Atividade B1-5 - Transforma Lista Ligada em Pilha

Autores: Pedro Viterbo Zacchi, Rafael Nicolas Campos, Valentina Foltran Carvalho, Victor Hugo Gonçalves e Vinícius da Silva Ramos.

Pontos estruturais adaptados de Lista Ligada para pilha:

- Estrutura dos Dados:

Na lista ligada era representado por um ponteiro para o primeiro nó:

```
Pedido *lista = NULL;
```

Na pilha foi criado um struct específico que contém um ponteiro para o elemento do topo, esse ponteiro representa sempre o último elemento inserido:

```
typedef struct Pilha {
   Pedido *topo;
} Pilha;
```

- Inserção dos dados:

Na lista ligada a inserção é feita adicionando o novo nó no início da lista:

```
void inserirPedido(Pedido **lista, Pedido *novo) {
   novo->proximo = *lista;
   *lista = novo;
}
```

Na pilha, para inserir o novo pedido no topo da pilha, é necessário utilizar o push:

```
void push(Pilha *p, Pedido *novo) {
   novo->proximo = p->topo;
   p->topo = novo;
}
```

- Acesso ao elemento do topo:

Esse passo é executado apenas na implementação por pilha, utilizando uma função Top, que retorna o valor do elemento no índice topo:

```
Pedido* Top(Pilha *pilha) {
   if (pilha->topo == NULL) {
      printf("Erro: a pilha está vazia\n");
      return NULL;
   }
   return pilha->topo;
}
```

- Remoção dos elementos:

Na lista ligada a remoção ocorre ao procurar o nó com o número informado e ajusta os ponteiros:

```
void removerPedido(Pedido **lista, int numero) {
}
```

Na implementação por pilha, a remoção (pop) é sempre feita do elemento que está no topo:

```
void pop(Pilha *p) {
   if (p->topo == NULL) {
      printf("Pilha vazia!\n");
      return;
   }
   Pedido *removido = p->topo;
   p->topo = removido->proximo;
   printf("Pedido #%d removido com sucesso!\n", removido->numero);
   free(removido);
}
```

- Atualização e busca de pedidos:

A lógica não foi alterada, porém na implementação por pilha, as funções adaptadas para percorrer a pilha a partir do topo:

```
Pedido* buscarPedido(Pilha *p, int numero) {
   Pedido *aux = p->topo;
   while (aux != NULL) {
      if (aux->numero == numero) {
         return aux;
      }
      aux = aux->proximo;
   }
   return NULL;
}

void atualizarStatus(Pilha *p, int numero, char novoStatus[]) {
   Pedido *pedido = buscarPedido(p, numero);
   if (pedido != NULL) {
      strcpy(pedido->status, novoStatus);
      printf("Status do pedido %d atualizado para: %s\n", numero, novoStatus);
   } else {
      printf("Pedido não encontrado!\n");
   }
}
```

- Listagem e liberar memória:

A lógica também não foi alterada, porém na implementação por pilha, as funções adaptadas para seguir a lógica pelo topo.

```
void listarPedidos(Pilha *p) {
   if (p->topo == NULL) {
      printf("Nenhum pedido cadastrado.\n");
      return;
   }
   Pedido *aux = p->topo;
   while (aux != NULL) {
      printf("Pedido #%d | Cliente: %s | Prato: %s | Qtd: %d | Status: %s\n",
           aux->numero, aux->cliente, aux->descricao, aux->quantidade, aux->status);
      aux = aux->proximo;
   }
}

void liberarPilha(Pilha *p) {
   while (p->topo != NULL) {
      pop(p);
   }
   free(p);
}
```