```
/home/acer/Arquivos_CodeBlocks/Atv1_mod2_ED_Q3_MARLON-DUARTE_493408/l_e_o.c
Página 1 de 3
                                                               dom 14 fev 2021 20:47:23
   1
       #include <stdio.h>
   2
       #include <stdlib.h>
   3
       #include "l e o.h"
   4
                                Consloute
   5.
       void create(int n){
          vetor = (int*) realloc (vetor, n*(sizeof(int))); ((5)
   6
   7
          tam = -1;
                      0(3)
   8
          if (vetor == NULL) {
            printf ("\nERRO!\n"); O(\)
  .9
  10
              exit (1); ((4)
  11
          } else
  12
              printf("\n\nEspaço alocado com sucesso!\n"); ()
 13
  14
  15
       void cria_vaga(int indice){
  16
                                        //Função local, complementa a função do TAD
       "add".
  17
  18
          for (int i=tam; i>indice; i--){ (m)
  19
         if (i>0){
             vetor[i]=vetor[(i-1)]; () [4]
  20
  21
  22
         }
  23
  24
  25
       int add(int valor, int espaco)( )
  26
           posicao = 0; O(3)
  27
  28
           if (isFull(espaco) == TRUE){
              printf("\m\nvetor cheio!!!\n"); O(3)
  29
  30
              return FALSE; Of
  31
  32
  33
          if (isempty() == TRUE){
  34
             tam++;
                      0151
              vetor[tam] = valor; 0(3)
  35
  36
              return TRUE; O(%)
  37
          tam++; () /
  38
  39
          while (postcao < tam){
              if (vetor[posicao] < valor){ ( ( )
  40
  41
                  postcao++;
  42
  43
             lelse {
  44
                  cria_vaga(postcao):
                 vetor[posicao] water; O(5)
  45
  46
                  return TRUE; O(5)
  47
         . }
  48
```

```
/home/acer/Arquivos_CodeBlocks/Atv1_mod2_ED_Q3_MARLON-DUARTE_493408/l_e_o.c
Página 2 de 3
                                                                    dom 14 fev 2021 20:47:23
  49
           vetor[posicao] = valor; ()()
  50
  51
         return TRUE;
  52
  53
  54
       void reorganiza(int indice){
                                             //Função local que complementa a função do
                                                                                            J
       TAD "remove_item".
  55
                                           0(0)
  56
           for (int i=indice; i<tam; i++){
  57
               vetor[i]=vetor[(i+1)];
  58
  59
  60
  61
       int remove_item(int chave){ = 0 0 (n)
  62
           posicao = 0; \bigcirc(\Delta)
  63
  64
           if (isempty() == TRUE){
                                      0(4)
               printf("\n\nVazio! Wada a remover!\n"); ( ( )
  65
  66
               return FALSE; O(1)
  67
  68
           while (posicao < tam+1){
               if (vetor[posicao] == chave){ O(4)
  69
  70
                    reorganiza(posicao); (4)
  71
                   return TRUE;
  72
                           0(1)
  73
               posicao++;
  74
           return FALSE; 0 (-1)
  75
  76
  77
       int size_lista()( =) count oute
  78
  79
           return tam+1;
  80
       int linearSearch(int chave){ = 0 (n) :

for (int i=0;i<tam+1;i++){ (n+3)

if (vetor[i] = chave)
  81
  82
  83
               if (vetor[i] == chave) O(A
  84
                  , return 1; ()(A)
  85
  86
           printf("\m\nChave não encontrada!\n"); (4)
  87
           return -404; ()(A)
  88
  89
  90
       int bynarySearch(int chave) = 0 (log n)
  91
           int esq = 0, dir = tam:
  92
  93
           while (esq <= dir){
  94
               int meio = (esq + dir)/2;
  95
  96
```

```
/home/acer/Arquivos_CodeBlocks/Atv1_mod2_ED_Q3_MARLON-DUARTE_493408/l_e_o.c
Página 3 de 3
                                                                 dom 14 fev 2021 20:47:23
  97
               if (vetor[meio] == chave)
  98
                  return meio:
  99
               else if (vetor[meio] > chave)
 100
                  dir = meio -1;
 101
               else
 102
                  esq = meio + 1;
 103
 104
           printf("\n\nChave não encontrada!\n");
 105
           return -404;
 106
 107
                             O(b) constante
 108
       int isempty(){
 109
 110
       if (tam < 0)
 111
              return TRUE;
 112
           else
 113
               return FALSE:
 114
       int is Full (int espaco) (-) (3) countoute
 115
 116
 117
         if ((tam+1) == espaco)
 118
            return TRUE;
 119
           else
 120
              return FALSE;
 121
       1
                                       constante
 122
       int clear_vetor(){-> (3)
 123
 124
           free(vetor);
 125
           return TRUE;
 126
 127
       void print_all(){ - 1
 128
 129
           printf("\n"); () ()
 130
           printf("["); 0 (5)
 131
           for (int i=0; i<tam+1; i++){ () \( \)
             printf(" wd ,", vetor[i]); O(
 132
 133
         printf("]"); (()
 134
 135
 136
       }
 137
```

```
/home/acer/Arquivos_CodeBlocks/Atv1_mod2_Q2_MARLON-DUARTE_493408/lcde.c
dom 14 fev 2021 20:46:42
Página 1 de 3
```

```
#include <stdio.h>
1
   #include <stdlib.h>
2
    #include "lcde.h"
3
    lista_cde *create (){ = D O (3) Connocte
4
5
6
        lista_cde *lista = (lista_cde *)malloc(sizeof (lista_cde)); ()
7
8
                           0151
        if(lista != NULL) {
9
            lista->inicio = NULL; ()
10
           lista->fim = NULL; ()(1)
11
            lista->tamanho = 0; () (4)
12
13
14
         return lista; O(A)
15
16
17
     int add_item(lista_cde *lista, int valor){ = > 0 (5) POW OUT
18
19
         no *pnovo = (no*) malloc(sizeof(no)); ()
20
21
         if(pnovo != NULL) {
 22
             pnovo->valor = valor; O (1)
 23
             pnovo->proximo = NULL; ( ) ( )
 24
 25
             if(lista->inicio == NULL) { ( )
 26
                 lista->inicio = pnovo; O (4)
 27
             }else {
 28
                 lista->fim->proximo = pnovo; () (4)
 29
 30
 31
             lista->fim = pnovo; ()[\scales
 32
              lista->tamanho++;
 33
              printf("\n\nValor add %d", pnovo->valor); ( ()
 34
              return TRUE; O(A)
 35
 36
          return FALSE; () (4)
 37
 38
      }
  39
      int remove_item(lista_cde *lista, int chave){ => (7)
 40
 41
          if (!isempty(lista)) { ( (4)
  42
 43
              no *alvo = lista->inicio; ()
  44
              no *anterior;
  45
  46
              while(alvo != NULL && alvo->valor != chave) { ( ( )
  47
  48
                  anterior = alvo;
                                     -(O(1)
  49
                  alvo = alvo->proximo; 0(4
```

```
/home/acer/Arquivos_CodeBlocks/Atv1_mod2_Q2_MARLON-DUARTE_493408/lcde.c
Página 2 de 3
                                                                   dom 14 fev 2021 20:46:42
  50
               }
  51
  52
               if(alvo != NULL) { (6)
  53
                   if(alvo != lista->inicio) { O (4)
                       anterior->proximo = alvo->proximo; ∩ (△)
  54
  55
                   } else {
  56
                       lista->inicio = alvo->proximo;
  57
  58
  59
                   if(alvo == lista->fim) {
  60
                       lista->fim = anterior;
  61
  62
  63
               lista->tamanho--;
  64
               free(alvo);
  65
  66
               return TRUE;
  67
  68
  69
           return FALSE; (3)
  70
       }
  71
       int size_item(lista_cde *lista){ (1) Consoute
  72
  73
           return lista->tamanho;
  74
 .75
  76
       int find_item(lista_cde *lista, int valor){ => ()(n)
  77
           int counter=0; O(b)
  78
  79
           if(isempty(lista)) ( ( )
               printf("Lista esta' vazia.\n"); [ ( 1 )
  80
  81
           else{
  82
               no *alvo = lista->inicio; O(3)
  83
  84
  85
               while(alvo != NULL){
                                         O(n)
  86
                                       //printf("\n\n %d %d \n\n", alvo->valor,
                                       alvo->proximo->valor);
  87
                   if (valor == alvo->valor){
                                                0(1)
                       printf("\n\nEncontrado no indice %d \n\n", counter); O(1)
  88
  89
                   return counter; O(4)
  90
  91
               alvo = alvo->proximo; (1)
  92
               counter++;
  93
  94
  95
  96
           return -1; (0(3)
  97
```

```
/home/acer/Arquivos_CodeBlocks/Atv1_mod2_Q2_MARLON-DUARTE_493498/lcde.c
Página 3 de 3
                                                                 dom 14 fev 2021 20:46:42
  98
       }
  99
       int isempty(lista_cde *lista)(-> 0 (1) earstoute
 100
 101
           return lista->tamanho <= 0;
 102
 103
       int clear_item(lista_cde *lista){BO(3) constonte
 164
 105
 106
           free(lista); O(b)
 107
           lista->tamanho = 0; ()
 108
 109
           return TRUE; ( )
 110
 111
       int printAll (lista_cde *1) { = D O (n)
 112
           no *p = l->inicio; O(5)
 113
 114
                              // faz p apontar para o nó inicial
 115
                              // testa se lista não é vazia
 116
           if ((p != NULL) && (1->tamanho != 0)) { ( )
 117
                              // percorre os elementos até alcançar novamente o início
               for (int i=0; i<(l->tamanho); i++){ ( ( )
 118
 119
                  printf(" %d -", p->valor); ((A) // imprime informação do nó
 120
                   p = p->proximo;
 121
 122
               printf("\n");
                                0(7)
 123
              return TRUE;
 124
           printf("\n\nLista Vazia!\n\n"); (4)
 125
 126
           return FALSE; O(4)
 127
 128
 129
 130
```

```
1
     #include <stdio.h>
2
     #include <stdlib.h>
3
     #include "No.h"
4
     No* createNo(int v, No* proximo){ =  Comploute
5
6
7
        No * n = (No*) malloc(sizeof(No)); _ 0
8
9
        n->valor = v;
10
        n->proximo = proximo;
11
12
       return n;
13
14
15
     void printNo(No* n){= } coust on to
16
17
     if(n != NULL && n->proximo != NULL) (5)
18
19
            printf("No [valor: %i, proximo: %i]\n", n->valor, n->proximo->valor); () (1)
        else if(n != NULL)
20
            printf("No [valor: %i, proximo: NULL]\n", n->valor); ()
21
22
23
            printf("NULL\n"); O(1)
24
25
26
      void freeNo(No* n){
27
28
         free(n);
29
30
```

```
/home/acer/Arquivos_CodeBlocks/Atv01_mod2_ED_ListaEncadeada/ListaEncadeada.c
Página 1 de 3
                                                              dom 14 fev 2021 20:49:05
   1
      #include <stdio.h>
   2
      #include <stdlib.h>
   3
      #include "ListaEncadeada.h"
   4
      ListaEncadeada *create() { DOWN Ou le
   5
           ListaEncadeada *lista = (ListaEncadeada *)malloc(sizeof (ListaEncadeada));
   6
   7
   8
           if(lista != NULL) {
  9
               lista->inicio = NULL;
  10
               lista->fim = NULL;
  11
              lista->tamanho = 0;
  12
           }
  13
  14
         return lista; ( )
  15
  16
  17
                                                => consonte
  18
       int add(ListaEncadeada *lista, int valor) {
  19
  20
          No* pnovo = (No*) malloc(sizeof(No));
  21
  22
           if(pnovo != NULL) {
                                               0(4)
  23
                   pnovo->valor = valor; O(1)
  24
                   pnovo->proximo = NULL; O(3)
  25
  26
                   if(lista->inicio == NULL) {
  27
                     lista->inicio = pnovo; O(1)
  28
                   } else {
                   lista->fim->proximo = pnovo; O(3)
  29
  30
  31
  32
                   lista->fim = pnovo;
  33
                   lista->tamanho++;
  34
                   return 1;
  35
  36
  37
           return 0; ( )
  38
  39
       int remover(ListaEncadeada *lista, int chave){
  40
  41
  42
           if (!isEmpty(lista)) { ( (1)
  43.
               No *alvo = lista->inicio; () ( )
  44
  45
               No *anterior;
  46
               while(alvo != NULL && alvo->valor != chave) { ( )
  47
  48
                  anterior = alvo; O(3)
                  alvo = alvo->proximo; ()
  49
```

```
/home/acer/Arquivos_CodeBlocks/Atv01_mod2_ED_ListaEncadeada/ListaEncadeada.c
Página 2 de 3
                                                                dom 14 fev 2021 20:49:05
  50
  51
  52
               if(alvo != NULL) {
  53
                   if(alvo != lista->inicio) {
  54
                      anterior->proximo = alvo->proximo; ()(3)
  55
                   } else {
  56
                      lista->inicio = alvo->proximo;
  57
  58
  59
                   if(alvo == lista->fim) {
  60
                      lista->fim = anterior:
  61
  62
  63
               lista->tamanho--: 0 (8)
  64
                   freeNo(alvo); 0(3)
  65
                   return 1;
  66
  67
  68
  69
           return 0;
  70
  71
      int size_p(ListaEncadeada *lista){=> Countout
  72
  73
           return lista->tamanho:
  74
       }
  75
       int finder(ListaEncadeada* lista, int valor){
  76
  77
          int counter=0; O(3)
  78
                                (2)0
 79
          if(isEmpty(lista))
  80
                  printf("Lista esta' vazia.\n"); O(4)
 81
               else{
                                                0(1)
  82
                  No *alvo = lista->inicio:
  83
  84
                                             0(n)
                   while(alvo != NULL){
  85
                    //printf("\n\n %d %d \n\n", alvo->valor, alvo->proximo->valor);
  86
                      if (valor == alvo->valor){ (4)
  87
                          printf("\n\nEncontrado na posição %d \n\n", counter); ( )
 88
                          return counter; (1)
 89
 90
 91
                   alvo = alvo->proximo; O(4)
 92
                      counter++;
 93
 94
 95
                       0(4)
 96
        return -1;
 97
 98
```

```
/home/acer/Arquivos_CodeBlocks/Atv01_mod2_ED_ListaEncadeada/ListaEncadeada.c
Página 3 de 3
                                                               dom 14 fev 2021 20:49:05
       int isEmpty(ListaEncadeada* lista){ = > Coun ou >
  99
 100
          return lista->tamanho <= 0; ()
 101
       }
 102
       void printListaEncadeada(ListaEncadeada* lista){ = }
 103
 104
 105
               printf("\n-----\n\n"); O(5)
 106
 107
               printf("Lista [tamanho: %i, limite: quantidade de memo'ria aloca'vel]
               \n\n", lista->tamanho); () (3)
 108
 109
               if(isEmpty(lista))
                  isEmpty(lista)) O(5)
printf("Lista esta' vazia.\n"); O(4)
 110
 111
               else{
 112
                  No *alvo = lista->inicio; O(*)
 113
 114
                   while(alvo != NULL){ O(n)
 115
                      printNo(alvo);
 116
                      alvo = alvo->proximo; (4)
 117
 118
 119
 120
 121
      void clear_p(ListaEncadeada* lista){ = 0 Countoute
 122
 123
          free(lista);
 124
 125
          lista->tamanho = 0; ()
 126
 127
```