MANIPULANDO LISTA LIGADA EM C

Instruções:

Vimos que a lista ligada ´e uma estrutura de dados dinâmica que consiste em uma sequência de elementos chamados de NÓS, onde cada NÓ contém um valor e um ponteiro para o próximo NÓ na sequência.

Você está a receber 5 códigos com as seguintes funcionalidades:

Criar uma lista

Inserir uma lista

obter uma lista

deletar um elemento da lista

liberar lista

Neste sentido pede-se:

Faça o "esquema" de memória de cada programa. Deverá ser demonstrado o estado da memória para cada programa.

Junte todos os programas em um único programa, que recebe uma função da entrada e processa:

0 = Sair

1 - Incluir

2 - Consultar

3 - Deletar

4 - Listar todos

*CÓDIGO ARQUIVADO (GITHUB)

* TABELA DE MEMÓRIA ARQUIVADO (GITHUB)

Tabela de Memória:

LINHA DE COMANDO	ALTERAÇÃO	LISTA	opt	valor	posicao	contador	numero	atual	anterior	prox	no*p	proximo
		main()	opcao()	trataOpcao()	trataOpcao()	Consultar()	CriarNo()	Consultar() | Deletar(	Deletar()	Struct No()	ListarTodos()	LiberarLista()
		0x1000	0x2000	0x3000	0x4000	0x5000	0xABCD	0x6000	0x7000	0x8000	0x9000	0x9200
17	int numero	0x1000	—	—	—	—	?	—	—	0x8000	—	—
18	struct No *prox	0x1000	—	—	—	—	?	—	—	0x8000	—	—
19	struct No *prox	0x1000	—	—	—	—	?	—	—	0x8000	—	—
30	int*opt	0x1000	?	—	—	—	?	—	—	0x8000	—	—
31	opt, int valor, int posi	0x1000	?	?	?	—	?	—	—	0x8000	—	—
39	int *opt	0x1000	?	—	?	—	?	—	—	0x8000	—	—
51	opt, int valor, int posi	0x1000	?	?	?	—	?	—	—	0x8000	—	—
53	opt 1	0x1000	1	?	?	—	?	—	—	0x8000	—	—
58	opt 2	0x1000	2	?	?	—	?	—	—	0x8000	—	—
63	opt 3	0x1000	3	?	?	—	?	—	—	0x8000	—	—
68	opt 4	0x1000	4	?	?	—	?	—	—	0x8000	—	—
71	opt 5	0x1000	5	?	?	—	?	—	—	0x8000	—	—
93	int posicao	0x1000	2	?	?	—	?	—	—	0x8000	—	—
94	int contador	0x1000	2	?	?	?	?	—	—	0x8000	—	—
108	int valor	0x1000	3	?	?	?	?	—	—	0x8000	—	—
109	no*atual=LISTA	0x1000	3	?	?	?	?	0x1000	—	0x8000	—	—
110	no*anterior=NULL	0x1000	3	?	?	?	?	0x6000	—	0x8000	—	—
134	no*p	0x1000	4	?	?	?	?	0x6000	NULL	0x8000	0x9000	—
141	no*atual=LISTA	0x1000	4	?	?	?	?	0x1000	NULL	0x8000	0x9000	—
142	no*proximo	0x1000	4	?	?	?	?	0x1000	NULL	0x8000	0x9000	0x9200
146	atual*proximo	0x1000	4	?	?	?	?	0x9080	NULL	0x8000	0x9000	0x9200
161	int opt	0x1000	?	?	?	?	?	0x9080	NULL	0x8000	0x9000	0x9200

	Nome da variável
	Local no Programa
	Endereço na memória
	Provoca alteração na memória

0x1000	0x1200	0x14000	0x1600	0x1800	0x2000	0x2200	0x2400	0x2600	0x2800	0x3000	0x3200	0x3400
LISTA					opt					valor		
0x1000					?					?		

0x3600	0x3800	0x4000	0x4200	0x4400	0x4600	0x4800	0x5000	0x5200	0x5400	0x5600	0x5800	0x6000
		posicao					contador					atual
		?					?					0x9080

0x6200	0x6400	0x6600	0x6800	0x7000	0x7200	0x7400	0x7600	0x7800	0x8000	0x8200	0x8400	0x8600
				anterior					prox			
				NULL					0x8000			

0x8800	0x9000	0x9200	0x9400	0x9600	0x9800	0x10000	0x10200	0x10400	0x10600	0x10800	0x11000	0x11200
	no*p	proximo										
	0x9000	0x9200										

	Nome da variável
	Local no Programa
	Endereço na memória
	Valor FINAL da memória

Código:

```
/*_____________________________________________________________________________
_________*/
/*                                                                             */
/*   Fatec - São Caetano Do Sul              Estrutura de Dados Profº Veríssimo    */
/*                          Sarah Melo                                         */
/*                  Objetivo: Manipulando Lista Ligada                         */
/*                  Paradigma: Usar nó e ponteiros                             */
/*                                                                             */
/*                                          02/04/24            */
/*_____________________________________________________________________________
_________*/


#include <stdio.h>
#include <stdlib.h>


#define tamanho_array 5
```

```
struct No {
    int numero;
    struct No *prox;
};

typedef struct No no;

// Funções
void CriarNo(no * LISTA);
no* Incluir();
int Consultar();
no* Deletar();
void ListarTodos();
void LiberarLista();
void opcao(no * LISTA, int *opt);
void trataOpcao(no * LISTA, int opt, int valor, int posicao);

// CRIAÇÃO DO NÓ
void CriarNo(no *LISTA) {
    LISTA->prox = NULL;
}

// MENU DE OPÇÕES
void opcao(no *LISTA, int *opt) {
    printf("\nO que deseja fazer?\n");
    printf("1 - Incluir\n");
    printf("2 - Consultar\n");
    printf("3 - Deletar\n");
    printf("4 - Listar todos\n");
    printf("5 - Sair\n");

    scanf("%d", opt);
}

// TRATAR AS OPÇÕES
void trataOpcao(no *LISTA, int opt, int valor, int posicao) {
```

```
    switch (opt) {
        case 1:
            printf("Digite o valor a incluir: ");
            scanf("%d", &valor);
            LISTA->prox = Incluir(LISTA->prox, valor);
            break;
        case 2:
            printf("Digite a posição a consultar: ");
            scanf("%d", &posicao);
            printf("Valor na posição %d: %d\n", posicao, Consultar(LISTA, posicao));
            break;
        case 3:
            printf("Digite o valor a deletar: ");
            scanf("%d", &valor);
            LISTA->prox = Deletar(LISTA->prox, valor);
            break;
        case 4:
            ListarTodos(LISTA->prox);
            break;
        case 5:
            LiberarLista(LISTA->prox);
            exit(0);
            break;
        default:
            printf("Opção inválida\n");
    }
}

// INCLUIR COMEÇO DA LISTA
no *Incluir(no *LISTA, int valor) {
    no *novoNo = (no *)malloc(sizeof(no));
    if (novoNo == NULL) {
        printf("Erro: Não foi possível alocar memória\n");
        exit(EXIT_FAILURE);
    }
    novoNo->numero = valor;
```

```
    novoNo->prox = LISTA;
    return novoNo;
}

// CONSULTAR A LISTA
int Consultar(no *LISTA, int posicao) {
  int contador = 0;
  no* atual = LISTA;
  while (atual != NULL) {
      if (contador == posicao) {
          return atual->numero;
      }
      atual = atual->prox;
      contador++;
  }
  printf("Erro: Posição inválida\n");
  return 0;
}

// DELETAR VALOR DA LISTA
no *Deletar(no *LISTA, int valor) {
    no *atual = LISTA;
    no *anterior = NULL;

    while (atual != NULL && atual->numero != valor) {
        anterior = atual;
        atual = atual->prox;
    }

    if (atual == NULL) {
        printf("Erro: Elemento não encontrado\n");
        return LISTA;
    }

    if (anterior == NULL) {
        LISTA = atual->prox;
```

```
    } else {
        anterior->prox = atual->prox;
    }

    free(atual);
    return LISTA;
}

// LISTAR TODOS OS VALORES DA LISTA
void ListarTodos(no *LISTA) {
    no *p;
    for (p = LISTA; p != NULL; p = p->prox)
        printf("-> %d\n", p->numero);
}

// LIBERAR LISTA
void LiberarLista(no *lista) {
    no *atual = lista;
    no *proximo;
    while (atual != NULL) {
        proximo = atual->prox;
        free(atual);
        atual = proximo;
    }
}

// MAIN
int main(void) {
    // Inicialização da lista
    no *LISTA = (no *)malloc(sizeof(no));
    if (!LISTA) {
        printf("Erro");
        exit(EXIT_FAILURE);
    }
    CriarNo(LISTA);
```

```
  // Chamando o menu de opções
    int opt;
    do {
        opcao(LISTA, &opt);
        trataOpcao(LISTA, opt, 0, 0);
    } while (opt != 5);

    return 0;
}
```