Atividade 7 - Listas Duplamente Encadeadas e Pilhas (T2)

15 de outubro de 2025

Problema A

```c
void Insere(TipoItem x, TipoLista *Lista)
{
  /* --- Insere no final ---*/
  Lista->Ultimo->Prox = (TipoApontador) malloc(sizeof(TipoCelula));
  TipoApontador NovoElemento = Lista->Ultimo->Prox;

  NovoElemento->Ant = Lista->Ultimo;
  NovoElemento->Item = x;
  NovoElemento->Prox = NULL;

  Lista->Ultimo = NovoElemento;
}

void Retira(TipoChave chave, TipoLista *Lista, TipoItem *Item)
{
  /* --- Remove o primeiro elemento da lista que contem a chave
     especificada --- */
  TipoApontador Aux;

  if (Vazia(*Lista))
  {
      printf(" Erro: Lista vazia\n");
      return;
  }

  // Percorre a lista procurando pela chave
  Aux = Lista->Primeiro;

  while (Aux != NULL && Aux->Item.Chave != chave)
  {
      Aux = Aux->Prox;
  }

  // Se nao encontrou a chave
  if (Aux == NULL)
  {
      printf(" Erro: Chave %d nao encontrada\n", chave);
      return;
  }

```

```
40    if(Aux->Ant && Aux->Prox){
41        Aux->Ant->Prox = Aux->Prox;
42        Aux->Prox->Ant = Aux->Ant;
43    } else if (Aux->Ant){
44        Aux->Ant->Prox = NULL;
45        Lista->Ultimo = Aux->Ant;
46    } else if (Aux->Prox){
47        Aux->Prox->Ant = NULL;
48        Lista->Primeiro = Aux->Prox;
49    } else {
50        Lista->Primeiro = NULL;
51        Lista->Ultimo = NULL;
52    }
53
54    free(Aux);
55 }
56
57 void ImprimeReverso(TipoLista Lista)
58 {
59    TipoApontador Aux;
60
61    if (Vazia(Lista))
62    {
63        printf("\n");
64        return;
65    }
66
67    Aux = Lista.Ultimo;
68
69    // Percorre do ultimo ate o primeiro elemento (depois da celula cabeca
        )
70    while (Aux != Lista.Primeiro)
71    {
72        printf("%d ", Aux->Item.Chave);
73        Aux = Aux->Ant;
74    }
75    printf("\n");
76 }
```

Problema B

```
1 void Empilha(TipoItem x, TipoPilha *Pilha)
2 {
3   TipoApontador Aux;
4   Aux = (TipoApontador) malloc(sizeof(TipoCelula));
5   Pilha->Topo->Item = x;
6   Aux->Prox = Pilha->Topo;
7   Pilha->Topo = Aux;
8   Pilha->Tamanho++;
9 }
```

Problema C

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct No {
    struct No* anterior;
    char c;
} No;

typedef struct Pilha {
    No* topo;
} Pilha;

Pilha* criaPilha(void){
    Pilha* pilha = (Pilha*) malloc (sizeof(Pilha));

    if(!pilha){
        printf("Erro! Nao foi possivel alocar a pilha na memoria!\n");
        return NULL;
    }

    pilha->topo = NULL;
    return pilha;
}

int pilhaVazia(Pilha* pilha){
    if(!pilha || !pilha->topo) return 1;
    return 0;
}

void empilha(Pilha* pilha, char c){
    if(!pilha){
        printf("Erro! A pilha nao existe!\n");
        return;
    }

    No* no = (No*) malloc (sizeof(No));

    if(!no){
        printf("Erro! Nao foi possivel alocar no na memoria!\n");
        return;
    }

    no->anterior = pilha->topo;
    no->c = c;
    pilha->topo = no;
}

char desempilha(Pilha* pilha){
    if(!pilha){
        printf("Erro! A pilha nao existe!\n");
        return EOF;
    }

    if(pilhaVazia(pilha)){
        printf("Erro! Nao e possivel desempilhar uma pilha vazia!\n");
```

```
57          return EOF;
58      }
59
60      No* aux = pilha->topo->anterior;
61      char c = pilha->topo->c;
62      free(pilha->topo);
63      pilha->topo = aux;
64
65      return c;
66  }
67
68  void destroiPilha(Pilha *pilha){
69      if(!pilha) return;
70
71      while(pilha->topo){
72          desempilha(pilha);
73      }
74
75      free(pilha);
76  }
77
78  int main(){
79      int n;
80      scanf("%d\n", &n);
81
82      for(int i = 0; i < n; ++i){
83          char expressao[1001];
84          scanf("%[^\n]\n", expressao);
85
86          Pilha *pilha = criaPilha();
87
88          int correta = 1;
89
90          int tam = strlen(expressao);
91          for(int j = 0; j < tam; ++j){
92              if(expressao[j] == '('){
93                  empilha(pilha, expressao[j]);
94              } else if (expressao[j] == ')'){
95                  if(pilhaVazia(pilha)){
96                      correta = 0;
97                  } else {
98                      desempilha(pilha);
99                  }
100             }
101         }
102
103         if(correta && !pilhaVazia(pilha))
104             correta = 0;
105
106         if(correta){
107             printf("correta\n");
108         } else {
109             printf("incorreta\n");
110         }
111
112         destroiPilha(pilha);
113     }
114
```

```
115     return 0;
116 }
```

Problema D

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

typedef struct No {
    struct No *anterior;
    int valor;
} No;

typedef struct Pilha {
    No* topo;
    int soma;
    int soma_maxima;
} Pilha;

Pilha* criaPilha(void){
    Pilha* pilha = (Pilha*) malloc (sizeof(Pilha));

    if(!pilha){
        printf("Erro! Nao foi possivel alocar memoria para a pilha!\n");
        return NULL;
    }

    pilha->topo = NULL;
    pilha->soma = 0;
    pilha->soma_maxima = pilha->soma;

    return pilha;
}

void empilha(Pilha* pilha, int valor){
    if(!pilha){
        printf("Erro! A pilha nao existe.\n");
        return;
    }

    No *no = (No*) malloc (sizeof(No));

    if(!no){
        printf("Erro! Nao foi possivel alocar o no.\n");
        return;
    }

    no->valor = valor;
    no->anterior = pilha->topo;
    pilha->topo = no;
    pilha->soma += no->valor;

    pilha->soma_maxima = pilha->soma > pilha->soma_maxima ? pilha->soma
        : pilha->soma_maxima;
}

void desempilha(Pilha* pilha){
    if(!pilha){
        printf("Erro! A pilha nao existe.\n");
        return;
    }
```

```
56
57      if(!pilha->topo){
58          printf("Erro! A pilha ja esta vazia.\n");
59          return;
60      }
61
62      No* no = pilha->topo->anterior;
63      pilha->soma -= pilha->topo->valor;
64      free(pilha->topo);
65      pilha->topo = no;
66  }
67
68  void destroiPilha(Pilha* pilha){
69      if(!pilha) return;
70
71      while(pilha->topo){
72          desempilha(pilha);
73      }
74
75      free(pilha);
76  }
77
78  int main(){
79      int n;
80      scanf("%d", &n);
81
82      Pilha* pilha = criaPilha();
83
84      for(int i = 0; i < n; ++i){
85          int op;
86          scanf("%d", &op);
87
88          switch(op){
89              case 0: {
90                  int valor;
91                  scanf("%d", &valor);
92
93                  empilha(pilha, valor);
94                  break;
95              }
96              case 1: {
97                  if(!pilha->topo){
98                      printf("Erro! A pilha esta vazia.\n");
99                      continue;
100                 }
101
102                 empilha(pilha, pilha->topo->valor * 2);
103                 break;
104             }
105             case 2: {
106                 desempilha(pilha);
107                 break;
108             }
109         }
110     }
111
112     printf("%d\n", pilha->soma_maxima);
113     destroiPilha(pilha);
```

```
114     return 0;
115 }
```