

## Exercício 1

Ex1.c

```
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 int main(void) {
25     Lista I = no(3,no(1,no(5,NULL)));
26     exibe(I);
27     return 0;
28 }
29
```

```
3
1
5

-----
Process exited after 0.02066 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```

## Exercício 2

Ex2.c

```
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     printf("[");
19     if(L == NULL){
20
21     }else{
22         printf("%d",L->item);
23         L = L->prox;
24         while( L != NULL ) {
25             printf(",%d",L->item);
26             L = L->prox;
27         }
28     }
29     printf("]");
30 }
31
32
33 int main(void) {
34     Lista I = NULL;
35     exibe(I);
36     return 0;
37 }

```

```
[ ]

-----
Process exited after 1.407 seconds with return value 0
Pressione qualquer tecla para continuar. . . |
```

Ex2.c

```
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     printf("[");
19     if(L == NULL){
20
21     }else{
22         printf("%d",L->item);
23         L = L->prox;
24         while( L != NULL ) {
25             printf(",%d",L->item);
26             L = L->prox;
27         }
28     }
29     printf("]");
30 }
31
32
33 int main(void) {
34     Lista I = no(3,no(1,no(5,NULL)));
35     exibe(I);

```

```
[3,1,5]

-----
Process exited after 1.469 seconds with return value 0
Pressione qualquer tecla para continuar. . . |
```

### Exercício 3

```
Ex1.c Ex2.1.c Ex2.c Ex3.c Ex4.c Ex5.c Ex6.c Ex7.c Ex8.c Ex9.c Ex10.c Ex11.c Ex12.c Ex13.c Ex14.c Ex15.c Ex16.c Ex17.c Ex18.c Ex19.c Ex20.c Ex21.c Ex21.1.c Ex22.c Ex22.1.c Ex23.c Ex24.c
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 int tamanho(Lista L) {
18     int t = 0;
19     while( L ) {
20         t++;
21         L = L->prox;
22     }
23     return t;
24 }
25
26 void exibe(Lista L) {
27     while( L != NULL ) {
28         printf("%d\n",L->item);
29         L = L->prox;
30     }
31 }
32
33 int main(void) {
34     Lista I = no(3,no(1,no(5,NULL)));
35     exibe(I);
36     printf("Tamanho = %d\n",tamanho(I));
37 }
```

```
C:\Users\P08012631\Desktop\
3
1
5
Tamanho = 3

-----
Process exited after 0.6836 seconds with return value 0
Pressione qualquer tecla para continuar. . . |
```

### Exercício 4

```
Ex2.c Ex3.c Ex2.1.c Ex4.c
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 int tamanho(Lista L) {
18     int t = 0;
19     while( L ) {
20         t++;
21         L = L->prox;
22     }
23     return t;
24 }
25
26 int soma(Lista L) {
27     int soma = 0;
28     while( L != NULL ) {
29         soma += L->item;
30         L = L->prox;
31     }
32     return soma;
33 }
34
35 void exibe(Lista L) {
```

```
C:\Users\P08012631\Desktop\
3
1
5
Tamanho = 3
A soma dos numeros acima eh: 9

-----
Process exited after 1.126 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```

## Exercício 5

```
Ex2.c Ex3.c Ex2.1.c Ex4.c Ex5.c
6 typedef struct no {
7     Item item;
8     struct no *prox;
9 } *Lista;
10
11 Lista no(Item x, Lista p) {
12     Lista n = malloc(sizeof(struct no));
13     n->item = x;
14     n->prox = p;
15     return n;
16 }
17
18 Lista aleatoria(int n, int m) {
19     Lista L = NULL;
20     while( n>0 ) {
21         L = no(rand()%m, L);
22         n--;
23     }
24     return L;
25 }
26
27 void exibe(Lista L) {
28     while( L != NULL ) {
29         printf("%d\n",L->item);
30         L = L->prox;
31     }
32 }
33
34 int main(void) {
35     srand(time(NULL));
36     Lista A = aleatoria(10,100);
37     exibe(A);
38     return 0;
39 }
40
```

```
C:\Users\P08012631\Desktop\ X + v
69
33
74
0
52
40
40
24
62
54

-----
Process exited after 2.276 seconds with return value 0
Pressione qualquer tecla para continuar. . . |
```

## Exercício 6

```
Ex2.c Ex3.c Ex2.1.c Ex4.c Ex5.c Ex6.c
15     return n;
16 }
17
18 Lista aleatoria(int n, int m) {
19     Lista L = NULL;
20     while( n>0 ) {
21         L = no(rand()%m, L);
22         n--;
23     }
24     return L;
25 }
26
27 void exibe(Lista L) {
28     while( L != NULL ) {
29         printf("%d\n",L->item);
30         L = L->prox;
31     }
32 }
33
34 Lista intervalo(int n) {
35     Lista L = NULL;
36     for (int i = n; i >= 1; i--) {
37         L = no(i, L);
38     }
39     return L;
40 }
41
42
43 int main(void) {
44     int num;
45     printf("Intervalo de 1 a (n). Informe o numero final");
46     scanf("%d", &num);
47     Lista B = intervalo(num);
48     exibe(B);
49
50     return 0;
51 }

```

```
C:\Users\P08012631\Desktop\ X + v
Intervalo de 1 a (n). Informe o numero final do intervalo:24
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24

-----
Process exited after 9.184 seconds with return value 0
Pressione qualquer tecla para continuar. . . |
```

## Exercício 7

```
Ex2.c Ex3.c Ex2.1.c Ex4.c Ex5.c Ex6.c Ex7.c
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 anexa(Lista *A, Lista B) {
18     if( !B ) return;
19     while( *A )
20         A = &(*A)->prox;
21     *A = B;
22 }
23
24 void exibe(Lista L) {
25     while( L != NULL ) {
26         printf("%d\n", L->item);
27         L = L->prox;
28     }
29 }
30
31 int main(void) {
32     Lista H = no(4, no(2, NULL));
33     Lista I = no(3, no(1, no(5, NULL)));
34     printf("H = "); exibe(H);
35     printf("I = "); exibe(I);
36     printf("Pressione enter");
37     getchar();
}
```

```
C:\Users\P08012631\Desktop\ x + v
H = 4
2
I = 3
1
5
Pressione enter
```

## Exercício 8

```
Ex1.c Ex2.c Ex2.1.c Ex3.c Ex4.c Ex5.c Ex6.c Ex7.c Ex8.c Ex9.c Ex10.c Ex11.c Ex12.c Ex13.c Ex14.c Ex15.c Ex16.c Ex17.c
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 Item ultimo(Lista L) {
18     if (L == NULL) {
19         printf("Erro fatal: a lista esta vazia.\n");
20     }
21
22     while (L->prox != NULL) {
23         L = L->prox;
24     }
25
26     return L->item;
27 }
28
29 void exibe(Lista L) {
30     while( L != NULL ) {
31         printf("%d\n", L->item);
32         L = L->prox;
33     }
34 }
35
36 int main(void) {
37     Lista L = no(3, no(1, no(5, no(66, no(6, NULL)))));
38     exibe(L);
39     printf("Ultimo item da lista: %d\n", ultimo(L));
40
41     return 0;
42 }
43
```

```
C:\Users\P08012631\Desktop\ x + v
3
1
5
66
6
Ultimo item da lista: 6
-----
Process exited after 0.9402 seconds with return value 0
Pressione qualquer tecla para continuar. . . |
```

```
Ex1.c Ex2.c Ex2.1.c Ex3.c Ex4.c Ex5.c Ex6.c Ex7.c Ex8.c Ex9.c Ex10.c Ex11.c Ex12.c Ex13.c Ex14.c Ex15.c Ex16.c Ex17.c
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 Item ultimo(Lista L) {
18     if (L == NULL) {
19         printf("Erro fatal: a lista esta vazia.\n");
20     }
21
22     while (L->prox != NULL) {
23         L = L->prox;
24     }
25
26     return L->item;
27 }
28
29 void exibe(Lista L) {
30     while( L != NULL ) {
31         printf("%d\n", L->item);
32         L = L->prox;
33     }
34 }
35
36 int main(void) {
37     Lista L = NULL;
38     exibe(L);
39     printf("Ultimo item da lista: %d\n", ultimo(L));
40
41     return 0;
42 }
43
```

```
C:\Users\P08012631\Desktop\ x + v
Erro fatal: a lista esta vazia.
-----
Process exited after 1.850 seconds with return value 3221225477
Pressione qualquer tecla para continuar. . . |
```

## Exercício 9

```
Ex1.c Ex2.c Ex2.1.c Ex3.c Ex4.c Ex5.c Ex6.c Ex7.c Ex8.c Ex9.c Ex10.c Ex11.c Ex12.c Ex13.c Ex14.c Ex15.c Ex16.c Ex17.c
11 Lista n = malloc(sizeof(struct no));
12 n->item = x;
13 n->prox = p;
14 return n;
15 }
16
17 Item maximo(Lista L) {
18     if (L == NULL) {
19         printf("Erro fatal: a lista esta vazia.\n");
20     }
21
22     Item max = L->item;
23     while (L != NULL) {
24         if (L->item > max) {
25             max = L->item;
26         }
27         L = L->prox;
28     }
29
30     return max;
31 }
32
33 void exibe(Lista L) {
34     while (L != NULL) {
35         printf("%d\n", L->item);
36         L = L->prox;
37     }
38 }
39
40 int main(void) {
41     Lista L = no(3, no(1, no(5, no(66, no(6, NULL))));
42     exibe(L);
43     printf("Maior item da lista: %d\n", maximo(L));
44
45     return 0;
46 }
```

```
C:\Users\P08012631\Desktop\ X + v
3
1
5
66
6
Maior item da lista: 66

-----
Process exited after 0.7492 seconds with return value 0
Pressione qualquer tecla para continuar. . . |
```

```
Ex1.c Ex2.c Ex2.1.c Ex3.c Ex4.c Ex5.c Ex6.c Ex7.c Ex8.c Ex9.c Ex10.c Ex11.c Ex12.c Ex13.c Ex14.c Ex15.c Ex16.c Ex17.c
11 Lista n = malloc(sizeof(struct no));
12 n->item = x;
13 n->prox = p;
14 return n;
15 }
16
17 Item maximo(Lista L) {
18     if (L == NULL) {
19         printf("Erro fatal: a lista esta vazia.\n");
20     }
21
22     Item max = L->item;
23     while (L != NULL) {
24         if (L->item > max) {
25             max = L->item;
26         }
27         L = L->prox;
28     }
29
30     return max;
31 }
32
33 void exibe(Lista L) {
34     while (L != NULL) {
35         printf("%d\n", L->item);
36         L = L->prox;
37     }
38 }
39
40 int main(void) {
41     Lista L = NULL;
42     exibe(L);
43     printf("Maior item da lista: %d\n", maximo(L));
44
45     return 0;
46 }
```

```
C:\Users\P08012631\Desktop\ X + v
Erro fatal: a lista esta vazia.

-----
Process exited after 1.747 seconds with return value 3221225477
Pressione qualquer tecla para continuar. . . |
```

## Exercício 10

```
Ex1.c Ex2.c Ex2.1.c Ex3.c Ex4.c Ex5.c Ex6.c Ex7.c Ex8.c Ex9.c Ex10.c Ex11.c Ex12.c Ex13.c Ex14.c Ex15.c Ex16.c Ex17.c
13 n->prox = p;
14 return n;
15 }
16
17 int pertence(Item x, Lista L) {
18     while (L != NULL) {
19         if (L->item == x) {
20             return 1;
21         }
22         L = L->prox;
23     }
24     return 0;
25 }
26
27 void exibe(Lista L) {
28     while (L != NULL) {
29         printf("%d\n", L->item);
30         L = L->prox;
31     }
32 }
33
34 int main(void) {
35     printf("Lista:\n");
36     Lista L = no(3, no(1, no(5, no(66, no(6, NULL))));
37     exibe(L);
38
39     int x = 66;
40     if (x, pertence(x, L)) {
41         printf("\n0 item %d pertence a lista(L) acima.", x);
42     }
43     else {
44         printf("\n0 item %d nao pertence a lista(L) acima.", x);
45     }
46
47     return 0;
48 }
```

```
C:\Users\P08012631\Desktop\ X + v
Lista:
3
1
5
66
6

0 item 66 pertence a lista(L) acima.

-----
Process exited after 1.135 seconds with return value 0
Pressione qualquer tecla para continuar. . . |
```

## Exercício 11

The screenshot shows a C program in a code editor with tabs from Ex1.c to Ex17.c. The code defines a linked list structure and functions to reverse it. The main function creates a list with nodes containing 3, 66, 6, and NULL, then calls the reverse function and prints the result. The output window shows the original list [3,1,5,66,6] and the reversed list [6,66,5,1,3].

```
14 }
15 return n;
16
17 Lista inversa(Lista L) {
18     Lista invertida = NULL;
19     while (L != NULL) {
20         invertida = no(L->item, invertida);
21         L = L->prox;
22     }
23     return invertida;
24 }
25
26
27 void exibe(Lista L) {
28     printf("[");
29     while (L != NULL) {
30         printf("%d", L->item);
31         if (L->prox != NULL) {
32             printf(",");
33         }
34         L = L->prox;
35     }
36     printf("]");
37 }
38
39 int main(void) {
40     printf("Lista:\n");
41     Lista L = no(3, no(1, no(5, no(66, no(6, NULL))));
42     exibe(L);
43
44     Lista L_invertida = inversa(L);
45     printf("\n\nLista invertida in 1/ 3 Lista inversa (Lista L)");
46     exibe(L_invertida);
47
48     return 0;
49 }
```

Output window content:

```
Lista:
[3,1,5,66,6]

Lista invertida:
[6,66,5,1,3]

-----
Process exited after 1.043 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```

## Exercício 12

The screenshot shows a C program in a code editor with tabs from Ex1.c to Ex13.c. The code defines a linked list structure and functions to reverse it within a specified interval. The main function prompts the user for an interval, creates a list, and calls the reverse function. The output window shows the original list [-2,-1,0,1,2,3] and the reversed list [-2,-1,0,1,2,3].

```
29 }
30
31 Lista inversa(Lista L) {
32     Lista invertida = NULL;
33     while (L != NULL) {
34         invertida = no(L->item, invertida);
35         L = L->prox;
36     }
37     return invertida;
38 }
39
40 void exibe(Lista L) {
41     printf("[");
42     while (L != NULL) {
43         printf("%d", L->item);
44         if (L->prox != NULL) {
45             printf(",");
46         }
47         L = L->prox;
48     }
49     printf("]");
50 }
51
52 int main(void) {
53     int p, u;
54     printf("Digite o intervalo de dois numeros (separados por espaço): ");
55     scanf("%d %d", &p, &u);
56
57     Lista L = intervalo(p, u);
58     Lista L_invertida = inversa(L);
59     printf("\nIntervalo escolhido: %d...%d\n", p, u);
60     printf("Lista: \n");
61     exibe(L_invertida);
62
63     return 0;
64 }
65 }
```

Output window content:

```
Digite o intervalo de dois numeros (separados por espaço): -2 3

Intervalo escolhido: -2...3

Lista:
[-2,-1,0,1,2,3]

-----
Process exited after 3.211 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```

The screenshot shows a C program in a code editor with tabs from Ex1.c to Ex24.c. The code defines a linked list structure and functions to reverse it within a specified interval. The main function prompts the user for an interval, creates a list, and calls the reverse function. The output window shows the original list [9,1] and the reversed list [9,1].

```
18 Lista lista = NULL;
19
20 if (p > u) {
21     return lista;
22 }
23
24 for (int i = p; i <= u; i++) {
25     lista = no(i, lista);
26 }
27
28 return lista;
29 }
30
31 Lista inversa(Lista L) {
32     Lista invertida = NULL;
33     while (L != NULL) {
34         invertida = no(L->item, invertida);
35         L = L->prox;
36     }
37     return invertida;
38 }
39
40 void exibe(Lista L) {
41 }
42
43 int main(void) {
44     int p, u;
45     printf("Digite o intervalo de dois numeros (separados por espaço): ");
46     scanf("%d %d", &p, &u);
47
48     Lista L = intervalo(p, u);
49     Lista L_invertida = inversa(L);
50     printf("\nIntervalo escolhido: %d...%d\n", p, u);
51     printf("Lista: \n");
52     exibe(L_invertida);
53
54     return 0;
55 }
```

Output window content:

```
Digite o intervalo de dois numeros (separados por espaço): 9 1

Intervalo escolhido: 9...1

Lista:
[9,1]

-----
Process exited after 4.4 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```

## Exercício 13

The screenshot shows a C program in a code editor with tabs for Ex1.c through Ex13.c. The code defines a linked list structure with functions for length calculation, display, and a main function. The main function creates a list with values 3, 1, and 5. The execution window shows the output: 'Lista: [3,1,5]', 'Tamanho da lista eh: 3', and a message indicating the process exited after 0.7663 seconds.

```
12 n->item = x;  
13 n->prox = p;  
14 return n;  
15 }  
16  
17 int len(Lista L) {  
18     if (L == NULL) {  
19         return 0;  
20     } else {  
21         return 1 + len(L->prox);  
22     }  
23 }  
24  
25 void exibe(Lista L) {  
26     printf("[");  
27     if(L == NULL){  
28  
29     }else(  
30         printf("%d",L->item);  
31         L = L->prox;  
32         while( L != NULL ) {  
33             printf(",%d",L->item);  
34             L = L->prox;  
35         }  
36     }  
37     printf("]");  
38 }  
39  
40  
41 int main(void) {  
42     printf("Lista: ");  
43     Lista L = no(3,no(1,no(5,NULL)));  
44     exibe (L);  
45     printf("\nTamanho da lista eh: %d", len(L));  
46  
47     return 0;  
48 }
```

C:\Users\P08012631\Desktop\ X + -  
Lista: [3,1,5]  
Tamanho da lista eh: 3  
-----  
Process exited after 0.7663 seconds with return value 0  
Pressione qualquer tecla para continuar. . .

## Exercício 14

The screenshot shows a C program in a code editor with tabs for Ex1.c through Ex14.c. The code defines a linked list structure and a recursive function to calculate the sum of the list. The main function creates a list with values 3, 1, and 5. The execution window shows the output: 'Lista: [3,1,5]', 'A soma da lista eh: 9', and a message indicating the process exited after 1.461 seconds.

```
6 Item item;  
7 struct no {  
8     Item item;  
9     struct no *prox;  
10 } *lista;  
11  
12 Lista no(Item x, Lista p) {  
13     Lista n = malloc(sizeof(struct no));  
14     n->item = x;  
15     n->prox = p;  
16     return n;  
17 }  
18  
19 int sum(Lista L) {  
20     if (L == NULL) {  
21         return 0;  
22     } else {  
23         return L->item + sum(L->prox);  
24     }  
25 }  
26  
27 void exibe(Lista L) {  
28     printf("[");  
29     if(L == NULL){  
30  
31     }else(  
32         printf("%d",L->item);  
33         L = L->prox;  
34         while( L != NULL ) {  
35             printf(",%d",L->item);  
36             L = L->prox;  
37         }  
38     }  
39     printf("]");  
40 }  
41  
42 int main(void) {  
43     printf("Lista: ");  
44     Lista L = no(3,no(1,no(5,NULL)));  
45     exibe (L);  
46     printf("\nA soma da lista eh: %d", sum(L));  
47  
48     return 0;  
49 }
```

C:\Users\P08012631\Desktop\ X + -  
Lista: [3,1,5]  
A soma da lista eh: 9  
-----  
Process exited after 1.461 seconds with return value 0  
Pressione qualquer tecla para continuar. . .

## Exercício 15

```
Ex1.c Ex2.c Ex2.1.c Ex3.c Ex4.c Ex5.c Ex6.c Ex7.c Ex8.c Ex9.c Ex10.c Ex11.c Ex12.c Ex13.c Ex14.c Ex15.c
15 }
16
17 int clone(Lista L) {
18     if (L == NULL) {
19         return 0;
20     } else {
21         return no(L->item, clone(L->prox));
22     }
23 }
24
25 void exibe(Lista L) {
26     printf("[");
27     if (L == NULL) {
28     } else {
29         printf("%d", L->item);
30         L = L->prox;
31         while (L != NULL) {
32             printf(",%d", L->item);
33             L = L->prox;
34         }
35     }
36     printf("]");
37 }
38
39 }
40
41 int main(void) {
42     Lista L = no(3, no(1, no(5, NULL)));
43     Lista clonada = clone(L);
44
45     printf("Lista original: ");
46     exibe(L);
47     printf("\n\nCopia da lista: ");
48     public int __cdecl printf(const char * __restrict,
49
50     return 0;
51 }
```

```
C:\Users\P08012631\Desktop\ X + v
Lista original: [3,1,5]
Copia da lista: [3,1,5]
-----
Process exited after 0.9045 seconds with return value 0
Pressione qualquer tecla para continuar. . . |
```

## Exercício 16

```
Ex1.c Ex2.c Ex2.1.c Ex3.c Ex4.c Ex5.c Ex6.c Ex7.c Ex8.c Ex9.c Ex10.c Ex11.c Ex12.c Ex13.c Ex14.c Ex15.c Ex16.c
21 for (int i = 0; i < n; i++) {
22     Item valor = rand() % m;
23     lista = no(valor, lista);
24 }
25
26 return lista;
27 }
28
29 void exibe(Lista L) {
30     printf("[");
31     if (L == NULL) {
32     } else {
33         printf("%d", L->item);
34         L = L->prox;
35         while (L != NULL) {
36             printf(",%d", L->item);
37             L = L->prox;
38         }
39     }
40     printf("]");
41 }
42
43 }
44
45 int main(void) {
46     int m = 100;
47     int n = 0;
48
49     printf("Informe a quantidade de numeros que deseja: ");
50     scanf("%d", &n);
51
52     Lista lista_aleatoria = rnd(n, m);
53     printf("\nLista de numeros aleatorios com %d itens entre 0 e 99:");
54     public int __cdecl printf(const char * __restrict, _Format,
55
56     return 0;
57 }
```

```
C:\Users\P08012631\Desktop\ X + v
Informe a quantidade de numeros que deseja: 6
Lista de numeros aleatorios com 6 itens entre 0 e 99:
[11,82,3,59,88,66]
-----
Process exited after 1.909 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```

## Exercício 17

```
Ex1.c Ex2.c Ex2.1.c Ex3.c Ex4.c Ex5.c Ex6.c Ex7.c Ex8.c Ex9.c Ex10.c Ex11.c Ex12.c Ex13.c Ex14.c Ex15.c Ex16.c Ex17.c
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 Item last(Lista L) {
18     if (L == NULL) {
19         printf("Erro: a lista esta vazia.\n");
20     }
21     if (L->prox == NULL) {
22         return L->item;
23     }
24     return last(L->prox);
25 }
26
27 void exibe(Lista L) {
28     printf("[");
29     if (L == NULL) {
30     } else {
31         printf("%d", L->item);
32         L = L->prox;
33         while (L != NULL) {
34             printf(",%d", L->item);
35             L = L->prox;
36         }
37     }
38 }
```

```
C:\Users\P08012631\Desktop\ X + v
Lista: [3,1,5,66,6]
O ultimo item da Lista eh: 6
-----
Process exited after 0.973 seconds with return value 0
Pressione qualquer tecla para continuar. . . |
```



## Exercício 18

```
Ex1.c Ex2.c Ex2.1.c Ex3.c Ex4.c Ex5.c Ex6.c Ex7.c Ex8.c Ex9.c Ex10.c Ex11.c Ex12.c Ex13.c Ex14.c Ex15.c Ex16.c Ex17.c Ex18.c
22     return 1;
23 }
24 return in(x, L->prox);
25 }
26
27 void exibe(Lista L) {
28     printf("[");
29     if(L == NULL){
30
31     }else{
32         printf("%d", L->item);
33         L = L->prox;
34         while( L != NULL ) {
35             printf(", %d", L->item);
36             L = L->prox;
37         }
38     }
39     printf("]");
40 }
41
42 int main(void) {
43     int x = 0;
44     printf("Informe um numero para verificar se ele pertence a lista: ");
45     scanf("%d", &x);
46
47     printf("\nLista: ");
48     Lista L = no(3, no(1, no(5, no(66, no(6, NULL))));
49     exibe(L);
50
51     if (in(x, L)){
52         printf("\n\nO numero %d pertence a Lista.", x);
53     }else{
54         printf("\n\nO numero %d nao pertence a Lista.", x);
55     }
56
57     return 0;
58 }
```

```
C:\Users\P08012631\Desktop\ X + v
Informe um numero para verificar se ele pertence a lista: 5

Lista: [3,1,5,66,6]

O numero 5 pertence a Lista.

-----
Process exited after 3.647 seconds with return value 0
Pressione qualquer tecla para continuar. . . |
```

## Exercício 19

```
Ex1.c Ex2.1.c Ex2.c Ex3.c Ex4.c Ex5.c Ex6.c Ex7.c Ex8.c Ex9.c Ex10.c Ex11.c Ex12.c Ex13.c Ex14.c Ex15.c Ex16.c Ex17.c Ex18.c Ex19.c
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 Item nth(int n, Lista L) {
18     if (n < 1 || L == NULL) {
19         printf("Índice inválido ou lista vazia.\n");
20     }
21     if (n == 1) {
22         return L->item;
23     }
24     return nth(n - 1, L->prox);
25 }
26
27 void exibe(Lista L) {
28     printf("[");
29     if(L == NULL){
30
31     }else{
32         printf("%d", L->item);
33         L = L->prox;
34         while( L != NULL ) {
35             printf(", %d", L->item);
36             L = L->prox;
37         }
38     }
39     printf("]");
40 }
41
42 int main(void) {
43     printf("\nLista: ");
44     Lista L = no(3, no(1, no(5, no(66, no(6, NULL))));
```

```
C:\Users\P08012631\Desktop\ X + v
Lista: [3,1,5,66,6]
Informe qual a posicao do numero que deseja visualizar: 4

O 4o. item da lista eh: 66

-----
Process exited after 9.694 seconds with return value 0
Pressione qualquer tecla para continuar. . . |
```

## Exercício 20

```
Ex1.c Ex2.1.c Ex2.c Ex3.c Ex4.c Ex5.c Ex6.c Ex7.c Ex8.c Ex9.c Ex10.c Ex11.c Ex12.c Ex13.c Ex14.c Ex15.c Ex16.c Ex17.c Ex18.c Ex19.c Ex20.c
15 L
16 }
17 Item minimum(Lista L) {
18     if (L == NULL) {
19         printf("Lista vazia.\n");
20     }
21     if (L->prox == NULL) {
22         return L->item;
23     }
24     Item menor = minimum(L->prox);
25     return (L->item < menor) ? L->item : menor;
26 }
27
28 void exibe(Lista L) {
29     printf("[");
30     if(L == NULL){
31
32     }else{
33         printf("%d", L->item);
34         L = L->prox;
35         while( L != NULL ) {
36             printf(", %d", L->item);
37             L = L->prox;
38         }
39     }
40     printf("]");
41 }
42
43 int main(void) {
44     printf("\nLista: ");
45     Lista L = no(3, no(1, no(5, no(66, no(6, no(0, no(12, NULL))));
46     exibe(L);
47
48     printf("\n\nO menor item da lista eh: %d\n", minimum(L));
49
50     return 0;
```

```
C:\Users\P08012631\Desktop\ X + v
Lista: [3,1,5,66,6,0,12]

O menor item da lista eh: 0

-----
Process exited after 2.018 seconds with return value 0
Pressione qualquer tecla para continuar. . . |
```

## Exercício 21

```
Ex1.c Ex2.1.c Ex2.c Ex3.c Ex4.c Ex5.c Ex6.c Ex7.c Ex8.c Ex9.c Ex10.c Ex11.c Ex12.c Ex13.c Ex14.c Ex15.c Ex16.c Ex17.c Ex18.c Ex19.c Ex20.c Ex21.c
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 int sorted(Lista L) {
18     if (L == NULL || L->prox == NULL) {
19         return 1;
20     }
21     if (L->item > L->prox->item) {
22         return 0;
23     }
24     return sorted(L->prox);
25 }
26
27 void exibe(Lista L) {
28     printf("[");
29     if(L == NULL){
30
31     }else{
32         printf("%d",L->item);
33         L = L->prox;
34         while( L != NULL ) {
35             printf(",%d",L->item);
36             L = L->prox;
37         }
38     }
39     printf("]");
40 }
41
42 int main(void) {
43     printf("\nLista: ");
44     Lista L = no(1, no(2, no(3, no(4, no(5, NULL))));
45     exibe(L);
46
47     if (sorted(L)){
48         printf("\n\nA lista esta ordenada!");
49     }else{
50         printf("\n\nA lista nao esta ordenada!");
51     }
52
53     return 0;
54 }
```

C:\Users\P08012631\Desktop\ x + v

Lista: [3,1,5,66,6,0,12]

A lista nao esta ordenada!

Process exited after 0.8975 seconds with return value 0  
Pressione qualquer tecla para continuar. . . |

C:\Users\P08012631\Desktop\ x + v

Lista: [1,2,3,4,5]

A lista esta ordenada!

Process exited after 0.8622 seconds with return value 0  
Pressione qualquer tecla para continuar. . . |

## Exercício 22

```
Ex1.c Ex2.1.c Ex2.c Ex3.c Ex4.c Ex5.c Ex6.c Ex7.c Ex8.c Ex9.c Ex10.c Ex11.c Ex12.c Ex13.c Ex14.c Ex15.c Ex16.c Ex17.c Ex18.c Ex19.c Ex20.c Ex21.c Ex21.1.c Ex22.c
4 typedef int Item;
5 typedef struct no {
9
10 Lista no(Item x, Lista p) {
16
17 int equal(Lista A, Lista B) {
18     if (A == NULL && B == NULL) {
19         return 1;
20     }
21     if (A == NULL || B == NULL) {
22         return 0;
23     }
24     if (A->item != B->item) {
25         return 0;
26     }
27     return equal(A->prox, B->prox);
28 }
29
30 void exibe(Lista L) {
44
45 int main(void) {
46     printf("\nLista A: ");
47     Lista A = no(1, no(2, no(3, no(4, no(5, NULL))));
48     exibe(A);
49
50     printf("\nLista B: ");
51     Lista B = no(1, no(2, no(3, no(4, no(5, NULL))));
52     exibe(B);
53
54     if (equal(A,B)){
55         printf("\n\nAs listas sao iguais!");
56     }else{
57         printf("\n\nAs listas nao sao iguais!");
58     }
59     public int __cdecl printf (const char * __restrict __Format
60
61     return 0;
62 }
```

```
C:\Users\P08012631\Desktop\ X + v
Lista A: [1,2,3,4,5]
Lista B: [1,2,3,4,5]

As listas sao iguais!

Process exited after 0.7122 seconds with return value 0
Pressione qualquer tecla para continuar. . . |
```

```
Ex1.c Ex2.1.c Ex2.c Ex3.c Ex4.c Ex5.c Ex6.c Ex7.c Ex8.c Ex9.c Ex10.c Ex11.c Ex12.c Ex13.c Ex14.c Ex15.c Ex16.c Ex17.c Ex18.c Ex19.c Ex20.c Ex21.c Ex21.1.c Ex22.c Ex22.1.c
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 typedef int Item;
5 typedef struct no {
9
10 Lista no(Item x, Lista p) {
16
17 int equal(Lista A, Lista B) {
18     if (A == NULL && B == NULL) {
19         return 1;
20     }
21     if (A == NULL || B == NULL) {
22         return 0;
23     }
24     if (A->item != B->item) {
25         return 0;
26     }
27     return equal(A->prox, B->prox);
28 }
29
30 void exibe(Lista L) {
44
45 int main(void) {
46     printf("\nLista A: ");
47     Lista A = no(3, no(1, no(5, no(66, no(6, no(0, no(12, NULL))));
48     exibe(A);
49
50     printf("\nLista B: ");
51     Lista B = no(1, no(2, no(3, no(4, no(5, NULL))));
52     exibe(B);
53
54     if (equal(A,B)){
55         printf("\n\nAs listas sao iguais!");
56     }else{
57         printf("\n\nAs listas nao sao iguais!");
58     }
59
60 }
```

```
C:\Users\P08012631\Desktop\ X + v
Lista A: [3,1,5,66,6,0,12]
Lista B: [1,2,3,4,5]

As listas nao sao iguais!

Process exited after 1.857 seconds with return value 0
Pressione qualquer tecla para continuar. . . |
```

## Exercício 23

```
Ex1.c Ex2.1.c Ex2.c Ex3.c Ex4.c Ex5.c Ex6.c Ex7.c Ex8.c Ex9.c Ex10.c Ex11.c Ex12.c Ex13.c Ex14.c Ex15.c Ex16.c Ex17.c Ex18.c Ex19.c Ex20.c Ex21.c Ex21.1.c Ex22.c Ex22.1.c Ex23.c
2 #include <stdlib.h>
3
4 typedef int Item;
5 typedef struct no {
9
10 Lista no(Item x, Lista p) {
16
17 int count(Item x, Lista L) {
18     if (L == NULL) {
19         return 0;
20     }
21     if (L->item == x) {
22         return 1 + count(x, L->prox);
23     } else {
24         return count(x, L->prox);
25     }
26 }
27
28 void exibe(Lista L) {
42
43 int main(void) {
44     printf("\nLista A: ");
45     Lista L = no(3, no(1, no(5, no(1, no(6, no(0, no(1, NULL))));
46     exibe(L);
47
48     int num = 0;
49     printf("\n\nInforme um item(numero) para verificar quantas
50     scanf("%d",&num);
51
52     int ocorrencias = count(num, L);
53     if(count(num,L)){
54         printf("\n0 item %d ocorre %d vezes na lista.\n", num,
55     }else{
56         printf("\n0 item %d nao ocorre na lista.\n",num);
57     }
58
59     . . .
60 }
```

```
C:\Users\P08012631\Desktop\ X + v
Lista A: [3,1,5,1,6,0,1]

Informe um item(numero) para verificar quantas vezes ele aparece na lista acima: 1

0 item 1 ocorre 3 vezes na lista.

Process exited after 6.307 seconds with return value 0
Pressione qualquer tecla para continuar. . . |
```

## Exercício 24

```
Ex1.c Ex21.c Ex2.c Ex3.c Ex4.c Ex5.c Ex6.c Ex7.c Ex8.c Ex9.c Ex10.c Ex11.c Ex12.c Ex13.c Ex14.c Ex15.c Ex16.c Ex17.c Ex18.c Ex19.c Ex20.c Ex21.c Ex22.c Ex23.c Ex24.c
9
10 Lista no(Item x, Lista p) {
16
17 Lista replace(Item x, Item y, Lista L) {
18     if (L == NULL) {
19         return NULL;
20     }
21     if (L->item == x) {
22         L->item = y;
23     }
24     L->prox = replace(x, y, L->prox);
25     return L;
26 }
27
28 void exibe(Lista L) {
42
43 int main(void) {
44     printf("\nLista Original: ");
45     Lista L = no(3, no(1, no(5, no(1, no(6, no(0, no(1, NULL))))));
46     exibe(L);
47
48     int num = 0;
49     printf("\nInforme qual numero da lista quer substituir: ");
50     scanf("%d", &num);
51
52     int substituto = 0;
53     printf("\nInforme o numero que substituir o numero %d: ", num);
54     scanf("%d", &substituto);
55
56     L = replace(num, substituto, L);
57
58     printf("\n\nLista Modificada: ");
59     exibe(L);
60
61     |
62     return 0;
63 }
```

C:\Users\P08012631\Desktop\ X + v

Lista Original: [3,1,5,1,6,0,1]

Informe qual numero da lista quer substituir: 1

Informe o numero que substituir o numero 1: 8

Lista Modificada: [3,8,5,8,6,0,8]

Process exited after 19.28 seconds with return value 0  
Pressione qualquer tecla para continuar. . . |