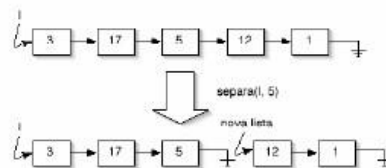


Exercícios de Listas Encadeadas

Observação: Para toda implementação do exercício para lista simples. Todas as novas funções deverão ser incorporadas às bibliotecas criadas anteriormente (TADs). Os testes deverão ser demonstrados no envio da resolução dos exercícios.

1. 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.

```

17 }
18
19 p2 = p1->prox;
20 p1->prox = NULL;
21
22 return p2;
23 }
24
25 Lista* insere(Lista *l, int valor){
26     Lista *novo;
27     novo = (Lista*)malloc(sizeof(Lista));
28     novo->num = valor;
29     novo->prox = l;
30     return novo;
31 }
32
33 void imprimir(Lista *l){
34     Lista *p = l;
35     while(p!=NULL){
36         printf("%x | %d | %x\n", p, p->num, p->prox);
37         p = p->prox;
38     }
39 }
40
41 int main(){
42     Lista *prox1;
43     Lista *prox2;
44
45     prox1 = NULL;
46
47     prox1 = insere(prox1, 1);
48     prox1 = insere(prox1, 12);
49     prox1 = insere(prox1, 5);
50     prox1 = insere(prox1, 17);
51     prox1 = insere(prox1, 3);
52     imprimir(prox1);
53
54     prox2 = separa(prox1, 5);
55     printf("\n\n");
56     imprimir(prox1);
57     printf("\n\n");
58     imprimir(prox2);
59
60     return 0;
61 }
62
63

```

```

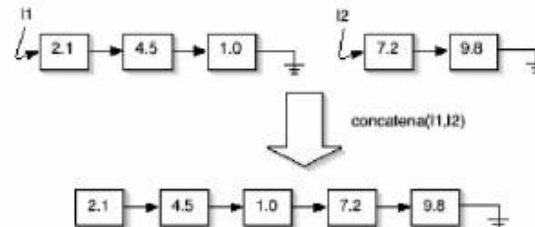
D:\Documentos\UTFPR\Kenji2023.2\Estrutura de Dados\Lista de Exercícios\Exercício 1.exe
d21618 [3 | d21608] -> d21608 [17 | d215f8] -> d215f8 [5 | d215e8] -> d215e8 [12 | d21570] -> d21570 [1 | 0] ->
d21618 [3 | d21608] -> d21608 [17 | d215f8] -> d215f8 [5 | 0] ->
d215e8 [12 | d21570] -> d21570 [1 | 0] ->
Process returned 0   execution time : 0.363 s
Press any key to continue.

```

2. 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.

```
22 while (p1->prox != NULL) {  
23     p1 = p1->prox;  
24 }  
25  
26 p1->prox = l2;  
27 return l1;  
28 }  
29  
30  
31  
32 Lista* insere(Lista *l, float valor){  
33     Lista *novo;  
34     novo = (Lista*)malloc(sizeof(Lista));  
35     novo->info = valor;  
36     novo->prox = l;  
37     return novo;  
38 }  
39  
40 void imprimir(Lista *l){  
41     Lista *p = l;  
42     while(p!=NULL){  
43         printf("%f | %f | %f", p->info, p->prox->info, p->prox->prox->info);  
44         p = p->prox;  
45     }  
46 }  
47  
48  
49 int main(){  
50     Lista *prox1;  
51     Lista *prox2;  
52  
53     prox1 = NULL;  
54     prox2 = NULL;  
55  
56     prox1 = insere(prox1, 1.0);  
57     prox1 = insere(prox1, 4.5);  
58     prox1 = insere(prox1, 2.1);  
59  
60     prox2 = insere(prox2, 9.8);  
61     prox2 = insere(prox2, 7.2);  
62  
63     prox1 = concatena(prox1, prox2);  
64     printf("\n");  
65     imprimir(prox1);  
66  
67     return 0;  
68 }
```

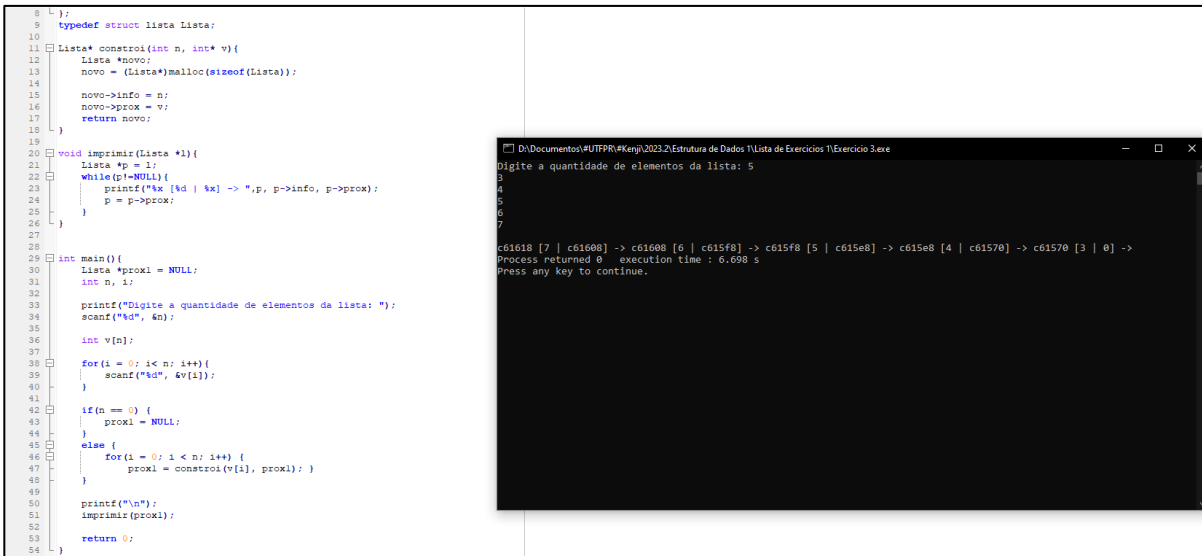
```
D:\Documentos\UFRRJ\camp2023-2\Estrutura de Dados I\Lista de Exercícios I\Exercício 2.exe  
c815f8 [2.100000 | c815e8 [4.500000 | c81570 [1.000000 | c81618 [7.200000 | c81608 [9.800000 | 0] ->  
Process returned 0   execution time : 0.226 s  
Press any key to continue.
```

3. 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);
```



The image shows a C program in a code editor on the left and its execution output in a terminal window on the right.

Code Editor (Left):

```
1 //
2
3 typedef struct lista Lista;
4
5 Lista* constroi(int n, int* v){
6     Lista *novo;
7     novo = (Lista*)malloc(sizeof(Lista));
8
9     novo->info = n;
10    novo->prox = v;
11    return novo;
12 }
13
14 void imprimir(Lista *l){
15     Lista *p = l;
16     while(p!=NULL){
17         printf("%d | %x" -> "p, p->info, p->prox);
18         p = p->prox;
19     }
20 }
21
22 int main(){
23     Lista *prox1 = NULL;
24     int n, i;
25
26     printf("Digite a quantidade de elementos da lista: ");
27     scanf("%d", &n);
28
29     int v[n];
30
31     for(i = 0; i < n; i++){
32         scanf("%d", &v[i]);
33     }
34
35     if(n == 0) {
36         prox1 = NULL;
37     }
38     else {
39         for(i = 0; i < n; i++) {
40             prox1 = constroi(v[i], prox1);
41         }
42     }
43
44     printf("\n");
45     imprimir(prox1);
46
47     return 0;
48 }
```

Terminal Window (Right):

DA\Documentos\UTFPR\#Kenj\2023.2\Estrutura de Dados\1Lista de Exercícios\1Exercício 3.exe

Digite a quantidade de elementos da lista: 5

3
8
1
7
2

c61618 [7 | c61608] -> c61608 [6 | c615f8] -> c615f8 [5 | c615e8] -> c615e8 [4 | c61570] -> c61570 [3 | 0] ->

Process returned 0 execution time : 6.698 s

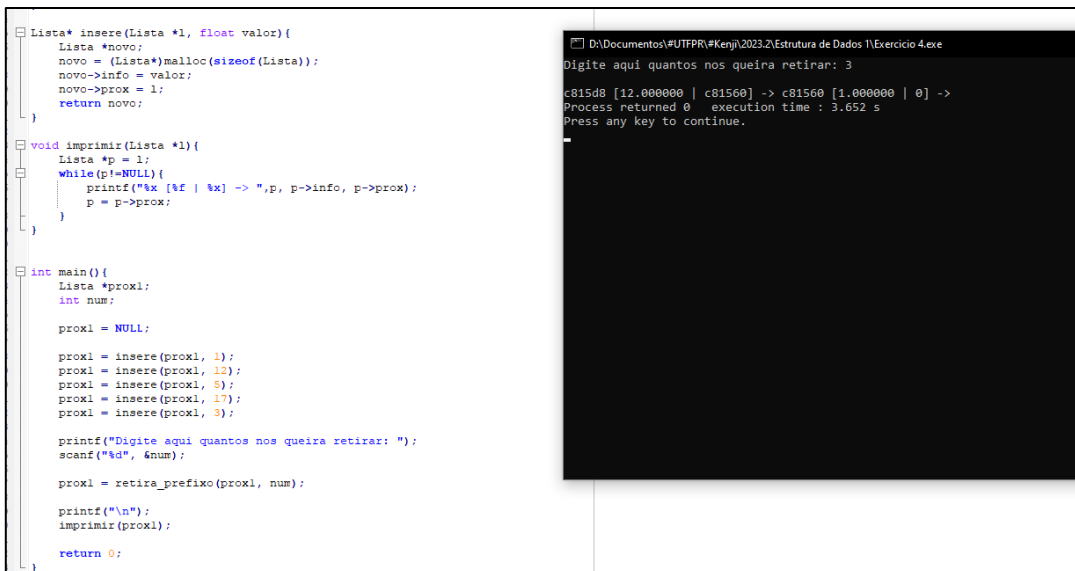
Press any key to continue.

4. 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);
```



The image shows a C program in a code editor on the left and its execution output in a terminal window on the right.

Code Editor:

```
#include <stdio.h>
#include <stdlib.h>

Lista* insere(Lista *l, float valor){
    Lista *novo;
    novo = (Lista*)malloc(sizeof(Lista));
    novo->info = valor;
    novo->prox = l;
    return novo;
}

void imprimir(Lista *l){
    Lista *p = l;
    while(p!=NULL){
        printf("%x [%f | %x] -> ", p, p->info, p->prox);
        p = p->prox;
    }
}

int main(){
    Lista *prox1;
    int num;

    prox1 = NULL;

    prox1 = insere(prox1, 1);
    prox1 = insere(prox1, 12);
    prox1 = insere(prox1, 5);
    prox1 = insere(prox1, 17);
    prox1 = insere(prox1, 3);

    printf("Digite aqui quantos nos queira retirar: ");
    scanf("%d", &num);

    prox1 = retira_prefixo(prox1, num);

    printf("\n");
    imprimir(prox1);

    return 0;
}
```

Terminal Output:

```
D:\Documentos\UTFPR\Kenji\2023.2\Estrutura de Dados 1\Exercicio 4.exe
Digite aqui quantos nos queira retirar: 3
c815d8 [12.000000 | c81560] -> c81560 [1.000000 | 0] ->
Process returned 0   execution time : 3.652 s
Press any key to continue.
```