`listaEncadeada.h`

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

typedef int Item;
typedef struct no {
  Item item;
  struct no *prox;
} *Lista;

Lista no(Item x, Lista p) {
  Lista n = malloc(sizeof(struct no));
  n->item = x;
  n->prox = p;
  return n;
}

void exibe(Lista L) {
  while (L != NULL) {
    printf("%d\n", L->item);
    L = L->prox;
  }
}

void exibe2(Lista L) {
  printf("[");

  if (L == NULL) {
    printf("]");
  }

  while (L != NULL) {
    printf("%d", L->item);
    if (L->prox != NULL) {
      printf(",");
    }
    L = L->prox;
  }

  printf("]");
}

int tamanho(Lista L) {
  int t = 0;
  while (L) {
    t++;
```

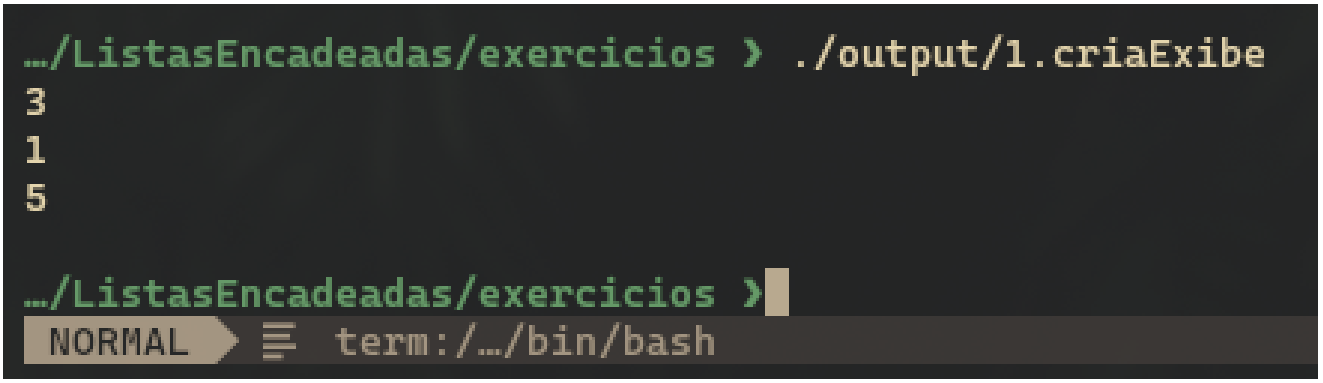
```c
    L = L->prox;
  }
  return t;
}

Lista aleatoria(int n, int m) {
  Lista L = NULL;
  while (n > 0) {
    L = no(rand() % m, L);
    n--;
  }

  return L;
}

void destroi(Lista *L) {
  while (*L) {
    Lista n = *L;
    *L = n->prox;
    free(n);
  }
}
```

Exercício 1

Completar e executar o programa exibido no material:

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

Saída:



Exercício 2

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

```c
#include "listaEncadeada.h"

int main(void) {
  Lista I = no(3, no(1, no(5, NULL)));
  exibe2(I);
  return 0;
}
```

Saída:

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

Exercício 3

completar o programa a seguir

```c
#include "listaEncadeada.h"

int main(void) {
  Lista I = no(3, no(1, no(5, NULL)));
  exibe2(I);
  printf("\nTamanho = %d\n", tamanho(I));
  return 0;
}
```

Saída:

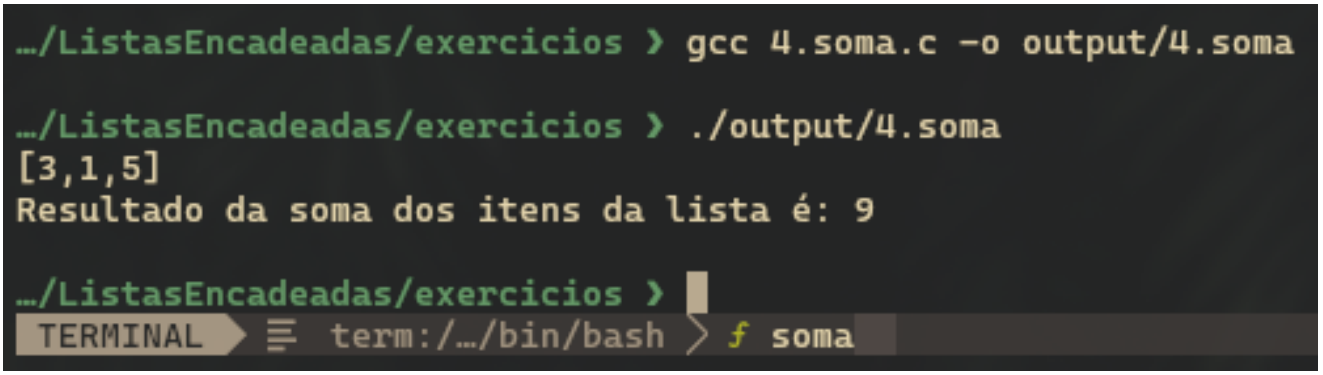
```
…/ListasEncadeadas/exercicios ❯ gcc 3.tamanho.c -o output/3.tamanho

…/ListasEncadeadas/exercicios ❯ ./output/3.tamanho
[3,1,5] = 3
```

Exercício 4

Adicionar uma função que some os itens da lista

```c
#include "listaEncadeada.h"
#include <stdio.h>

int soma(Lista L) {
  int result = 0;

  while (L != NULL) {
    result = result + L->item;
    L = L->prox;
  }
  return result;
}

int main(void) {
  Lista I = no(3, no(1, no(5, NULL)));
  exibe2(I);
  if (soma(I) == 0) {
    printf("lista vazia");
    return 0;
  }
  printf("\nResultado da soma dos itens da lista é: %d\n", soma(I));
}
```

Saída:



Exercício 5

```c
#include "listaEncadeada.h"
#include <stdlib.h>
#include <time.h>

Lista aleatoria(int n, int m) {
  Lista L = NULL;
  while (n > 0) {
    L = no(rand() % m, L);
    n--;
  }
  return L;
}

int main(void) {
  srand(time(NULL));
  Lista A = aleatoria(10, 100);
  exibe2(A);
  return 0;
}
```

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

```c
#include "listaEncadeada.h"

Lista intervalo(int n) {
  Lista L = NULL;
  for (int i = n; i > 0; i--) {
    L = no(i, L);
  }

  return L;
}

int main(void) {
  int n;
  printf("Até quanto devo contar? ");
  scanf("%d", &n);

  printf("Contando... \n");
  exibe(intervalo(n));
  return 0;
}
```

Saída:

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

Exercício 7

```c
#include "listaEncadeada.h"
#include <stdio.h>

void anexa(Lista *A, Lista B) {
  if (!B)
    return;

  while (*A)
    A = &(*A)->prox;
  *A = B;
}

int main(void) {
  Lista H = no(4, no(2, NULL));
  Lista I = no(3, no(1, no(5, NULL)));

  printf("H = ");
  exibe2(H);
  printf("\nI = ");
  exibe2(I);
  printf("\nPressione enter");

  getchar();

  anexa(&H, I);
  printf("H = ");
  exibe2(H);
  printf("\nI = ");
  exibe2(I);

  return 0;
}
```

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

```c
#include "listaEncadeada.h"
#include <stdio.h>
#include <time.h>

Lista ultimo(Lista L) {
  if (L == NULL)
    return NULL;

  while (L->prox != NULL) {
    L = L->prox;
  }

  return L;
}

int main(void) {
  srand(time(NULL));
  Lista A = aleatoria(10, 100);
  Lista final = ultimo(A);

  if (final == NULL) {
    printf("ERROR FATAL");
    return 0;
  }
  printf("O ultimo item da lista é: ");
  exibe(final);
  exibe2(A);
  destroi(&A);
  return 0;
}
```

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

```c
#include "listaEncadeada.h"
#include <stdio.h>
#include <time.h>

int maximo(Lista L) {
  int max = L->item;

  if (L == NULL)
    return -1;

  while (L != NULL) {
    if (max < L->item) {
      max = L->item;
    } else {
      L = L->prox;
    }
  }

  return max;
}

int main(void) {
  Lista A = aleatoria(10, 100);
  int max = maximo(A);

  if (max == -1) {
    printf("FATAL ERROR");
    return 0;
  }
  exibe2(A);
  printf("\nO maior item na lista é: %d", max);
}
```

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

```c
#include "listaEncadeada.h"
#include <stdio.h>
#include <time.h>

int pertence(int x, Lista L) {
  if (L == NULL) {
    return 0;
  }

  while (L != NULL) {
    if (x == L->item) {
      return 1;
    }
    L = L->prox;
  }
  return 0;
}

int main(void) {
  srand(time(NULL));
  Lista A = aleatoria(10, 100);
  int num;

  printf("Qual o número? ");
  scanf("%d", &num);

  int pert = pertence(num, A);

  exibe2(A);
  if (pert == 1) {
    printf("\nNúmero %d pertence à lista.", num);
    return 0;
  }
  printf("\nNúmero %d não pertence a lista", num);
  return 0;
}
```

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

```c
#include "listaEncadeada.h"
#include <stdio.h>

Lista inversa(Lista L) {
  Lista R = NULL;

  while (L != NULL) {
    R = no(L->item, R);
    L = L->prox;
  }
  return R;
}

int main(void) {
  Lista A = no(1, no(2, no(3, no(4, NULL))));
  exibe2(A);
  printf("\n");
  Lista B = inversa(A);
  exibe2(B);

  destroi(&B);
  destroi(&A);
  return 0;
}
```

Saída:

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

Exercício 12

```c
#include "listaEncadeada.h"
#include <stdio.h>
#include <time.h>

Lista intervalo(int part, int final) {
  Lista L = NULL;

  if (part > final) {
    return L;
  }

  for (int i = final; i >= part; i--) {
    L = no(i, L);
  }

  return L;
}

int main(void) {
  int part, final;

  printf("Defina o intervalo da contagem.\n");
  scanf("%d %d", &part, &final);

  Lista L = intervalo(part, final);

  printf("A contagem: ");
  exibe2(L);

  destroi(&L);
  return 0;
}
```

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

```c
#include "listaEncadeada.h"
#include <stdio.h>
#include <time.h>

int len(Lista L) {
  if (L == NULL) {
    return 0;
  }

  return 1 + len(L->prox);
}

int main(void) {
  srand(time(NULL));
  Lista L = aleatoria(10, 100);

  printf("O tamanho da lista é: %d\n", len(L));
  exibe2(L);

  return 0;
}
```

Saída:

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

Exercício 14

```c
#include "listaEncadeada.h"
#include <stdio.h>
#include <time.h>

int sum(Lista L) {
  if (L == NULL) {
    return 0;
  }
  return L->item + sum(L->prox);
}

int main(void) {
  srand(time(NULL));
  Lista L = aleatoria(5, 10);

  printf("A soma de todos os números da lista é: %d\n", sum(L));
  exibe2(L);

  destroi(&L);
  return 0;
}
```

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

```c
#include "listaEncadeada.h"
#include <stdio.h>
#include <time.h>

Lista clone(Lista L) {
  if (L == NULL)
    return NULL;

  return no(L->item, clone(L->prox));
}

int main(void) {
  srand(time(NULL));
  Lista L = aleatoria(5, 10);

  printf("A lista L = ");
  exibe2(L);
  printf("\no Clone da Lista L = ");
  Lista C = clone(L);
  exibe2(C);

  destroi(&L);
  destroi(&C);
  return 0;
}
```

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

```c
#include "listaEncadeada.h"
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

Lista rnd(int n, int m) {
  if (n == 0) {
    return NULL;
  }

  int valor = rand() % m;
  return no(valor, rnd(n - 1, m));
}

int main(void) {
  srand(time(NULL));

  int n = 5, m = 10;
  Lista L = rnd(n, m);

  printf("Lista aleatória com %d itens em [0, %d]: ", n, m);
  exibe2(L);

  destroi(&L);
  return 0;
}
```

Saída:

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

Exercício 17

```c
#include "listaEncadeada.h"
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int last(Lista L) {
  if (L == NULL)
    return -1;

  if (L->prox == NULL) {
    int ultimo = L->item;
    return ultimo;
  }

  return last(L->prox);
}

int main(void) {
  srand(time(NULL));

  Lista L = aleatoria(5, 10);
  printf("O ultimo item é: %d\n", last(L));
  exibe2(L);

  destroi(&L);
  return 0;
}
```

Saída:

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

Exercício 18

```c
#include "listaEncadeada.h"
#include <stdio.h>
#include <time.h>

int in(int x, Lista L) {
  if (L == NULL)
    return 0;

  if (x == L->item) {
    return 1;
  }
  return in(x, L->prox);
}

int main(void) {
  srand(time(NULL));
  Lista A = aleatoria(10, 20);
  int num;

  printf("Qual o número? ");
  scanf("%d", &num);

  int pert = in(num, A);

  exibe2(A);
  if (pert == 1) {
    printf("\nNúmero %d pertence à lista.", num);
    return 0;
  }
  printf("\nNúmero %d não pertence a lista", num);
  return 0;
}
```

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

```c
#include "listaEncadeada.h"
#include <stdio.h>
#include <time.h>

int enesimo(Lista L, int n) {
  if (L == NULL)
    return -1; // ou valor sentinela
  if (n == 1)
    return L->item;
  return enesimo(L->prox, n - 1);
}

int main(void) {
  srand(time(NULL));
  Lista L = aleatoria(10, 20);

  int num;

  printf("Qual a posição deseja checar? ");
  scanf("%d", &num);

  int nesimo = enesimo(L, num);

  printf("O valor na posição %d é %d\n", num, nesimo);
  exibe2(L);

  destroi(&L);
  return 0;
}
```

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

```c
#include "listaEncadeada.h"
#include <stdio.h>
#include <time.h>

int minimum(Lista L) {
  if (L == NULL)
    return -1;

  if (L->prox == NULL)
    return L->item;

  int minResto = minimum(L->prox);

  return (L->item < minResto) ? L->item : minResto;
}

int main(void) {
  srand(time(NULL));
  Lista L = aleatoria(5, 10);

  printf("Lista: ");
  exibe2(L);

  printf("\nO menor número da lista é: %d\n", minimum(L));

  destroi(&L);
  return 0;
}
```

Saída

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

Exercício 21

```c
#include "listaEncadeada.h"
#include <stdio.h>
#include <time.h>

void divide(Lista L, Lista *metade1, Lista *metade2) {
  if (L == NULL || L->prox == NULL) {
    *metade1 = L;
    *metade2 = NULL;
    return;
  }

  Lista lento = L;
  Lista rapido = L->prox;

  while (rapido != NULL) {
    rapido = rapido->prox;
    if (rapido != NULL) {
      lento = lento->prox;
      rapido = rapido->prox;
    }
  }

  *metade1 = L;
  *metade2 = lento->prox;
  lento->prox = NULL;
}

Lista intercalar(Lista a, Lista b) {
  if (a == NULL)
    return b;
  if (b == NULL)
    return a;

  Lista resultado = NULL;

  if (a->item <= b->item) {
    resultado = a;
    resultado->prox = intercalar(a->prox, b);
  } else {
    resultado = b;
    resultado->prox = intercalar(a, b->prox);
  }

  return resultado;
}

```

```c
Lista ordenar(Lista L) {
  if (L == NULL || L->prox == NULL)
    return L;

  Lista metade1, metade2;

  divide(L, &metade1, &metade2);

  metade1 = ordenar(metade1);
  metade2 = ordenar(metade2);

  return intercalar(metade1, metade2);
}

int main() {
  srand(time(NULL));
  Lista L = aleatoria(10, 100);

  printf("A lista atual é: ");
  exibe2(L);

  printf("\nA lista ordenada: ");
  Lista A = ordenar(L);
  exibe2(A);

  destroi(&L);
  return 0;
}
```

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

```c
#include "listaEncadeada.h"
#include <stdio.h>
#include <time.h>

int equals(Lista A, Lista B) {
  if (A == NULL && B == NULL)
    return 1;

  if (A == NULL || B == NULL)
    return 0;

  if (A->item != B->item)
    return 0;

  return equals(A->prox, B->prox);
}

int main(void) {
  // srand(time(NULL));
  Lista L1 = no(1, no(2, no(3, NULL)));
  Lista L2 = no(1, no(2, no(3, NULL)));

  printf("Lista 1: ");
  exibe2(L1);

  printf("\nLista 2: ");
  exibe2(L2);

  if (equals(L1, L2))
    printf("\nAs listas são iguais");
  else
    printf("\nAs listas são diferentes");

  destroi(&L1);
  destroi(&L2);

  return 0;
}
```

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

```c
#include "listaEncadeada.h"
#include <stdio.h>
#include <time.h>

int count(int x, Lista L) {
  if (L == NULL)
    return 0;

  if (x == L->item) {
    return 1 + count(x, L->prox);
  }

  return count(x, L->prox);
}

int main() {
  srand(time(NULL));
  Lista L = aleatoria(20, 10);

  int num;

  printf("Qual o número quer procurar? ");
  scanf("%d", &num);

  int conta = count(num, L);

  printf("
      A quantidade que o item %d aparece na lista é: %d.
      \nLista: ",
      num,
      conta
  );
  exibe2(L);

  destroi(&L);
  return 0;
}
```

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

```c
#include "listaEncadeada.h"
#include <stdio.h>
#include <time.h>

void replace(int x, int y, Lista L) {
  if (L == NULL)
    return;

  if (L->item == x) {
    L->item = y;
  }

  return replace(x, y, L->prox);
}

int main() {
  srand(time(NULL));
  Lista L = aleatoria(15, 8);

  exibe2(L);

  int num1, num2;

  printf("\nQual o número quer alterar? ");
  scanf("%d", &num1);
  printf("Qual o número quer colocar no lugar? ");
  scanf("%d", &num2);

  printf("Nova lista: ");
  replace(num1, num2, L);
  exibe2(L);

  destroi(&L);
  return 0;
}
```

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]
```