Atividade B1-4 – manipulando lista ligada

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:

1. Criar uma lista
2. Inserir uma lista
3. obter uma lista
4. deletar um elemento da lista
5. 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 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 fonte: *código no github

```
/*------------------------------------------------------------------------*/
/*   FATEC - São Caetano do Sul              Estrutura de Dados        */
/*                                                                        */
/*                    Camille Guillen                                     */
/*              Objetivo: Manipulando Lista Ligada          */
/*                                                                        */
/*                                        Data:02/04/2024 */
/*------------------------------------------------------------------------*/
#include <stdio.h>
#include <stdlib.h>
```

```
// Definição da estrutura do nó da lista ligada
struct Node {
    int data;
    struct Node* next;
};

// Declaração da lista ligada global
struct Node* head = NULL;

// Função para criar um novo nó
struct Node* createNode(int value) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    if (newNode == NULL) {
        printf("Erro ao alocar memória para o novo nó.\n");
        exit(1);
    }
    newNode->data = value;
    newNode->next = NULL;
    return newNode;
}

// Função para inserir um nó no final da lista
void insertNode(int value) {
    struct Node* newNode = createNode(value);
    if (head == NULL) {
        head = newNode;
    } else {
        struct Node* temp = head;
        while (temp->next != NULL) {
            temp = temp->next;
```

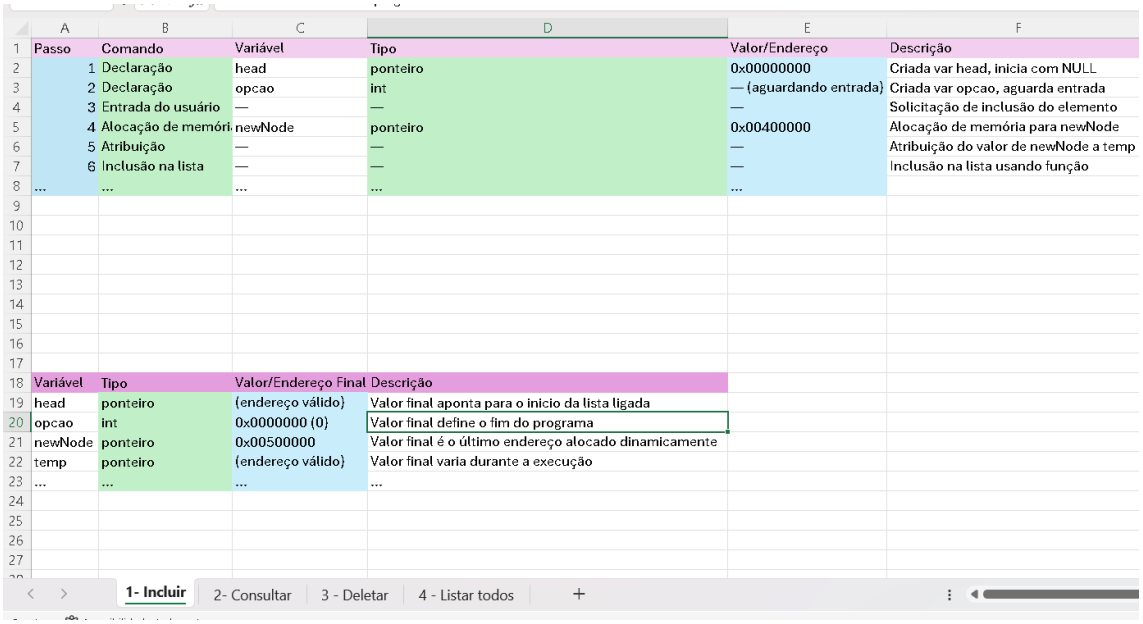
```
        }
        temp->next = newNode;
    }
}

// Função para obter um nó com base no índice (posição) na lista
struct Node* getNode(int index) {
    struct Node* temp = head;
    int count = 0;
    while (temp != NULL) {
        if (count == index) {
            return temp;
        }
        count++;
        temp = temp->next;
    }
    return NULL; // Se o índice estiver fora do intervalo
}

// Função para excluir um nó com base no valor
void deleteNode(int value) {
    struct Node* temp = head;
    struct Node* prev = NULL;

    // Verifica se o nó a ser excluído é o primeiro
    if (temp != NULL && temp->data == value) {
        head = temp->next;
        free(temp);
        return;
    }
```

```
    // Procura pelo nó a ser excluído
    while (temp != NULL && temp->data != value) {
        prev = temp;
        temp = temp->next;
    }

    // Se o nó não estiver presente na lista
    if (temp == NULL) {
        printf("O elemento %d não está presente na lista.\n", value);
        return;
    }

    // Desvincula o nó a ser excluído da lista
    prev->next = temp->next;
    free(temp);
}

// Função para liberar toda a lista
void freeList() {
    struct Node* temp;
    while (head != NULL) {
        temp = head;
        head = head->next;
        free(temp);
    }
}

// Função para listar todos os elementos da lista
void listAllNodes() {
```

```
    struct Node* temp = head;
    if (temp == NULL) {
        printf("A lista está vazia.\n");
        return;
    }
    printf("Elementos da lista:\n");
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}
int main() {
    int opcao, elemento;

    do {
        printf("\nEscolha uma opção:\n");
        printf("0 - Sair\n");
        printf("1 - Incluir\n");
        printf("2 - Consultar\n");
        printf("3 - Deletar\n");
        printf("4 - Listar todos\n");
        scanf("%d", &opcao);

        switch (opcao) {
            case 0:
                printf("Programa encerrado.\n");
                break;
            case 1:
                printf("Digite o elemento a incluir: ");
```

```
            scanf("%d", &elemento);
            insertNode(elemento);
            break;
        case 2:
            printf("Digite o índice do elemento a consultar: ");
            scanf("%d", &elemento);
            if (getNode(elemento) != NULL)
                printf("Elemento encontrado: %d\n", getNode(elemento)->data);
            else
                printf("Índice fora do intervalo.\n");
            break;
        case 3:
            printf("Digite o elemento a deletar: ");
            scanf("%d", &elemento);
            deleteNode(elemento);
            break;
        case 4:
            listAllNodes();
            break;
        default:
            printf("Opção inválida. Tente novamente.\n");
            break;
    }
  } while (opcao != 0);

  // Liberar memória alocada para a lista
  freeList();

  return 0;
}
```

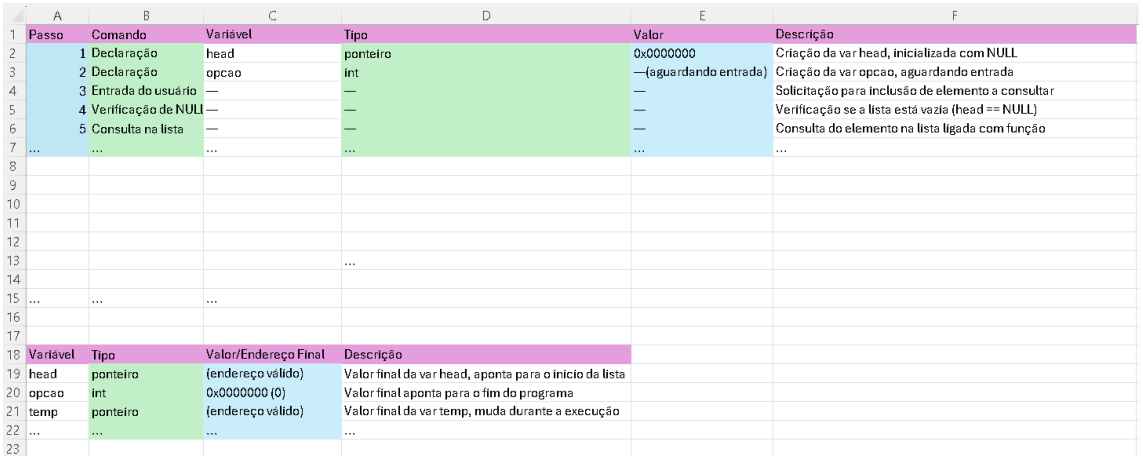
Tabela de Memória: *código no github



Passo	Comando	Variável	Tipo	Valor/Endereço	Descrição
1	Declaração	head	ponteiro	0x00000000	Criada var head, inicia com NULL
2	Declaração	opcao	int	— (aguardando entrada)	Criada var opcao, aguarda entrada
3	Entrada do usuário	—	—	—	Solicitação de inclusão do elemento
4	Alocação de memóri	newNode	ponteiro	0x00400000	Alocação de memória para newNode
5	Atribuição	—	—	—	Atribuição do valor de newNode a temp
6	Inclusão na lista	—	—	—	Inclusão na lista usando função
...	...	...	...	...	

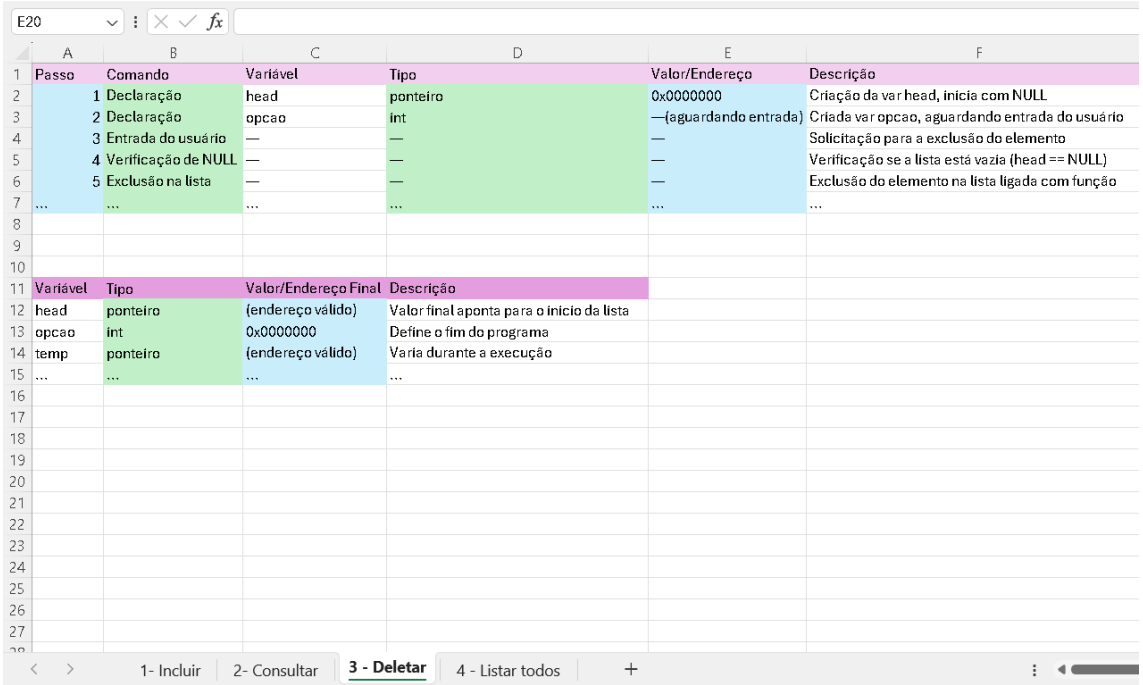
Variável	Tipo	Valor/Endereço Final	Descrição
head	ponteiro	(endereço válido)	Valor final aponta para o inicio da lista ligada
opcao	int	0x0000000 (0)	Valor final define o fim do programa
newNode	ponteiro	0x00500000	Valor final é o último endereço alocado dinamicamente
temp	ponteiro	(endereço válido)	Valor final varia durante a execução
...	...	...	...

1- Incluir | 2- Consultar | 3 - Deletar | 4 - Listar todos



Passo	Comando	Variável	Tipo	Valor	Descrição
1	Declaração	head	ponteiro	0x0000000	Criação da var head, inicializada com NULL
2	Declaração	opcao	int	—(aguardando entrada)	Criação da var opcao, aguardando entrada
3	Entrada do usuário	—	—	—	Solicitação para inclusão de elemento a consultar
4	Verificação de NULL	—	—	—	Verificação se a lista está vazia (head == NULL)
5	Consulta na lista	—	—	—	Consulta do elemento na lista ligada com função
...	...	...	...	...	...

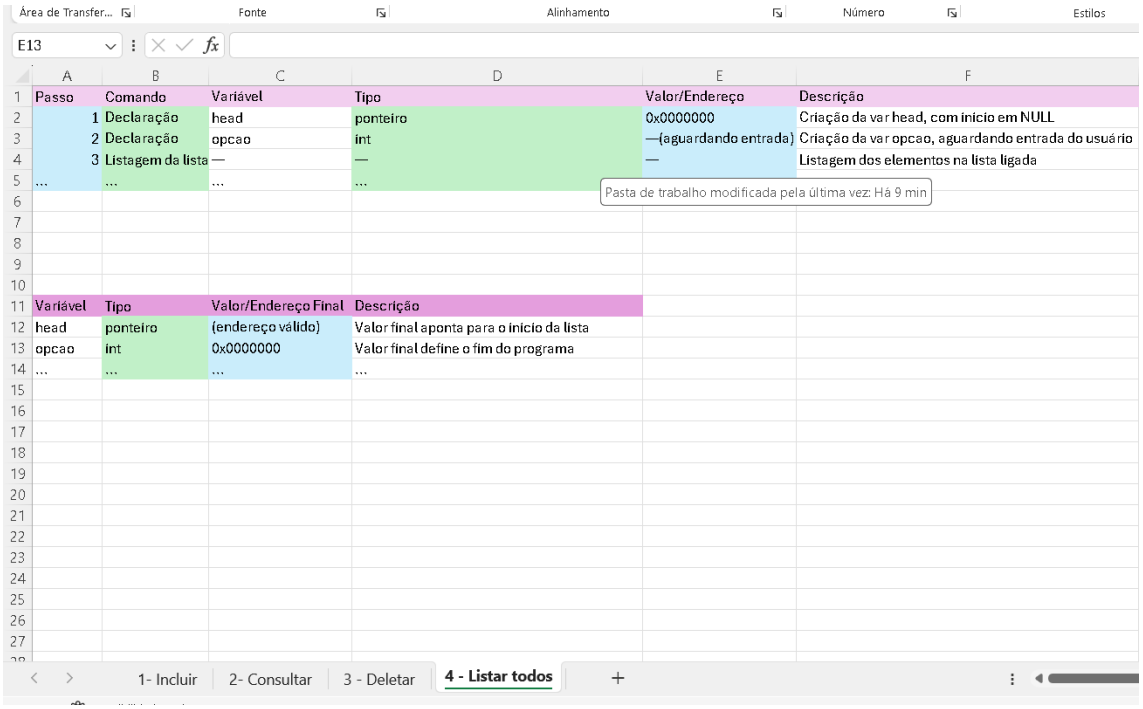
Variável	Tipo	Valor/Endereço Final	Descrição
head	ponteiro	(endereço válido)	Valor final da var head, aponta para o inicio da lista
opcao	int	0x0000000 (0)	Valor final aponta para o fim do programa
temp	ponteiro	(endereço válido)	Valor final da var temp, muda durante a execução
...	...	...	...



Passo	Comando	Variável	Tipo	Valor/Endereço	Descrição
1	Declaração	head	ponteiro	0x0000000	Criação da var head, inicia com NULL
2	Declaração	opcao	int	—(aguardando entrada)	Criada var opcao, aguardando entrada do usuário
3	Entrada do usuário	—	—	—	Solicitação para a exclusão do elemento
4	Verificação de NULL	—	—	—	Verificação se a lista está vazia (head == NULL)
5	Exclusão na lista	—	—	—	Exclusão do elemento na lista ligada com função
...	...	...	...	...	...

Variável	Tipo	Valor/Endereço Final	Descrição
head	ponteiro	(endereço válido)	Valor final aponta para o inicio da lista
opcao	int	0x0000000	Define o fim do programa
temp	ponteiro	(endereço válido)	Varia durante a execução
...	...	...	...

1- Incluir | 2- Consultar | 3 - Deletar | 4 - Listar todos



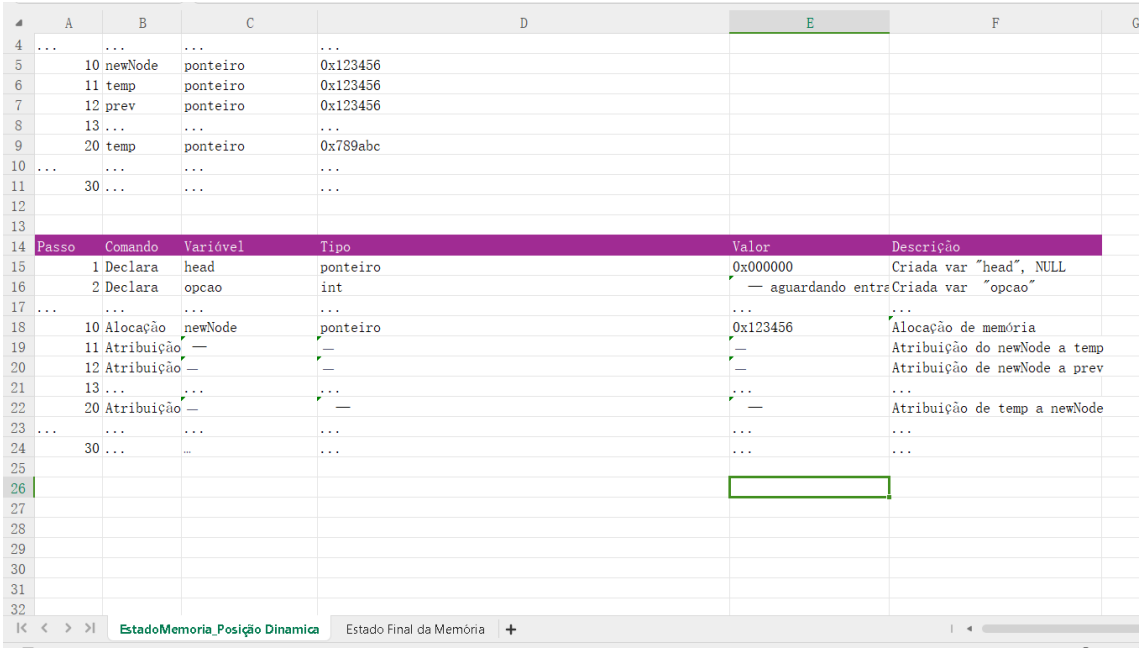
Passo	Comando	Variável	Tipo	Valor/Endereço	Descrição
1	Declaração	head	ponteiro	0x0000000	Criação da var head, com início em NULL
2	Declaração	opcao	int	—(aguardando entrada)	Criação da var opcao, aguardando entrada do usuário
3	Listagem da lista	—	—	—	Listagem dos elementos na lista ligada
...	...	...	...		

Pasta de trabalho modificada pela última vez: Há 9 min

Variável	Tipo	Valor/Endereço Final	Descrição
head	ponteiro	(endereço válido)	Valor final aponta para o início da lista
opcao	int	0x0000000	Valor final define o fim do programa
...	...	...	...

1- Incluir | 2- Consultar | 3 - Deletar | **4 - Listar todos**

Tabela de Memória (Visão Geral) *código no github



...	...	...	...		
10	newNode	ponteiro	0x123456		
11	temp	ponteiro	0x123456		
12	prev	ponteiro	0x123456		
13	...	...	...		
20	temp	ponteiro	0x789abc		
...	...	...	...		
30	...	...	...		

Passo	Comando	Variável	Tipo	Valor	Descrição
1	Declara	head	ponteiro	0x000000	Criada var "head", NULL
2	Declara	opcao	int	— aguardando entra	Criada var "opcao"
...	...	...	...	...	...
10	Alocação	newNode	ponteiro	0x123456	Alocação de memória
11	Atribuição	—	—	—	Atribuição do newNode a temp
12	Atribuição	—	—	—	Atribuição de newNode a prev
13	...	...	...	...	...
20	Atribuição	—	—	—	Atribuição de temp a newNode
...	...	...	...	...	...
30	...	...	...	...	...

EstadoMemoria_Posição Dinamica | Estado Final da Memória

D17

	A	B	C	D	E	F
1	Variável	Tipo	Valor/Endereço Final			
2	head	ponteiro	NULL			
3	opcao	int	0 (final)			
4	newNode	ponteiro	0x789abc			
5	temp	ponteiro	NULL			
6	prev	ponteiro	0x123456			
7	...	...	...			
8						
9						
10	Variável	Tipo	Valor	Desc		
11	head	ponteiro	0x000000	Valor final da var head, permanece null após a execução		
12	opcao	int	0x000000 (0)	Valor final da var opcao, define o fim do programa		
13	newNode	ponteiro	0x123456	Valor final da var newNode, último endereço alocado dinamicamente		
14	temp	ponteiro	0x000000	Valor final da var temp, muda durante a execução		
15	prev	ponteiro	0x123456	Valor final da var prev, varia durante a execução		
16	...	...	...	...		
17						
18						