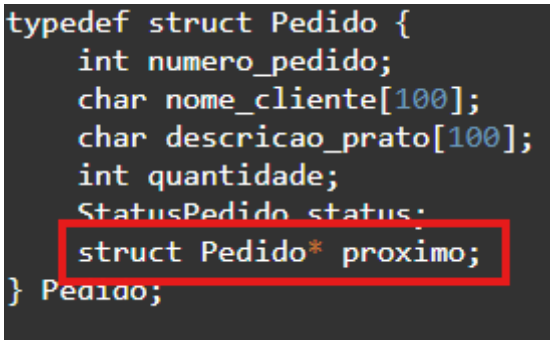
Atividade B1-5 – Transforma Lista ligada em Pilha

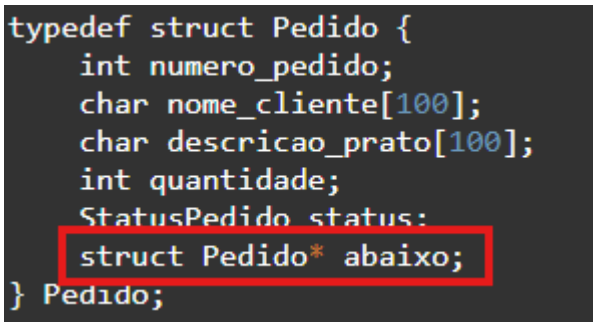
Alterações feitas do código original para o código de Pilha alterado

Uma das principais alterações feitas no código foi a alteração do ponteiro “proximo” por “abaixo”. Antes, no modelo de lista, cada “Pedido” possuía um ponteiro “proximo” que apontava pro próximo pedido na lista.



(Linha 12-19 da 1ª versão)

No modelo atual de pilha, os “Pedidos” tem um ponteiro “abaixo”, que aponta pro pedido anterior, que com base no modelo de pilha, os elementos mais recentes estão no topo.



(Linha 12-19 da 2ª versão)

Funções de Inserção

Antes, o pedido era inserido no final da lista, o que exigia um loop pra encontrar o último elemento na lista para então adicionar o novo pedido.

```
while (temp->proximo != NULL) {
    temp = temp->proximo;
}
temp->proximo = novo_pedido;
```

(Linha 50-53 da 1ª versão)

Agora, a inserção dos elementos é realizada no topo da pilha, ou seja, o novo pedido é colocado no topo, empurrando os outros pedidos para baixo

```
novo_pedido->abaixo = *pilha;
*pilha = novo_pedido;
```
(Linha 46-47 da 2ª versão)

Funções de Deletar

No primeiro código, para remover um pedido, o código procurava o pedido na lista e então ajustava o ponteiro “proximo” do pedido anterior para “pular” o pedido removido.

```
anterior->proximo = temp->proximo;
free(temp);
return 1;
```
(Linha 96-98 da 1ª versão)

Na versão de pilha, para remover um pedio, basta remover o topo da lista, que é o último pedido inserido. O pedido removido é liberado e o topo da lista vai para o próximo elemento.

```
*pilha = temp->abaixo;
free(temp);
```
(Linha 73-74 da 2ª versão)

Acessando Pedidos

Antes era necessário percorrer toda lista para encontrar o pedido com o id correspondente.

```
Pedido* temp = lista;
while (temp != NULL) {
    if (temp->numero_pedido == numero) {
        return temp;
    }
    temp = temp->proximo;
}
```
(Linha 58-62 da 1ª versão)

Agora com a pilha, a busca continua sendo linear, mas é percorrida de cima para baixo, sendo o topo o primeiro e último elemento da pilha.

```
  while (temp != NULL) {
      if (temp->numero_pedido == numero) {
          return temp;
      }
      temp = temp->abaixo;
  }
```
(Linha 52-57 da 2ª versão)

Em questão sobre a liberação de memória em ambos os códigos funcionam da mesma forma, percorrendo a lista/pilha e liberando cada pedido individualmente.

```
    while (lista != NULL) {
        temp = lista;
        lista = lista->proximo;
        free(temp);
    }
```

(Linha 103-107 da 1ª versão)

```
Pedido* temp;
while (pilha != NULL) {
    temp = pilha;
    pilha = pilha->abaixo;
    free(temp);
}
```

(Linha 81-86 da 2ª versão)