```
//Implementação de Listas (dinâmicas)
#include <stdio.h>
#include <stdlib.h>

//Definição de constantes
#define FALSE 0
#define TRUE 1

//Definição de tipos

//definindo o struct que representará cada posição da lista, denominado
de Nó.
//Cada Nó conterá o elemento da lista e o ponteiro para o próximo
elemento
typedef struct No {
        int valor;                      //cada valor sendo manipulado
pela lista
        struct No* prox;        //endereço do nó que contém o próximo
elemento
} TNo;

typedef TNo* TLista;

//protótipos das funções
int inserir (TLista* L, int numero);
int removerI (TLista* L, int numero);
TLista removerR (TLista* L, int numero);
int alterar (TLista L, int velho, int novo);
TLista buscar (TLista L, int numero);
void exibir (TLista L);
void destruirI (TLista* L);
void destruirR (TLista* L);

int menu ();

//MAIN
int main ()
{
        TLista pos, L = NULL;   //criando a lista, inicialmente vazia
        int op, num1, num2, quant;

        do
        {
                system ("cls");
                op = menu ();

                switch (op)
                {
                        case 1: //Inserir
                                        printf ("Entre com o elemento a
ser inserido: ");
                                        scanf ("%d", &num1);

                                        if (inserir (&L, num1) == TRUE)
                                        {
                                                printf ("Elemento %d
inserido com sucesso!\n", num1);
```

```
                                        }
                                        else
                                        {
                                                printf ("ERRO: Elemento
%d nao inserido!\n", num1);
                                        }
                                        break;

                        case 2: //Remover
                                        printf ("Entre com o elemento a
ser removido: ");
                                        scanf ("%d", &num1);

                                        quant = remover (&L, num1);
                                        if (quant > 0)
                                        {
                                                printf ("Elemento %d foi
removido %d vezes!\n", num1, quant);
                                        }
                                        else
                                        {
                                                printf ("ERRO: Elemento
%d nao existe na lista!\n", num1);
                                        }
                                        break;

                        case 3: //Alterar
                                        printf ("Entre com o elemento a
ser alterado: ");
                                        scanf ("%d", &num1);

                                        printf ("Entre com o novo
elemento: ");
                                        scanf ("%d", &num2);

                                        quant = alterar (L, num1, num2);
                                        if (quant > 0)
                                        {
                                                printf ("%d alteracoes
realizadas!\n", quant);
                                        }
                                        else
                                        {
                                                printf ("ERRO: Elemento
%d nao existe na lista!\n", num1);
                                        }
                                        break;

                        case 4: //Buscar
                                        printf ("Entre com o elemento a
ser buscado: ");
                                        scanf ("%d", &num1);

                                        pos = buscar (L, num1);
                                        if (pos)
                                        {
                                                printf ("Elemento %d foi
```

```
encontrado!\n", num1);
                                        }
                                        else
                                        {
                                                printf ("ERRO: Elemento
%d nao existe na lista!\n", num1);
                                        }
                                        break;

                        case 5: //Exibir
                                        exibir (L);
                                        break;

                        case 6: //Destruir
                                        destruirR (&L);
                                        break;

                        case 7: //Sair
                                        printf ("Fim do programa!\n");
                }
                system ("pause");
        }
        while (op != 7);
}

//Implementações

//Insere um elemento no início da Lista L, retornando TRUE ou FALSE
int inserir (TLista* L, int numero)
{
        //Passo 1: alocar memória para o novo elemento
        TLista aux = (TLista) malloc (sizeof(TNo));

        //if (aux == NULL)
        if (!aux)
        {
                return FALSE;   //sem memória disponível
        }
        else
        {
                //Passo 2: armazenando o valor na nova posição de
memória
                aux->valor = numero;

                //Passo 3: fazendo com que o novo No aponte para o
"antigo primeiro No da lista"
                aux->prox = *L;

                //Passo 4: fazer o L apontar para o novo No
                *L = aux;

                return TRUE;
        }
}

//Remove todas as ocorrências de numero em L, retornando o total de
remoções realizadas
```

```
int removerI (TLista* L, int numero)
{
        TLista pre, pos;
        int cont = 0;

        //removendo as ocorrencias no inicio da lista
        while ((*L != NULL) && ((*L)->valor == numero))
        {
                pre = *L;
                *L = (*L)->prox;
                free (pre);

                cont++;
        }

        //verificando se a lista ainda existe
        if (*L)
        {
                //tentando remover as demais ocorrências do numero
                pre = *L;
                pos = (*L)->prox;

                while (pos != NULL)
                {
                        if (pos->valor == numero)   //elemento
encontrado
                        {
                                pre->prox = pos->prox;
                                free (pos);
                                pos = pre->prox;

                                cont++;
                        }
                        else
                        {
                                pre = pos;
                                pos = pos->prox;
                        }
                }
        }

        return cont;
}

//Remover recursivo - precisar ser corrigido
TLista removerR (TLista* L, int numero)
{
        TLista aux;

        if (!(*L))
        {
                return NULL;
        }
        else
        {
                *L = removerR (&((*L)->prox), numero);
                if ((*L) && ((*L)->valor == numero))
```

```
                {
                        aux = *L;
                        *L = (*L)->prox;
                        free (aux);
                }
                return *L;
        }
}

//Altera todas as ocorrências de 'velho' por 'novo'.
//Retorna a quantidade de alterações feitas
int alterar (TLista L, int velho, int novo)
{
        TLista aux = L;
        int cont = 0;

        //while (aux != NULL)
        while (aux)
        {
                if (aux->valor == velho)
                {
                        aux->valor = novo;
                        cont++;
                }
                aux = aux->prox;
        }

        return cont;
}

//Busca o numero em L, retornando a posição de sua primeira ocorrência
(caso ele exista).
//Se o numero não for encontrado, NULL será retornado
TLista buscar (TLista L, int numero)
{
        TLista aux = L;

        while (aux != NULL)
        {
                if (aux->valor == numero)
                {
                        return aux;
                }
                aux = aux->prox;
        }

        return NULL;
}

//Exibe todos os elementos da lista
void exibir (TLista L)
{
        TLista aux;

        if (L == NULL)
        {
                printf ("Lista vazia\n");
```

```
        }
        else
        {
                aux = L;
                printf ("Lista: ");

                while (aux != NULL)
                {
                        printf ("%d ", aux->valor);
                        aux = aux->prox;
                }

                printf ("\n");
        }
}

//Destrói todos os elementos da Lista (versão iterativa)
void destruirI (TLista* L)
{
        TLista aux;

        while (*L)
        {
                aux = *L;
                *L = (*L)->prox;   //ou *L = aux->prox;
                free (aux);
        }
}

//Destrói todos os elementos da Lista (versão recursiva)
void destruirR (TLista* L)
{
        if (*L)
        {
                destruirR (&((*L)->prox));
                free (*L);
                *L = NULL;
        }
}

//Exibe um menu ao usuário, retornando a opção escolhida
int menu ()
{
        int opcao;

        printf ("Menu de opcoes:\n");
        printf ("(1) Inserir\n(2) Remover\n(3) Alterar\n(4) Buscar\n(5)
Exibir\n(6) Destruir\n(7) Sair\n\n");
        printf ("Entre com a sua opcao: ");
        scanf ("%d", &opcao);

        return opcao;
}
```