Atividade B1-5 - Transforma Lista Ligada em Pilha

Autores: Yan Rogério Dall'Acqua Rodrigues

Pontos estruturais adaptados de Lista Ligada para pilha:
• Estrutura dos Dados:

Na implementação original com **lista ligada**, os pedidos eram gerenciados por um ponteiro para o primeiro nó:

Pedido *lista = NULL;

Agora, com a transformação para **pilha**, é criada uma estrutura específica chamada `Pilha`, contendo um ponteiro para o topo e um campo `tamanho` que armazena a quantidade de elementos:

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

• Inserção dos dados:
Na lista ligada, a função `inserirPedido` adicionava sempre no início da lista. Esse mesmo comportamento é mantido na pilha com a função `push`, que adiciona o novo pedido no topo:

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

Remoção de Elementos (Pop)

Na lista ligada, a remoção podia ser feita com base no número do pedido. Na pilha, a remoção é **sempre feita a partir do topo**, com a função `pop`:

```
Pedido* pop(Pilha* p) {
	Pedido* removido = p->topo;
	p->topo = removido->proximo;
	p->tamanho--;
	return removido;
}
```

Acesso ao Elemento do Topo (Top)

A função `top` permite visualizar o pedido que está no topo da pilha, sem removê-lo:

```
Pedido* top(Pilha* p) {
	return p ? p->topo : NULL;
}
```

·Busca e Atualização de Pedidos

As funções `buscarPedido` e `atualizarStatus` percorrem a pilha **a partir do topo**, mantendo a lógica da lista ligada, apenas adaptando o ponteiro de início:

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

Se o pedido é encontrado, ele pode ser atualizado com um novo status:

```
void atualizarStatus(Pilha* p, int numero, StatusPedido novoStatus) {
	Pedido* pedido = buscarPedido(p, numero);
	if (pedido) pedido->status = novoStatus;
}
```

Listagem e Exibição

A função `listarPedidos` exibe todos os pedidos na pilha, iniciando do topo até o último elemento (base da pilha):

```
void listarPedidos(Pilha* p) {
	Pedido* atual = p->topo;
	while (atual) {
	exibirPedido(atual);
	atual = atual->proximo;
	}
}
```

Liberação de Memória

A função `liberarPilha` remove todos os elementos da pilha e libera a memória alocada:

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

Interação com o Usuário (Menu Principal)

A função `main()` fornece um menu de controle com as opções de inserção, busca, atualização, remoção e visualização da pilha. O fluxo do sistema é simples e intuitivo:

- **1**: Insere um novo pedido
- **2**: Busca um pedido por número
- **3**: Atualiza o status do pedido
- **4**: Remove o pedido do topo
- **5**: Lista todos os pedidos
- **6**: Mostra o pedido no topo da pilha
- **0**: Encerra o programa