

roteiro-04/lista_seq_est.h

```
1  /*----- File: Lista.h -----+
2  |Lista Sequencial Estatica      |
3  |                               |
4  |                               |
5  | Implementado por Guilherme C. Pena em 12/09/2023 |
6  +-----+ */
7
8  #ifndef LISTA_H
9  #define LISTA_H
10
11 #include <stdio.h>
12 #include <stdlib.h>
13
14 #define MAX 100
15
16 typedef struct {
17     int qtd;
18     int dados[MAX];
19 } Lista;
20
21 Lista *criaLista() {
22     Lista *li;
23     li = (Lista *)malloc(sizeof(Lista));
24     if (li != NULL)
25         li->qtd = 0;
26     return li;
27 }
28
29 void destroiLista(Lista *li) {
30     if (li != NULL)
31         free(li);
32 }
33
34 int tamanhoLista(Lista *li) {
35     if (li == NULL)
36         return -1;
37     return li->qtd;
38 }
39
40 int listaCheia(Lista *li) {
41     if (li == NULL)
42         return -1;
43     return (li->qtd == MAX);
44 }
45
46 int listaVazia(Lista *li) {
47     if (li == NULL)
48         return -1;
49     return (li->qtd == 0);
50 }
51
52 int insereFim(Lista *li, int elem) {
53     if (li == NULL) return 0;
54     if (!listaCheia(li)) {
55         li->dados[li->qtd] = elem;
```

```

56         li->qtd++;
57         return 1;
58     } else {
59         return 0;
60     }
61 }
62
63 int insereIni(Lista *li, int elem) {
64     if (li == NULL) return 0;
65     if (!listaCheia(li)) {
66         int i;
67         for (i = li->qtd; i > 0; i--) {
68             li->dados[i] = li->dados[i - 1];
69         }
70         li->dados[0] = elem;
71         li->qtd++;
72         return 1;
73     } else {
74         return 0;
75     }
76 }
77
78 int imprimeLista(Lista *li) {
79     if (li == NULL) return 0;
80     int i;
81     printf("Elementos:\n");
82     for (i = 0; i < li->qtd; i++) {
83         printf("%d ", li->dados[i]);
84     }
85     printf("\n");
86     return 1;
87 }
88
89 int removeFim(Lista *li) {
90     if (li == NULL) return 0;
91     if (!listaVazia(li)) {
92         li->qtd--;
93         return 1;
94     } else
95         return 0;
96 }
97
98 int removeIni(Lista *li) {
99     if (li == NULL) return 0;
100    if (!listaVazia(li)) {
101        int i;
102        for (i = 0; i < li->qtd - 1; i++)
103            li->dados[i] = li->dados[i + 1];
104        li->qtd--;
105        return 1;
106    } else {
107        return 0;
108    }
109 }
110
111 #endif

```

roteiro-04/ex01-02.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  #include "lista_seq_est.h"
5
6  int procura(Lista* li, int x) {
7      for (int i = 0; i < tamanhoLista(li); i++) {
8          if (li->dados[i] == x) {
9              return i;
10         }
11     }
12     return -1;
13 }
14
15 int main() {
16     Lista* l = criaLista();
17
18     insereFim(l, 23);
19     insereFim(l, 14);
20     insereFim(l, 7);
21     insereFim(l, 85);
22     insereFim(l, 19);
23     insereFim(l, 56);
24
25     imprimeLista(l);
26     printf("Indice do 19 = %d\n", procura(l, 19));
27
28     destroiLista(l);
29
30     return 0;
31 }
```

Terminal

```
gabriel.meira.2004@lab-02-07-03:~/Desktop/ufsj/UFSJ-LabProg2/roteiro-04$ ./ex01-02
Elementos:
23 14 7 85 19 56
Indice do 19 = 4
gabriel.meira.2004@lab-02-07-03:~/Desktop/ufsj/UFSJ-LabProg2/roteiro-04$
```

roteiro-04/ex01-03.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  #include "lista_seq_est.h"
5
6  int insereOrdenado(Lista* li, int x) {
7      if (li == NULL || listaCheia(li)) return 0;
8
9      int i;
10     for (i = 0; i < tamanhoLista(li) && li->dados[i] < x; i++)
11         ;
12     for (int j = li->qtd; j > i; j--) {
13         li->dados[j] = li->dados[j - 1];
14     }
15     li->dados[i] = x;
16     li->qtd++;
17
18     return 1;
19 }
20
21 int main() {
22     Lista* l = criaLista();
23
24     insereOrdenado(l, 23);
25     imprimeLista(l);
26     insereOrdenado(l, 14);
27     imprimeLista(l);
28     insereOrdenado(l, 7);
29     imprimeLista(l);
30     insereOrdenado(l, 85);
31     imprimeLista(l);
32     insereOrdenado(l, 19);
33     imprimeLista(l);
34     insereOrdenado(l, 56);
35     imprimeLista(l);
36
37     destroiLista(l);
38
39     return 0;
40 }
```

Terminal

```
gabriel.meira.2004@lab-02-07-03:~/Desktop/ufsj/UFSJ-LabProg2/roteiro-04$ ./ex01-03
Elementos:
23
Elementos:
14 23
Elementos:
7 14 23
Elementos:
7 14 23 85
Elementos:
7 14 19 23 85
Elementos:
7 14 19 23 56 85
gabriel.meira.2004@lab-02-07-03:~/Desktop/ufsj/UFSJ-LabProg2/roteiro-04$
```

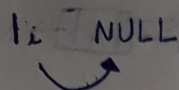
roteiro-04/ex01-04.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  #include "lista_seq_est.h"
5
6  int removePrimeiroX(Lista* li, int x) {
7      if (li == NULL || listaVazia(li)) return 0;
8
9      int i;
10     for (i = 0; i < li->qtd; i++) {
11         if (li->dados[i] == x) break;
12     }
13
14     if (i < li->qtd) {
15         for (int j = i; j < li->qtd; j++) {
16             li->dados[j] = li->dados[j + 1];
17         }
18         li->qtd--;
19     }
20
21     return 1;
22 }
23
24 int main() {
25     Lista* l = criaLista();
26
27     insereFim(l, 23);
28     insereFim(l, 14);
29     insereFim(l, 7);
30     insereFim(l, 85);
31     insereFim(l, 19);
32     insereFim(l, 56);
33
34     imprimeLista(l);
35     removePrimeiroX(l, 7);
36     imprimeLista(l);
37
38     destroiLista(l);
39
40     return 0;
41 }
```

```
gabriel.meira.2004@lab-02-07-03:~/Desktop/ufsj/UFSJ-LabProg2/roteiro-04$ ./ex01-04
Elementos:
23 14 7 85 19 56
Elementos:
23 14 85 19 56
gabriel.meira.2004@lab-02-07-03:~/Desktop/ufsj/UFSJ-LabProg2/roteiro-04$
```

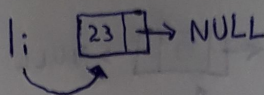

2.1) Lista Simplesmente Encadeada

I) Lista vazia



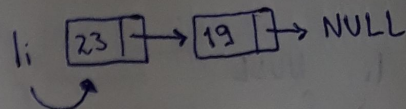
- criar ponteiro "li" para NULL

II) Inserção (23)



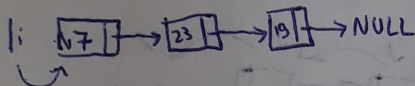
- Criar novo nó (23)
- "li" aponta para o novo nó
- o nó aponta para NULL

III) Inserção Fim (19)



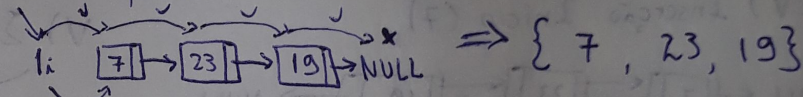
- criar novo nó (19)
- "23" que apontava para NULL vai apontar para "19"
- "19" vai apontar para NULL

IV) Inserção Início



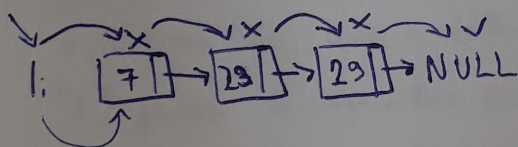
- Criar novo nó (7)
- "7" vai apontar para o nó que "li" aponta (19)
- "li" agora aponta para "7"

V) Impressão



- se a lista for vazia não imprime
- caso tenha elementos percorre todos os nós até encontrar NULL.

VI) Liberação da Lista



{ Del: 7, Del: 23, Del: 19 }

- caso a lista esteja vazia, libera a lista
- caso haja elementos, percorre de 1 em 1 cada nó, desalocando a memória, até que chegue em NULL, executando o passo anterior

roteiro-04/lista_sim_enc.h

```
1  /*----- File: LSE.h -----+
2  |Lista Simplesmente Encadeada |
3  | Implementado por Guilherme C. Pena em 14/09/2023 |
4  +-----+ */
5  #ifndef LISTASE_H
6  #define LISTASE_H
7
8  #include <stdio.h>
9  #include <stdlib.h>
10
11 typedef struct NO {
12     int info;
13     struct NO* prox;
14 } NO;
15 typedef struct NO* Lista;
16
17 Lista* criaLista() {
18     Lista* li;
19     li = (Lista*)malloc(sizeof(Lista));
20     if (li != NULL) {
21         *li = NULL;
22     }
23     return li;
24 }
25
26 int listaVazia(Lista* li) {
27     if (li == NULL) return 1;
28     if (*li == NULL) return 1; // True - Vazia!
29     return 0; // False - tem elemento!
30 }
31
32 NO* alocarNO() {
33     return (NO*)malloc(sizeof(NO));
34 }
35
36 void liberarNO(NO* q) {
37     free(q);
38 }
39
40 int insereIni(Lista* li, int elem) {
41     if (li == NULL) return 0;
42     NO* novo = alocarNO();
43     if (novo == NULL) return 0;
44     novo->info = elem;
45     novo->prox = *li;
46     *li = novo;
47     return 1;
48 }
49
50 int insereFim(Lista* li, int elem) {
51     if (li == NULL) return 0;
52     NO* novo = alocarNO();
53     if (novo == NULL) return 0;
54     novo->info = elem;
55     novo->prox = NULL;
56     if (listaVazia(li)) {
57         *li = novo;
58     } else {
```

```

59     NO* aux = *li;
60     while (aux->prox != NULL)
61         aux = aux->prox;
62     aux->prox = novo;
63 }
64 return 1;
65 }
66
67 int removeIni(Lista* li) {
68     if (li == NULL) return 0;
69     if (listaVazia(li)) return 0;
70     NO* aux = *li;
71     *li = aux->prox;
72     liberarNO(aux);
73     return 1;
74 }
75
76 int removeFim(Lista* li) {
77     if (li == NULL) return 0;
78     if (listaVazia(li)) return 0;
79     NO *ant, *aux = *li;
80     while (aux->prox != NULL) {
81         ant = aux;
82         aux = aux->prox;
83     }
84     if (aux == *li)
85         *li = aux->prox;
86     else
87         ant->prox = aux->prox;
88     liberarNO(aux);
89     return 1;
90 }
91
92 void imprimeLista(Lista* li) {
93     if (li == NULL) return;
94     if (listaVazia(li)) {
95         printf("Lista Vazia!\n");
96         return;
97     }
98     printf("Elementos:\n");
99     NO* aux = *li;
100    while (aux != NULL) {
101        printf("%d ", aux->info);
102        aux = aux->prox;
103    }
104    printf("\n");
105 }
106
107 void destroiLista(Lista* li) {
108     if (li != NULL) {
109         NO* aux;
110         while ((*li) != NULL) {
111             aux = *li;
112             *li = (*li)->prox;
113             liberarNO(aux);
114         }
115         free(li);
116     }
117 }
118
119 #endif

```

roteiro-04/ex02-02.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  #include "lista_sim_enc.h"
5
6  int tamanho(Lista* li) {
7      int t = 0;
8      NO* p = *li;
9      while (p != NULL) {
10         p = p->prox;
11         t++;
12     }
13
14     return t;
15 }
16
17 int procura(Lista* li, int x) {
18     int i = 0;
19     NO* p = *li;
20     while (p != NULL) {
21         if (p->info == x) return i;
22         p = p->prox;
23         i++;
24     }
25
26     return -1;
27 }
28
29 int main() {
30     Lista* l = criaLista();
31
32     insereFim(l, 23);
33     insereFim(l, 14);
34     insereFim(l, 7);
35     insereFim(l, 85);
36     insereFim(l, 19);
37     insereFim(l, 56);
38
39     imprimeLista(l);
40     printf("Tamanho = %d\n", tamanho(l));
41     printf("Indice do 19 = %d\n", procura(l, 19));
42
43     destroiLista(l);
44
45     return 0;
46 }
```

Terminal

```
gabriel.meira.2004@lab-02-07-03:~/Desktop/ufsj/UFSJ-LabProg2/roteiro-04$ ./ex02-02
Elementos:
23 14 7 85 19 56
Tamanho = 6
Indice do 19 = 4
gabriel.meira.2004@lab-02-07-03:~/Desktop/ufsj/UFSJ-LabProg2/roteiro-04$
```

roteiro-04/ex02-03.c

```
6  int insereOrdenado(Lista* li, int x) {
7      if (li == NULL) return 0;
8
9      NO* novo = alocarNO();
10     novo->info = x;
11     novo->prox = NULL;
12
13     if (listaVazia(li)) {
14         *li = novo;
15     } else {
16         NO i;
17         i.prox = *li;
18
19         NO* p = &i;
20         while (p->prox != NULL && p->prox->info < x) {
21             p = p->prox;
22         }
23
24         if (p->prox == *li) {
25             novo->prox = *li;
26             *li = novo;
27         }
28
29         novo->prox = p->prox;
30         p->prox = novo;
31     }
32
33     return 1;
34 }
```


Terminal

```
gabriel.meira.2004@lab-02-07-03:~/Desktop/ufsj/UFSJ-LabProg2/roteiro-04$ ./ex02-03
Elementos:
23
Elementos:
14 23
Elementos:
7 14 23
Elementos:
7 14 23 85
Elementos:
7 14 19 23 85
Elementos:
7 14 19 23 56 85
gabriel.meira.2004@lab-02-07-03:~/Desktop/ufsj/UFSJ-LabProg2/roteiro-04$
```

roteiro-04/ex02-04.c

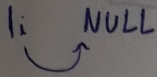
```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  #include "lista_sim_enc.h"
5
6  int removePrimeiroX(Lista* li, int x) {
7      if (li == NULL || listaVazia(li)) return 0;
8
9      NO i; // No inicial auxiliar
10     i.prox = *li;
11
12     NO* p = &i;
13     while (p->prox != NULL) {
14         if (p->prox->info == x) {
15             NO* aux = p->prox;
16             p->prox = aux->prox;
17             liberarNO(aux);
18             break;
19         }
20
21         p = p->prox;
22     }
23
24     return 1;
25 }
26
27 int main() {
28     Lista* l = criaLista();
29
30     insereFim(l, 23);
31     insereFim(l, 14);
32     insereFim(l, 7);
33     insereFim(l, 85);
34     insereFim(l, 19);
35     insereFim(l, 56);
36
37     imprimeLista(l);
38     removePrimeiroX(l, 7);
39     imprimeLista(l);
40
41     destroiLista(l);
42
43     return 0;
44 }
```



```
gabriel.meira.2004@lab-02-07-03:~/Desktop/ufsj/UFSJ-LabProg2/roteiro-04$ ./ex02-04
Elementos:
23 14 7 85 19 56
Elementos:
23 14 85 19 56
gabriel.meira.2004@lab-02-07-03:~/Desktop/ufsj/UFSJ-LabProg2/roteiro-04$
```

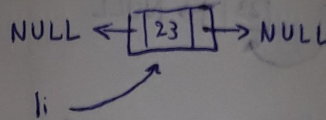
3.1) Lista Duplamente Encadeada

I) Lista vazia



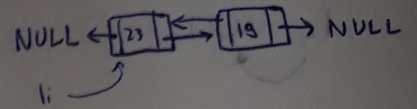
- Cria ponteiro "li" para NULL

II) Inserção (23)



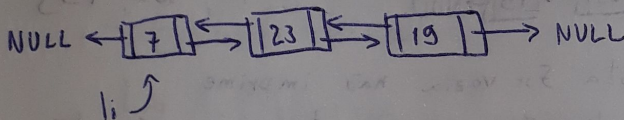
- Cria novo nó (23)
- anterior e próximo são NULL
- "li" aponta para "23"

III) Inserção Fim (19)



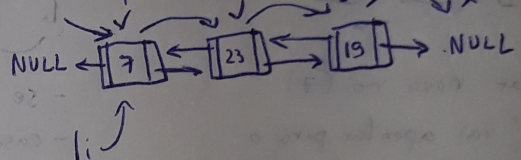
- Cria novo nó (19)
- "23" deixa de apontar para o próximo NULL e aponta para "19", cujo próximo é NULL
- O anterior de "19" é "23"

IV) Inserção Início (7)



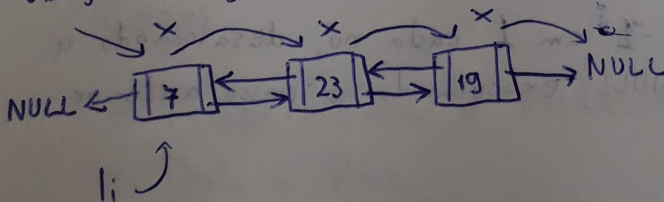
- Cria novo nó (7)
- o anterior de "7" é NULL e o próximo é o nó apontado por "li"
- o anterior de "li" será o "7"
- "li" agora será o "7"

V) Imprimir



- caso a lista seja vazia não imprime
- percorre todos os nós imprimindo seus dados até encontrar NULL

VI) Liberação da lista



{ Del: 7, Del: 23, Del: 19 }

- caso a lista esteja vazia, libera a lista
- caso haja elementos, percorre a lista liberando cada nó até chegar em NULL

roteiro-04/lista_dup_enc.h

```
1  /*----- File: LDE.h -----+
2  |Lista Duplamente Encadeada    |
3  |                               |
4  |                               |
5  | Implementado por Guilherme C. Pena em 19/09/2023 |
6  +-----+ */
7
8  #ifndef LDE_H
9  #define LDE_H
10
11 #include <stdio.h>
12 #include <stdlib.h>
13
14 typedef struct NO {
15     int info;
16     struct NO* prox;
17     struct NO* ant;
18 } NO;
19
20 typedef struct NO* Lista;
21
22 Lista* criaLista() {
23     Lista* li;
24     li = (Lista*)malloc(sizeof(Lista));
25     if (li != NULL) {
26         *li = NULL;
27     }
28     return li;
29 }
30
31 int listaVazia(Lista* li) {
32     if (li == NULL) return 1;
33     if (*li == NULL) return 1; // True - Vazia!
34     return 0;                 // False - tem elemento!
35 }
36
37 NO* alocarNO() {
38     return (NO*)malloc(sizeof(NO));
39 }
40
41 void liberarNO(NO* q) {
42     free(q);
43 }
44
45 int insereIni(Lista* li, int elem) {
46     if (li == NULL) return 0;
47     NO* novo = alocarNO();
48     if (novo == NULL) return 0;
49     novo->info = elem;
50     novo->prox = *li;
51     novo->ant = NULL;
52     if (!listaVazia(li))
53         (*li)->ant = novo;
54     *li = novo;
55     return 1;
56 }
```

```

56 }
57
58 int insereFim(Lista* li, int elem) {
59     if (li == NULL) return 0;
60     NO* novo = alocarNO();
61     if (novo == NULL) return 0;
62     novo->info = elem;
63     novo->prox = NULL;
64     if (listaVazia(li)) {
65         novo->ant = NULL;
66         *li = novo;
67     } else {
68         NO* aux = *li;
69         while (aux->prox != NULL)
70             aux = aux->prox;
71         aux->prox = novo;
72         novo->ant = aux;
73     }
74     return 1;
75 }
76
77 int removeIni(Lista* li) {
78     if (li == NULL) return 0;
79     if (listaVazia(li)) return 0;
80     NO* aux = *li;
81     *li = aux->prox;
82     if (aux->prox != NULL)
83         aux->prox->ant = NULL;
84     liberarNO(aux);
85     return 1;
86 }
87
88 int removeFim(Lista* li) {
89     if (li == NULL) return 0;
90     if (listaVazia(li)) return 0;
91     NO* aux = *li;
92     while (aux->prox != NULL)
93         aux = aux->prox;
94     if (aux->ant == NULL)
95         *li = aux->prox;
96     else
97         aux->ant->prox = NULL;
98     liberarNO(aux);
99     return 1;
100 }
101
102 void imprimeLista(Lista* li) {
103     if (li == NULL) return;
104     if (listaVazia(li)) {
105         printf("Lista Vazia!\n");
106         return;
107     }
108     printf("Elementos:\n");
109     NO* aux = *li;
110     while (aux != NULL) {
111         printf("%d ", aux->info);
112         aux = aux->prox;
113     }
114     printf("\n");

```

```
115 }
116
117 void destroiLista(Lista* li) {
118     if (li != NULL) {
119         NO* aux;
120         while ((*li) != NULL) {
121             aux = *li;
122             *li = (*li)->prox;
123             // printf("Destruindo.. %d\n", aux->info);
124             liberarNO(aux);
125         }
126         free(li);
127     }
128 }
129
130 #endif
```

roteiro-04/ex03-02.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  #include "lista_dup_enc.h"
5
6  int tamanho(Lista* li) {
7      int t = 0;
8      NO* p = *li;
9      while (p != NULL) {
10         p = p->prox;
11         t++;
12     }
13
14     return t;
15 }
16
17 int procura(Lista* li, int x) {
18     int i = 0;
19
20     NO* p = *li;
21     while (p != NULL) {
22         if (p->info == x) return i;
23         p = p->prox;
24         i++;
25     }
26
27     return -1;
28 }
29
30 int main() {
31     Lista* l = criaLista();
32
33     insereFim(l, 23);
34     insereFim(l, 14);
35     insereFim(l, 7);
36     insereFim(l, 85);
37     insereFim(l, 19);
38     insereFim(l, 56);
39
40     imprimeLista(l);
41     printf("Tamanho = %d\n", tamanho(l));
42     printf("Indice do 19 = %d\n", procura(l, 19));
43
44     destroiLista(l);
45
46     return 0;
47 }
48
```


Terminal

```
gabriel.meira.2004@lab-02-07-03:~/Desktop/ufsj/UFSJ-LabProg2/roteiro-04$ ./ex03-02
Elementos:
23 14 7 85 19 56
Tamanho = 6
Indice do 19 = 4
gabriel.meira.2004@lab-02-07-03:~/Desktop/ufsj/UFSJ-LabProg2/roteiro-04$
```

roteiro-04/ex03-03.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  #include "lista_dup_enc.h"
5
6  int insereOrdenado(Lista* li, int x) {
7      if (li == NULL) return 0;
8
9      NO* novo = alocarNO();
10     novo->info = x;
11     novo->ant = NULL;
12     novo->prox = NULL;
13
14     if (listaVazia(li)) {
15         *li = novo;
16     } else {
17         NO i;
18         i.prox = *li;
19
20         NO* p = &i;
21         while (p->prox != NULL && p->prox->info < x) {
22             p = p->prox;
23         }
24
25         if (p->prox == *li) {
26             novo->prox = *li;
27             *li = novo;
28         } else {
29             novo->ant = p;
30         }
31         novo->prox = p->prox;
32         p->prox = novo;
33     }
34
35     return 1;
36 }
37
38 int main() {
39     Lista* l = criaLista();
40
41     insereOrdenado(l, 23);
42     imprimeLista(l);
43     insereOrdenado(l, 14);
44     imprimeLista(l);
45     insereOrdenado(l, 7);
46     imprimeLista(l);
47     insereOrdenado(l, 85);
48     imprimeLista(l);
49     insereOrdenado(l, 19);
50     imprimeLista(l);
51     insereOrdenado(l, 56);
52     imprimeLista(l);
53
54     destroiLista(l);
55
56     return 0;
57 }
```


Terminal

```
gabriel.meira.2004@lab-02-07-03:~/Desktop/ufsj/UFSJ-LabProg2/roteiro-04$ ./ex03-03
Elementos:
23
Elementos:
14 23
Elementos:
7 14 23
Elementos:
7 14 23 85
Elementos:
7 14 19 23 85
Elementos:
7 14 19 23 56 85
gabriel.meira.2004@lab-02-07-03:~/Desktop/ufsj/UFSJ-LabProg2/roteiro-04$
```

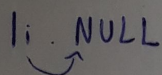
roteiro-04/ex03-04.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  #include "lista_dup_enc.h"
5
6  int removePrimeiroX(Lista* li, int x) {
7      if (li == NULL || listaVazia(li)) return 0;
8
9      NO i; // No inicial auxiliar
10     i.prox = *li;
11
12     NO* p = &i;
13     while (p->prox != NULL) {
14         if (p->prox->info == x) {
15             NO* aux = p->prox;
16             p->prox = aux->prox;
17             p->prox->ant = aux->ant;
18             liberarNO(aux);
19             break;
20         }
21
22         p = p->prox;
23     }
24
25     return 1;
26 }
27
28 int main() {
29     Lista* l = criaLista();
30
31     insereFim(l, 23);
32     insereFim(l, 14);
33     insereFim(l, 7);
34     insereFim(l, 85);
35     insereFim(l, 19);
36     insereFim(l, 56);
37
38     imprimeLista(l);
39     removePrimeiroX(l, 7);
40     imprimeLista(l);
41
42     destroiLista(l);
43
44     return 0;
45 }
```

```
gabriel.meira.2004@lab-02-07-03:~/Desktop/ufsj/UFSJ-LabProg2/roteiro-04$ ./ex03-04
Elementos:
23 14 7 85 19 56
Elementos:
23 14 85 19 56
gabriel.meira.2004@lab-02-07-03:~/Desktop/ufsj/UFSJ-LabProg2/roteiro-04$
```

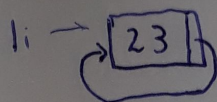
4.1) Lista Circular Simplesmente Encadeada

I) Lista vazia



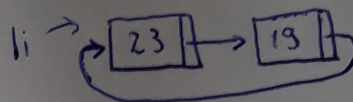
- cria o ponteiro "li" para NULL

II) Inserção (23)



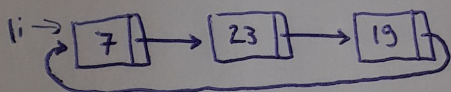
- cria o novo nó (23)
- o próximo de "23" é ele mesmo
- "li" aponta para "23"

III) Inserção Fim (19)



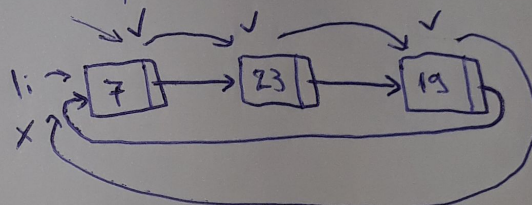
- cria o novo nó (19)
- o próximo de "19" é o nó apontado por "li"
- o último nó, que aponta para "li" passa a apontar para o "19".

IV) Inserção Início (7)



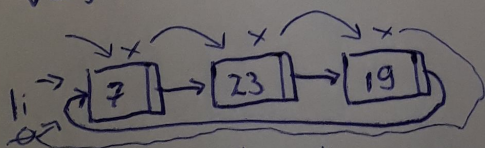
- cria o novo nó (7)
- o próximo de "7" será "li"
- o último nó, que aponta para "li" passa a apontar para "7"
- "li" aponta para "7"

V) Imprimir



- caso a lista esteja vazia não imprime
- percorre a lista imprimindo os dados de cada nó até que "li" seja visitado pela segunda vez (fim da lista)

VI) Liberar Lista



{ Del: 7, Del: 23, Del: 19 }

- caso a lista esteja vazia, libera a lista
- caso haja elementos, percorre a lista deletando cada nó até que "li" seja visitado pela segunda vez

roteiro-04/lista_cir_sim_enc.h

```
1  /*----- File: LCSE.h -----+
2  |Lista Circular Simplesmente Encadeada |
3  | | |
4  | | |
5  | Implementado por Guilherme C. Pena em 19/09/2023 |
6  +-----+ */
7
8  #ifndef LCSE_H
9  #define LCSE_H
10
11 #include <stdio.h>
12 #include <stdlib.h>
13
14 typedef struct NO {
15     int info;
16     struct NO* prox;
17 } NO;
18
19 typedef struct NO* Lista;
20
21 Lista* criaLista() {
22     Lista* li;
23     li = (Lista*)malloc(sizeof(Lista));
24     if (li != NULL) {
25         *li = NULL;
26     }
27     return li;
28 }
29
30 int listaVazia(Lista* li) {
31     if (li == NULL) return 1;
32     if (*li == NULL) return 1; // True - Vazia!
33     return 0; // False - tem elemento!
34 }
35
36 NO* alocarNO() {
37     return (NO*)malloc(sizeof(NO));
38 }
39
40 void liberarNO(NO* q) {
41     free(q);
42 }
43
44 int insereIni(Lista* li, int elem) {
45     if (li == NULL) return 0;
46     NO* novo = alocarNO();
47     if (novo == NULL) return 0;
48     novo->info = elem;
49     if (listaVazia(li)) {
50         novo->prox = novo;
51         *li = novo;
52     } else {
53         NO* aux = *li;
54         while (aux->prox != (*li))
55             aux = aux->prox;
```



```

56     aux->prox = novo;
57     novo->prox = *li;
58     *li = novo;
59 }
60 return 1;
61 }
62
63 int insereFim(Lista* li, int elem) {
64     if (li == NULL) return 0;
65     NO* novo = alocarNO();
66     if (novo == NULL) return 0;
67     novo->info = elem;
68     if (listaVazia(li)) {
69         novo->prox = novo;
70         *li = novo;
71     } else {
72         NO* aux = *li;
73         while (aux->prox != (*li))
74             aux = aux->prox;
75         aux->prox = novo;
76         novo->prox = *li;
77     }
78     return 1;
79 }
80
81 int removeIni(Lista* li) {
82     if (li == NULL) return 0;
83     if (listaVazia(li)) return 0;
84     NO* prim = *li;
85     if (prim == prim->prox) {
86         // apenas 1 elemento
87         *li = NULL;
88     } else {
89         NO* aux = *li;
90         while (aux->prox != (*li))
91             aux = aux->prox;
92         aux->prox = (*li)->prox;
93         *li = (*li)->prox;
94     }
95     liberarNO(prim);
96     return 1;
97 }
98
99 int removeFim(Lista* li) {
100     if (li == NULL) return 0;
101     if (listaVazia(li)) return 0;
102     NO* aux = *li;
103     if (aux == aux->prox) {
104         // apenas 1 elemento
105         *li = NULL;
106     } else {
107         NO* ant;
108         while (aux->prox != (*li)) {
109             ant = aux; // anterior
110             aux = aux->prox;
111         }
112         ant->prox = *li;
113     }
114     liberarNO(aux);

```

```
115     return 1;
116 }
117
118 void imprimeLista(Lista* li) {
119     if (li == NULL) return;
120     if (listaVazia(li)) {
121         printf("Lista Vazia!\n");
122         return;
123     }
124     printf("Elementos:\n");
125     NO* aux = *li;
126     while (aux->prox != *li) {
127         printf("%d ", aux->info);
128         aux = aux->prox;
129     }
130     printf("%d ", aux->info);
131     printf("\n");
132 }
133
134 void destroiLista(Lista* li) {
135     if (li != NULL && (*li) != NULL) {
136         NO *prim, *aux;
137         prim = *li;
138         *li = (*li)->prox;
139         while ((*li) != prim) {
140             aux = *li;
141             *li = (*li)->prox;
142             liberarNO(aux);
143         }
144         liberarNO(prim);
145         free(li);
146     }
147 }
148
149 #endif
```

roteiro-04/ex04-02.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  #include "lista_cir_sim_enc.h"
5
6  int tamanho(Lista* li) {
7      int t = 0;
8      NO* p = *li;
9
10     while (p != *li || t == 0) {
11         p = p->prox;
12         t++;
13     }
14
15     return t;
16 }
17
18 int procura(Lista* li, int x) {
19     int i = 0;
20     NO* p = *li;
21     while (p != *li || i == 0) {
22         if (p->info == x) return i;
23         p = p->prox;
24         i++;
25     }
26
27     return -1;
28 }
29
30 int main() {
31     Lista* l = criaLista();
32
33     insereFim(l, 23);
34     insereFim(l, 14);
35     insereFim(l, 7);
36     insereFim(l, 85);
37     insereFim(l, 19);
38     insereFim(l, 56);
39
40     imprimeLista(l);
41     printf("Tamanho = %d\n", tamanho(l));
42     printf("Indice do 19 = %d\n", procura(l, 19));
43
44     destroiLista(l);
45
46     return 0;
47 }
```



```
gabriel.meira.2004@lab-02-07-03:~/Desktop/ufsj/UFSJ-LabProg2/roteiro-04$ ./ex04-02
Elementos:
23 14 7 85 19 56
Tamanho = 6
Indice do 19 = 4
gabriel.meira.2004@lab-02-07-03:~/Desktop/ufsj/UFSJ-LabProg2/roteiro-04$
```