Universidade de Trás-os-Montes e Alto Douro
Escola de Ciências e Tecnologia. Departamento de Engenharias

MIEEC + EINF
ano lectivo: **2016 / 2017**
unidade curricular: **Algoritmia**

Caderno de Exercícios
(Anexo 01): proposta de código base para Listas Encadeadas genéricas

```c
#include "stdafx.h"
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <string.h>

typedef struct _LIST_NODE
{
	void * data;
	struct _LIST_NODE * next;
} LIST_NODE;

typedef LIST_NODE * LIST;

typedef struct _ALUNO
{
	char nome[30];
	int  idade;
}
ALUNO;

typedef enum _BOOLEAN { FALSE = 0, TRUE = 1 } BOOLEAN;
typedef enum _LIST_LOCATION { LIST_START, LIST_END } LIST_LOCATION;
typedef enum _STATUS { OK, ERROR } STATUS;

#define DATA(node)   ((node)->data)
#define NEXT(node)   ((node)->next)

#define EMPTY        NULL
#define NO_LINK             NULL

void initList(LIST *list);
BOOLEAN emptyList(LIST list);
LIST_NODE * NewNode(void * data);
STATUS InsertIni(LIST *list, void *data);
STATUS InsertEnd(LIST *list, void *data);

int ListSize(LIST list);
void ShowValues(LIST list);
```

```c
/*****************************************************************
* Programa principal
*****************************************************************/
int main()
{
	LIST list = NULL;
	ALUNO *ptr_aluno;
	ALUNO al1, al2;
	char nome_aux[30];
	int idade_aux;

	strcpy(al1.nome, "Aluno 1");
	al1.idade = 11;
	strcpy(al2.nome, "Aluno 2");
	al2.idade = 22;
	if (InsertIni(&list, &al1) == ERROR)
	{
		printf("\nErro na alocacao de memoria\n");
		printf("\n<Prima qualquer tecla>\n");
		getch();
		exit(1);
	}
	if (InsertIni(&list, &al2) == ERROR)
	{
		printf("\nErro na alocacao de memoria\n");
		printf("\n<Prima qualquer tecla>\n");
		getch();
		exit(1);
	}
	printf("\n");
	printf("\nTamanho = %d\n", ListSize(list));
	printf("ALUNOS:\n");
	ShowValues(list);
	printf("\n");

	ptr_aluno = (ALUNO *)malloc(sizeof(ALUNO));
	printf("Introduza o nome do aluno: ");
	flushall();
	gets_s(ptr_aluno->nome);
	printf("Introduza a idade do aluno: ");
	scanf("%d", &(ptr_aluno->idade));
	if (InsertEnd(&list, ptr_aluno) == ERROR)
	{
		printf("\nErro na alocacao de memoria\n");
		printf("\n<Prima qualquer tecla>\n");
		getch();
		exit(1);
	}
	printf("\n");
	printf("\nTamanho = %d\n", ListSize(list));
	printf("ALUNOS:\n");
	ShowValues(list);
	printf("\n");

	strcpy(((ALUNO *)DATA(list))->nome, "Desconhecido 1");
	//	strcpy(((ALUNO *)(list->data))->nome,"Desconhecido 1");
	((ALUNO *)DATA(list))->idade = 111;
	//	((ALUNO *)(list->data))->idade = 111;

	((ALUNO *)DATA(NEXT(list)))->idade = 222;
	//	((ALUNO *)(list->next->data))->idade = 222;

	((ALUNO *)DATA(NEXT(NEXT(list))))->idade = 333;
	//	((ALUNO *)(list->next->next->data))->idade = 333;
```

```
	printf("\n");
	printf("\nTamanho = %d\n", ListSize(list));
	printf("ALUNOS:\n");
	ShowValues(list);
	printf("\n");

	//Nunca fazer o seguinte segmento de código. Porquê?
	while (NEXT(list) != NULL)
		list = NEXT(list);

	strcpy(((ALUNO *)DATA(list))->nome, "Desconhecido 3");

	printf("\n");
	printf("\nTamanho = %d\n", ListSize(list));
	printf("ALUNOS:\n");
	ShowValues(list);
	printf("\n");

	getch();
	return 0;
}

/*************************************************************
* Funcao: Inicializa a lista
*
* Parametros: list - apontador para lista
*                    (duplo apontador para o primeiro no')
* Saida:      void
*************************************************************/
void initList(LIST *list)
{
	*list = NULL;
}

/*************************************************************
* Funcao: verifica se a lista é vazia
*
* Parametros: list - apontador para lista
* Saida:      TRUE se a lista for vazia, ERROR caso contrário
*************************************************************/
BOOLEAN emptyList(LIST list)
{
	return (list == NULL) ? TRUE : FALSE;
}

/*************************************************************
* Funcao: Cria um no'
*
* Parametros: data - apontador generico para os dados a inserir no no' criado
* Saida:      apontador para o no' criado ou NULL em caso de erro
*************************************************************/
LIST_NODE *NewNode(void *data)
{
	LIST_NODE *new_node;

	if ((new_node = (LIST_NODE *)malloc(sizeof(LIST_NODE))) != NULL)
	{
		DATA(new_node) = data;
		NEXT(new_node) = NULL;
	}
	return(new_node);
}
```

```c
/************************************************************
* Funcao: Insere um no' no inicio da lista
*
* Parametros: list - apontador para lista
*                    (duplo apontador para o primeiro no')
*             data - apontador generico para os dados a inserir no no' criado
* Saida:      OK se o nó foi inserido na LISTA e ERROR caso contrário
************************************************************/
STATUS InsertIni(LIST *list, void *data)
{
	LIST_NODE *new_node;

	if ((new_node = NewNode(data)) != NULL)
	{
		NEXT(new_node) = *list;
		*list = new_node;
		return OK;
	}
	return ERROR;
}

/************************************************************
* Funcao: Insere um no' no fim da lista
*
* Parametros: list - apontador para lista
*                    (duplo apontador para o primeiro no')
*             data - apontador generico para os dados a inserir no no' criado
* Saida:      OK se o nó foi inserido na LISTA e ERROR caso contrário
************************************************************/
STATUS InsertEnd(LIST *list, void *data)
{
	LIST_NODE *new_node, *temp;

	if ((new_node = NewNode(data)) != NULL)
	{
		if (*list == NULL)
			*list = new_node;
		else
		{
			temp = *list;
			while (NEXT(temp) != NULL)
				temp = NEXT(temp);
			NEXT(temp) = new_node;
		}
		return(OK);
	}
	return(ERROR);
}
```

```c
/*****************************************************************************
* Funcao: calcula quantos elementos contem a lista
*
* Parametros: list - apontador para uma lista
* Saida:      numero de elementos da lista
*****************************************************************************/
int ListSize(LIST list)
{
	int count = 0;

	while (list != NULL)
	{
		count++;
		list = NEXT(list);
	}
	return count;
}

/*************************************************************
* Funcao: Escreve no ecra o conteudo da lista
*
* Parametros: list - apontador para o primeiro no'
* Saida:      void
*************************************************************/
void ShowValues(LIST list)
{
    if (emptyList(list) == TRUE)
    {
       printf("\nLista vazia.\n");
       return;
    }

    printf("LISTA ->");
    while (list != NULL)
    {
       printf(" %s;%d ->", ((ALUNO *)DATA(list))->nome, ((ALUNO *)DATA(list))->idade);
       list = NEXT(list);
    }
    printf(" FIM");
}
```