

listaEncadeada.h

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  typedef int Item;
5  typedef struct no {
6      Item item;
7      struct no *prox;
8  } *Lista;
9
10 Lista no(Item x, Lista p) {
11     Lista n = malloc(sizeof(struct no));
12     n->item = x;
13     n->prox = p;
14     return n;
15 }
16
17 void exibe(Lista L) {
18     while (L != NULL) {
19         printf("%d\n", L->item);
20         L = L->prox;
21     }
22 }
23
24 void exibe2(Lista L) {
25     printf("[");
26
27     if (L == NULL) {
28         printf("]");
29     }
30
31     while (L != NULL) {
32         printf("%d", L->item);
33         if (L->prox != NULL) {
34             printf(",");
35         }
36         L = L->prox;
37     }
38
39     printf("]");
40 }
41
42 int tamanho(Lista L) {
43     int t = 0;
44     while (L) {
45         t++;
```

```
46     L = L->prox;
47 }
48 return t;
49 }
50
51 Lista aleatoria(int n, int m) {
52     Lista L = NULL;
53     while (n > 0) {
54         L = no(rand() % m, L);
55         n--;
56     }
57
58     return L;
59 }
60
61 void destroi(Lista *L) {
62     while (*L) {
63         Lista n = *L;
64         *L = n->prox;
65         free(n);
66     }
67 }
```

# Exercício 1

Completar e executar o programa exibido no material:

```
1  #include "listaEncadeada.h"
2  // Fiz um arquivo para conter todo o arquivo de construção de listas
3  int main(void) {
4      Lista I = no(3, no(1, no(5, NULL)));
5      exibe(I);
6      return 0;
7  }
```

Saída:

```
.../ListasEncadeadas/exercicios > ./output/1.criaExibe
3
1
5

.../ListasEncadeadas/exercicios >
NORMAL ➤ ≡ term:~/bin/bash
```

## Exercício 2

Alterar a função `exibe()` para que a saída seja `[3,1,5]`

```
1  #include "listaEncadeada.h"
2
3  int main(void) {
4      Lista I = no(3, no(1, no(5, NULL)));
5      exibe2(I);
6      return 0;
7  }
```

Saída:

```
.../ListasEncadeadas/exercicios > ./output/2.exibe2
[3,1,5]
```

## Exercício 3

completar o programa a seguir

```
1  #include "listaEncadeada.h"
2
3  int main(void) {
4      Lista I = no(3, no(1, no(5, NULL)));
5      exibe2(I);
6      printf("\nTamanho = %d\n", tamanho(I));
7      return 0;
8  }
```

Saída:

```
.../ListasEncadeadas/exercicios > gcc 3.tamanho.c -o output/3.tamanho
.../ListasEncadeadas/exercicios > ./output/3.tamanho
[3,1,5] = 3
```

## Exercício 4

Adicionar uma função que some os itens da lista

```
1  #include "listaEncadeada.h"
2  #include <stdio.h>
3
4  int soma(Lista L) {
5      int result = 0;
6
7      while (L != NULL) {
8          result = result + L->item;
9          L = L->prox;
10     }
11     return result;
12 }
13
14 int main(void) {
15     Lista I = no(3, no(1, no(5, NULL)));
16     exibe2(I);
17     if (soma(I) == 0) {
18         printf("lista vazia");
19         return 0;
20     }
21     printf("\nResultado da soma dos itens da lista é: %d\n", soma(I));
22 }
```

Saída:

```
.../ListasEncadeadas/exercicios > gcc 4.soma.c -o output/4.soma
.../ListasEncadeadas/exercicios > ./output/4.soma
[3,1,5]
Resultado da soma dos itens da lista é: 9
.../ListasEncadeadas/exercicios >
TERMINAL > term:~/bin/bash > f soma
```

## Exercício 5

```
1  #include "listaEncadeada.h"
2  #include <stdlib.h>
3  #include <time.h>
4
5  Lista aleatoria(int n, int m) {
6      Lista L = NULL;
7      while (n > 0) {
8          L = no(rand() % m, L);
9          n--;
10     }
11     return L;
12 }
13
14 int main(void) {
15     srand(time(NULL));
16     Lista A = aleatoria(10, 100);
17     exhibe2(A);
18     return 0;
19 }
```

Saída:

```
.../ListasEncadeadas/exercicios > gcc 5.aleatoria.c -o output/5.aleatoria
.../ListasEncadeadas/exercicios > ./output/5.aleatoria
[41,35,47,93,39,91,41,65,72,46]
```

## Exercício 6

```
1  #include "listaEncadeada.h"
2
3  Lista intervalo(int n) {
4      Lista L = NULL;
5      for (int i = n; i > 0; i--) {
6          L = no(i, L);
7      }
8
9      return L;
10 }
11
12 int main(void) {
13     int n;
14     printf("Até quanto devo contar? ");
15     scanf("%d", &n);
16
17     printf("Contando... \n");
18     exhibe(intervalo(n));
19     return 0;
20 }
```

Saída:

```
.../ListasEncadeadas/exercicios > ./output/6.listIntervalo
Até quanto devo contar? 10
Contando...
1
2
3
4
5
6
7
8
9
10
```



# Exercício 7

```

1  #include "listaEncadeada.h"
2  #include <stdio.h>
3
4  void anexa(Lista *A, Lista B) {
5      if (!B)
6          return;
7
8      while (*A)
9          A = &(*A)->prox;
10     *A = B;
11 }
12
13 int main(void) {
14     Lista H = no(4, no(2, NULL));
15     Lista I = no(3, no(1, no(5, NULL)));
16
17     printf("H = ");
18     exibe2(H);
19     printf("\nI = ");
20     exibe2(I);
21     printf("\nPressione enter");
22
23     getchar();
24
25     anexa(&H, I);
26     printf("H = ");
27     exibe2(H);
28     printf("\nI = ");
29     exibe2(I);
30
31     return 0;
32 }

```

Saída:

```

.../ListasEncadeadas/exercicios > ./output/7.anexacao
H = [4,2]
I = [3,1,5]
Pressione enter
H = [4,2,3,1,5]
I = [3,1,5]

```



## Exercício 8

```
1  #include "listaEncadeada.h"
2  #include <stdio.h>
3  #include <time.h>
4
5  Lista ultimo(Lista L) {
6      if (L == NULL)
7          return NULL;
8
9      while (L->prox != NULL) {
10         L = L->prox;
11     }
12
13     return L;
14 }
15
16 int main(void) {
17     srand(time(NULL));
18     Lista A = aleatoria(10, 100);
19     Lista final = ultimo(A);
20
21     if (final == NULL) {
22         printf("ERROR FATAL");
23         return 0;
24     }
25     printf("O ultimo item da lista é: ");
26     exibe(final);
27     exibe2(A);
28     destroi(&A);
29     return 0;
30 }
```

Saída:

```
.../ListasEncadeadas/exercicios > ./output/8.ultimo
O ultimo item da lista é: 41
[40,51,68,13,5,32,3,60,46,41]
.../ListasEncadeadas/exercicios >
```

## Exercício 9

```
1  #include "listaEncadeada.h"
2  #include <stdio.h>
3  #include <time.h>
4
5  int maximo(Lista L) {
6      int max = L->item;
7
8      if (L == NULL)
9          return -1;
10
11     while (L != NULL) {
12         if (max < L->item) {
13             max = L->item;
14         } else {
15             L = L->prox;
16         }
17     }
18
19     return max;
20 }
21
22 int main(void) {
23     Lista A = aleatoria(10, 100);
24     int max = maximo(A);
25
26     if (max == -1) {
27         printf("FATAL ERROR");
28         return 0;
29     }
30     exibe2(A);
31     printf("\nO maior item na lista é: %d", max);
32 }
```

Saída

```
.../ListasEncadeadas/exercicios > ./output/9.maximo
[21,49,92,86,35,93,15,77,86,83]
O maior item na lista é: 93
```

## Exercício 10

```
1  #include "listaEncadeada.h"
2  #include <stdio.h>
3  #include <time.h>
4
5  int pertence(int x, Lista L) {
6      if (L == NULL) {
7          return 0;
8      }
9
10     while (L != NULL) {
11         if (x == L->item) {
12             return 1;
13         }
14         L = L->prox;
15     }
16     return 0;
17 }
18
19 int main(void) {
20     srand(time(NULL));
21     Lista A = aleatoria(10, 100);
22     int num;
23
24     printf("Qual o número? ");
25     scanf("%d", &num);
26
27     int pert = pertence(num, A);
28
29     exibe2(A);
30     if (pert == 1) {
31         printf("\nNúmero %d pertence à lista.", num);
32         return 0;
33     }
34     printf("\nNúmero %d não pertence a lista", num);
35     return 0;
36 }
```

Saída:

```
.../ListasEncadeadas/exercicios ) ./output/10.pertinencia
Qual o número? 10
[23,96,11,4,97,42,43,8,41,71]
Número 10 não pertence a lista
.../ListasEncadeadas/exercicios ) ./output/10.pertinencia
Qual o número? 10
[40,29,97,10,37,61,86,0,76,65]
Número 10 pertence à lista.
```

# Exercício 11

```
1  #include "listaEncadeada.h"
2  #include <stdio.h>
3
4  Lista inversa(Lista L) {
5      Lista R = NULL;
6
7      while (L != NULL) {
8          R = no(L->item, R);
9          L = L->prox;
10     }
11     return R;
12 }
13
14 int main(void) {
15     Lista A = no(1, no(2, no(3, no(4, NULL))));
16     exibe2(A);
17     printf("\n");
18     Lista B = inversa(A);
19     exibe2(B);
20
21     destroi(&B);
22     destroi(&A);
23     return 0;
24 }
```

Saída:

```
.../ListasEncadeadas/exercicios > ./output/11.inversao
[1,2,3,4]
[4,3,2,1]
```

## Exercício 12

```
1  #include "listaEncadeada.h"
2  #include <stdio.h>
3  #include <time.h>
4
5  Lista intervalo(int part, int final) {
6      Lista L = NULL;
7
8      if (part > final) {
9          return L;
10     }
11
12     for (int i = final; i >= part; i--) {
13         L = no(i, L);
14     }
15
16     return L;
17 }
18
19 int main(void) {
20     int part, final;
21
22     printf("Defina o intervalo da contagem.\n");
23     scanf("%d %d", &part, &final);
24
25     Lista L = intervalo(part, final);
26
27     printf("A contagem: ");
28     exhibe2(L);
29
30     destroi(&L);
31     return 0;
32 }
```



Saída:

```
.../ListasEncadeadas/exercicios › ./output/12.intervalo  
Defina o intervalo da contagem.  
5 3  
A contagem: []  
.../ListasEncadeadas/exercicios › ./output/12.intervalo  
Defina o intervalo da contagem.  
-2 3  
A contagem: [-2,-1,0,1,2,3]
```

## Exercício 13

```
1  #include "listaEncadeada.h"
2  #include <stdio.h>
3  #include <time.h>
4
5  int len(Lista L) {
6      if (L == NULL) {
7          return 0;
8      }
9
10     return 1 + len(L->prox);
11 }
12
13 int main(void) {
14     srand(time(NULL));
15     Lista L = aleatoria(10, 100);
16
17     printf("O tamanho da lista é: %d\n", len(L));
18     exibe2(L);
19
20     return 0;
21 }
```

Saída:

```
.../ListasEncadeadas/exercicios > ./output/13.tamanho
O tamanho da lista é: 10
[31,38,18,5,76,53,71,94,53,35]
```

## Exercício 14

```

1  #include "listaEncadeada.h"
2  #include <stdio.h>
3  #include <time.h>
4
5  int sum(Lista L) {
6      if (L == NULL) {
7          return 0;
8      }
9      return L->item + sum(L->prox);
10 }
11
12 int main(void) {
13     srand(time(NULL));
14     Lista L = aleatoria(5, 10);
15
16     printf("A soma de todos os números da lista é: %d\n", sum(L));
17     exibe2(L);
18
19     destroi(&L);
20     return 0;
21 }

```

Saída:

```

.../ListasEncadeadas/exercicios > ./output/14.soma
A soma de todos os números da lista é: 25
[7,5,0,9,4]
.../ListasEncadeadas/exercicios > gcc 14.soma.c -o output/14.soma

.../ListasEncadeadas/exercicios > ./output/14.soma
A soma de todos os números da lista é: 30
[8,7,8,0,7]
.../ListasEncadeadas/exercicios > gcc 14.soma.c -o output/14.soma

.../ListasEncadeadas/exercicios > ./output/14.soma
A soma de todos os números da lista é: 29
[9,7,4,4,5]
.../ListasEncadeadas/exercicios > gcc 14.soma.c -o output/14.soma

.../ListasEncadeadas/exercicios > ./output/14.soma
A soma de todos os números da lista é: 26
[9,6,3,8,0]

```



## Exercício 15

```
1  #include "listaEncadeada.h"
2  #include <stdio.h>
3  #include <time.h>
4
5  Lista clone(Lista L) {
6      if (L == NULL)
7          return NULL;
8
9      return no(L->item, clone(L->prox));
10 }
11
12 int main(void) {
13     srand(time(NULL));
14     Lista L = aleatoria(5, 10);
15
16     printf("A lista L = ");
17     exibe2(L);
18     printf("\no Clone da Lista L = ");
19     Lista C = clone(L);
20     exibe2(C);
21
22     destroi(&L);
23     destroi(&C);
24     return 0;
25 }
```

Saída:

```
.../ListasEncadeadas/exercicios > ./output/15.clone
A lista L = [3,3,6,3,3]
o Clone da Lista L = [3,3,6,3,3]
.../ListasEncadeadas/exercicios > gcc 15.clone.c -o output/15.clone

.../ListasEncadeadas/exercicios > ./output/15.clone
A lista L = [6,9,9,9,0]
o Clone da Lista L = [6,9,9,9,0]
```

## Exercício 16

```
1  #include "listaEncadeada.h"
2  #include <stdio.h>
3  #include <stdlib.h>
4  #include <time.h>
5
6  Lista rnd(int n, int m) {
7      if (n == 0) {
8          return NULL;
9      }
10
11     int valor = rand() % m;
12     return no(valor, rnd(n - 1, m));
13 }
14
15 int main(void) {
16     srand(time(NULL));
17
18     int n = 5, m = 10;
19     Lista L = rnd(n, m);
20
21     printf("Lista aleatória com %d itens em [0, %d]: ", n, m);
22     exibe2(L);
23
24     destroi(&L);
25     return 0;
26 }
```

Saída:

```
.../ListasEncadeadas/exercicios > ./output/16.aleatória
Lista aleatória com 5 itens em [0, 10]: [1,1,2,7,3]
```

## Exercício 17

```
1  #include "listaEncadeada.h"
2  #include <stdio.h>
3  #include <stdlib.h>
4  #include <time.h>
5
6  int last(Lista L) {
7      if (L == NULL)
8          return -1;
9
10     if (L->prox == NULL) {
11         int ultimo = L->item;
12         return ultimo;
13     }
14
15     return last(L->prox);
16 }
17
18 int main(void) {
19     srand(time(NULL));
20
21     Lista L = aleatoria(5, 10);
22     printf("O ultimo item é: %d\n", last(L));
23     exibe2(L);
24
25     destroi(&L);
26     return 0;
27 }
```

Saída:

```
.../ListasEncadeadas/exercicios ) ./output/17.ultimo
O ultimo item é: 3
[1,8,5,1,3]
```

## Exercício 18

```
1  #include "listaEncadeada.h"
2  #include <stdio.h>
3  #include <time.h>
4
5  int in(int x, Lista L) {
6      if (L == NULL)
7          return 0;
8
9      if (x == L->item) {
10         return 1;
11     }
12     return in(x, L->prox);
13 }
14
15 int main(void) {
16     srand(time(NULL));
17     Lista A = aleatoria(10, 20);
18     int num;
19
20     printf("Qual o número? ");
21     scanf("%d", &num);
22
23     int pert = in(num, A);
24
25     exibe2(A);
26     if (pert == 1) {
27         printf("\nNúmero %d pertence à lista.", num);
28         return 0;
29     }
30     printf("\nNúmero %d não pertence a lista", num);
31     return 0;
32 }
```

Saída:



```
.../ListasEncadeadas/exercicios › ./output/18.pertinencia
Qual o número? 15
[15,12,6,11,15,16,11,16,9,9]
Número 15 pertence à lista.
.../ListasEncadeadas/exercicios › ./output/18.pertinencia
Qual o número? 15
[12,2,9,3,1,18,0,9,13,18]
Número 15 não pertence a lista
```

## Exercício 19

```

1  #include "listaEncadeada.h"
2  #include <stdio.h>
3  #include <time.h>
4
5  int enesimo(Lista L, int n) {
6      if (L == NULL)
7          return -1; // ou valor sentinela
8      if (n == 1)
9          return L->item;
10     return enesimo(L->prox, n - 1);
11 }
12
13 int main(void) {
14     srand(time(NULL));
15     Lista L = aleatoria(10, 20);
16
17     int num;
18
19     printf("Qual a posição deseja checar? ");
20     scanf("%d", &num);
21
22     int nesimo = enesimo(L, num);
23
24     printf("O valor na posição %d é %d\n", num, nesimo);
25     exibe2(L);
26
27     destroi(&L);
28     return 0;
29 }

```

Saída:

```

.../ListasEncadeadas/exercicios > ./output/19.enesimo
Qual a posição deseja checar? 10
O valor na posição 10 é 6
[0,13,14,15,1,3,1,6,19,6]
.../ListasEncadeadas/exercicios > ./output/19.enesimo
Qual a posição deseja checar? 3
O valor na posição 3 é 17
[0,0,17,16,0,14,4,2,17,8]

```



## Exercício 20

```
1  #include "listaEncadeada.h"
2  #include <stdio.h>
3  #include <time.h>
4
5  int minimum(Lista L) {
6      if (L == NULL)
7          return -1;
8
9      if (L->prox == NULL)
10         return L->item;
11
12     int minResto = minimum(L->prox);
13
14     return (L->item < minResto) ? L->item : minResto;
15 }
16
17 int main(void) {
18     srand(time(NULL));
19     Lista L = aleatoria(5, 10);
20
21     printf("Lista: ");
22     exibe2(L);
23
24     printf("\nO menor número da lista é: %d\n", minimum(L));
25
26     destroi(&L);
27     return 0;
28 }
```

### Saída

```
.../ListasEncadeadas/exercicios > ./output/20.minimo
Lista: [5,3,5,2,5]
O menor número da lista é: 2
```

# Exercício 21

```
1  #include "listaEncadeada.h"
2  #include <stdio.h>
3  #include <time.h>
4
5  void divide(Lista L, Lista *metade1, Lista *metade2) {
6      if (L == NULL || L->prox == NULL) {
7          *metade1 = L;
8          *metade2 = NULL;
9          return;
10     }
11
12     Lista lento = L;
13     Lista rapido = L->prox;
14
15     while (rapido != NULL) {
16         rapido = rapido->prox;
17         if (rapido != NULL) {
18             lento = lento->prox;
19             rapido = rapido->prox;
20         }
21     }
22
23     *metade1 = L;
24     *metade2 = lento->prox;
25     lento->prox = NULL;
26 }
27
28 Lista intercalar(Lista a, Lista b) {
29     if (a == NULL)
30         return b;
31     if (b == NULL)
32         return a;
33
34     Lista resultado = NULL;
35
36     if (a->item <= b->item) {
37         resultado = a;
38         resultado->prox = intercalar(a->prox, b);
39     } else {
40         resultado = b;
41         resultado->prox = intercalar(a, b->prox);
42     }
43
44     return resultado;
45 }
46
```

```
47 Lista ordenar(Lista L) {
48     if (L == NULL || L->prox == NULL)
49         return L;
50
51     Lista metade1, metade2;
52
53     divide(L, &metade1, &metade2);
54
55     metade1 = ordenar(metade1);
56     metade2 = ordenar(metade2);
57
58     return intercalar(metade1, metade2);
59 }
60
61 int main() {
62     srand(time(NULL));
63     Lista L = aleatoria(10, 100);
64
65     printf("A lista atual é: ");
66     exhibe2(L);
67
68     printf("\nA lista ordenada: ");
69     Lista A = ordenar(L);
70     exhibe2(A);
71
72     destroi(&L);
73     return 0;
74 }
```

Saída:

```
.../ListasEncadeadas/exercicios > ./output/21.ordenada
A lista atual é: [44,1,87,66,16,90,47,79,27,52]
A lista ordenada: [1,16,27,44,47,52,66,79,87,90]
```

## Exercício 22

```
1  #include "listaEncadeada.h"
2  #include <stdio.h>
3  #include <time.h>
4
5  int equals(Lista A, Lista B) {
6      if (A == NULL && B == NULL)
7          return 1;
8
9      if (A == NULL || B == NULL)
10         return 0;
11
12     if (A->item != B->item)
13         return 0;
14
15     return equals(A->prox, B->prox);
16 }
17
18 int main(void) {
19     // srand(time(NULL));
20     Lista L1 = no(1, no(2, no(3, NULL)));
21     Lista L2 = no(1, no(2, no(3, NULL)));
22
23     printf("Lista 1: ");
24     exibe2(L1);
25
26     printf("\nLista 2: ");
27     exibe2(L2);
28
29     if (equals(L1, L2))
30         printf("\nAs listas são iguais");
31     else
32         printf("\nAs listas são diferentes");
33
34     destroi(&L1);
35     destroi(&L2);
36
37     return 0;
38 }
```

Saída:

```
.../ListasEncadeadas/exercicios › ./output/22.igualdade
Lista 1: [3,5,5,5,4]
Lista 2: [3,0,7,4,1]
As listas são diferentes
.../ListasEncadeadas/exercicios › gcc 22.igualdade.c -o output/22.igualdade

.../ListasEncadeadas/exercicios › ./output/22.igualdade
Lista 1: [1,2,3]
Lista 2: [1,2,3]
As listas são iguais
```



## Exercício 23

```
1  #include "listaEncadeada.h"
2  #include <stdio.h>
3  #include <time.h>
4
5  int count(int x, Lista L) {
6      if (L == NULL)
7          return 0;
8
9      if (x == L->item) {
10         return 1 + count(x, L->prox);
11     }
12
13     return count(x, L->prox);
14 }
15
16 int main() {
17     srand(time(NULL));
18     Lista L = aleatoria(20, 10);
19
20     int num;
21
22     printf("Qual o número quer procurar? ");
23     scanf("%d", &num);
24
25     int conta = count(num, L);
26
27     printf("
28         A quantidade que o item %d aparece na lista é: %d.
29         \nLista: ",
30         num,
31         conta
32     );
33     exhibe2(L);
34
35     destroi(&L);
36     return 0;
37 }
```

Saída:

```
.../ListasEncadeadas/exercicios ) ./output/23.contagem
Qual o número quer procurar? 3
A quantidade que o item 3 aparece na lista é: 2.
Lista: [0,8,5,9,2,4,3,5,9,9,0,2,8,5,4,3,0,8,4,2]
.../ListasEncadeadas/exercicios ) ./output/23.contagem
Qual o número quer procurar? 3
A quantidade que o item 3 aparece na lista é: 4.
Lista: [4,7,9,9,9,3,7,3,7,3,9,8,2,0,5,8,5,1,4,3]
```

## Exercício 24

```
1  #include "listaEncadeada.h"
2  #include <stdio.h>
3  #include <time.h>
4
5  void replace(int x, int y, Lista L) {
6      if (L == NULL)
7          return;
8
9      if (L->item == x) {
10         L->item = y;
11     }
12
13     return replace(x, y, L->prox);
14 }
15
16 int main() {
17     srand(time(NULL));
18     Lista L = aleatoria(15, 8);
19
20     exibe2(L);
21
22     int num1, num2;
23
24     printf("\nQual o número quer alterar? ");
25     scanf("%d", &num1);
26     printf("Qual o número quer colocar no lugar? ");
27     scanf("%d", &num2);
28
29     printf("Nova lista: ");
30     replace(num1, num2, L);
31     exibe2(L);
32
33     destroi(&L);
34     return 0;
35 }
```

Saída:

```
.../ListasEncadeadas/exercicios > ./output/24.substituicao  
[1,0,5,5,4,2,0,6,6,6,5,1,1,6,1]  
Qual o número quer alterar? 6  
Qual o número quer colocar no lugar? 10  
Nova lista: [1,0,5,5,4,2,0,10,10,10,5,1,1,10,1]
```