{3, 8, 1, 7, 2\}$ , a função deve retornar uma nova lista cujo primeiro nó tem a informação 3, o segundo a informação 8, e assim por diante. Se o vetor tiver zero elementos, a função deve ter como valor de retorno uma lista vazia. O protótipo da função é dado por:

```
Lista* constroi (int n, int* v);
```

## 2004.1 – P2 – Questão 2

```
Lista* constroi (int n, int* v)
{
    Lista* p; /* variável auxiliar para percorrer a lista */
    Lista* q; /* variável auxiliar para criar a nova lista */
    int i;
    q = NULL;
    for(i=n-1; i>=0; i--)
    { p = (Lista*) malloc(sizeof(Lista));
      p->prox = q;
      p->info = v[i];
      q = p;
    }
    return q;
}
```

## 2004.2 – P2 – Questão 2

Considere a implementação de uma lista encadeada para armazenar números reais dada pelo tipo abaixo:

```
struct lista {
    float info;
    struct lista* prox;
};
typedef struct lista Lista;
```

Implemente uma função que, dados uma lista encadeada e um número inteiro não negativo  $n$ , remova da lista seus  $n$  primeiros nós e retorne a lista resultante. Caso  $n$  seja maior do que o comprimento da lista, todos os seus elementos devem ser removidos e o resultado da função deve ser uma lista vazia. Essa função deve obedecer o seguinte protótipo:

```
Lista* retira_prefixo (Lista* l, int n);
```

## 2004.2 – P2 – Questão 2

```
Lista* retira_prefixo (Lista* l, int n)
{
    Lista* p; /* variável auxiliar para percorrer a lista */
    Lista* q; /* variável auxiliar */
    int m = 1;
    p = l;
    while(p != NULL && m <= n)
    { m = m+1;
      q = p->prox;
      free(p);
      p = q;
    }
    return p;
}
```

# 2005.1 – P2 – Questão 1

Considere uma lista simplesmente encadeada que armazena os seguintes dados de alunos de uma disciplina:

- Número de matrícula: número inteiro
- Nome: com até 80 caracteres
- Média na disciplina: número de ponto flutuante

(a) Considerando que o tipo que representa um nó da lista é dado por:

```
typedef struct lista Lista;
```

Defina a estrutura denominada `lista`, que tenha os campos apropriados para guardar as informações de um aluno na lista, conforme descrito acima.

(b) Implemente uma função que insira, *em ordem crescente de número de matrícula*, os dados de um novo aluno na lista. Essa função deve obedecer ao seguinte protótipo, retornando o ponteiro para o primeiro elemento:

```
Lista* ins_ordenado (Lista* l, int mat, char* nome, float nota);
```

# 2005.1 – P2 – Questão 1

```
struct lista {  
    int matricula;  
    char nome[81];  
    float media;  
    struct lista* prox;  
}  
typedef struct lista Lista;
```

# 2005.1 – P2 – Questão 1

```
Lista* ins_ordenado(Lista* L, int mat, char* nome, float nota)
{
    Lista* p = L;
    Lista ant = NULL;

    Lista* novo = (Lista*)malloc(sizeof(Lista));
    novo ->mat = mat;
    strcpy(novo->nome, nome);
    novo->media = nota;
    novo->prox = NULL;
    if (p==NULL) return novo;

    while ((p!=NULL) && (novo->mat > p->mat))
    { ant = p; p = p->prox;}

    if (ant == NULL) {novo->prox = p; return novo;}
    novo ->prox = ant->prox;
    ant->prox = novo;
    return L;
}
```