```
//AL2 - Aula de 12/06/2017
//Lista din?mica sem repeti??o de elementos

//Implementa??o de Listas (din?micas)
#include <stdio.h>
#include <stdlib.h>

//Defini??o de constantes
#define FALSE 0
#define TRUE 1

//Defini??o de tipos

//definindo o struct que representar? cada posi??o da lista, denominado
de N?.
//Cada N? conter? o elemento da lista e o ponteiro para o pr?ximo
elemento
typedef struct No {
        int valor;                      //cada valor sendo manipulado
pela lista
        struct No* prox;        //endere?o do n? que cont?m o pr?ximo
elemento
} TNo;

typedef TNo* TLista;

//prot?tipos das fun??es
int inserir (TLista* L, int numero);
int remover (TLista* L, int numero);
int alterar (TLista L, int velho, int novo);
TLista buscar (TLista L, int numero);
void exibir (TLista L);
void destruir (TLista* L);

int menu ();

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

        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);

                                        if (remover (&L, num1) == TRUE)
                                        {
                                                printf ("Elemento %d foi
removido!\n", num1);
                                        }
                                        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);

                                        if (alterar (L, num1, num2) ==
TRUE)
                                        {
                                                printf ("Alteracao
realizada!\n");
                                        }
                                        else
                                        {
                                                printf ("ERRO: alteracao
nao realizada!\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
                                        destruir (&L);
                                        break;

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

//Implementa??es

//Insere um elemento no in?cio da Lista L, retornando TRUE ou FALSE
int inserir (TLista* L, int numero)
{
        TLista aux;

        if (buscar (*L, numero) == NULL)    //verificando se o elemento
n?o existe
        {
                aux = (TLista) malloc (sizeof(TNo));

                if (!aux)
                {
                        return FALSE;   //sem mem?ria dispon?vel
                }
                else
                {
                        aux->valor = numero;

                        aux->prox = *L;

                        *L = aux;

                        return TRUE;
                }
        }
        else
        {
                return FALSE;  // pois o elemento j? existe na lista
        }
}
```

```
//Remove numero em L, retornando TRUE ou FALSE
int remover (TLista* L, int numero)
{
        TLista pre, pos;

        //verificando se o primeiro elemento ? o numero a ser removido
        if ((*L != NULL) && ((*L)->valor == numero))
        {
                pre = *L;
                *L = (*L)->prox;
                free (pre);

                return TRUE;
        }
        else
        {
                //tentando remover numero da segunda posicao em diante
                pre = *L;
                pos = (*L)->prox;

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

                                return TRUE;
                        }
                        else
                        {
                                pre = pos;
                                pos = pos->prox;
                        }
                }

                return FALSE;
        }
}

//Altera 'velho' por 'novo', se possivel, retornando TRUE ou FALSE
int alterar (TLista L, int velho, int novo)
{
        TLista posNovo = buscar (L, novo);
        TLista posVelho;

        if (posNovo == NULL)   //se o novo n?o existir...
        {
                posVelho = buscar (L, velho);

                if (posVelho != NULL) //se o velho existir ...
                {
                        posVelho->valor = novo;
                        return TRUE;
```

```
                }
                else
                {
                        return FALSE;
                }
        }
        else
        {
                return FALSE;
        }
}

//Busca o numero em L, retornando a posi??o de sua primeira ocorr?ncia
(caso ele exista).
//Se o numero n?o for encontrado, NULL ser? 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");
        }
}

//Destr?i todos os elementos da Lista (vers?o iterativa)
void destruir (TLista* L)
```

```
{
        TLista aux;

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

//Exibe um menu ao usu?rio, retornando a op??o 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;
}
```