Vinícius do Amaral Brunheroto

Estrutura de dados I

Profº Carlos Fischer

Unesp RC

Existem duas formas principais de se pensar e implementar uma pilha usando ponteiros e alocação dinâmica.

Considerando então, a utilização de semelhanças com uma lista encadeada, com uma cabeça (topo) estando ligada ao resto dos elementos da pilha, na forma de uma “cadeia de elementos”.

Na operação de inserção, a primeira forma é sempre inserir elementos antes do antigo topo, o que indica a operação de inserir no início.

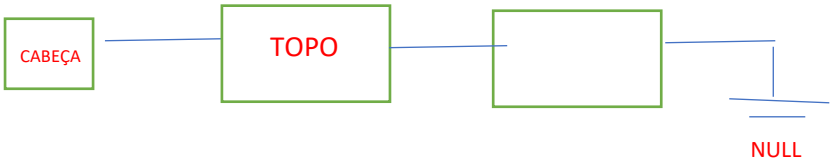
Imaginando isso dentro de uma subrotina, como parâmetros da função há o registro a ser inserido na pilha e um ponteiro para pilha.

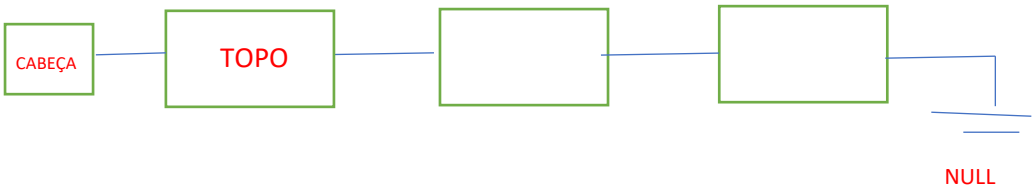
Primeiro , aloca dinamicamente memória para novo elemento, associando o endereço de memória alocado a uma variável..

Verifica-se se a pilha está vazia, caso sim, deverá copiar o registro passado pelo parâmetro nesse novo elemento alocado, indicar que o próximo endereço é NULL e apontar a cabeça da pilha para essa variável alocada.

Caso contrário,(ou seja não está vazia), entra em uma outra condição em que copia-se o registro passado como parâmetro nesse novo elemento alocado e indica que o próximo elemento é quem estava no topo da pilha.

Por fim, aponta-se a cabeça da pilha para essa variável alocada(com registro novo).





A segunda forma é sempre inserir elementos depois do antigo topo, o que indica a operação de inserir no fim. Para isso teremos a variável principal e a variável auxiliar.

Os parâmetros continuam os mesmos.

Primeiro , aloca dinamicamente memória para novo elemento, associando o endereço de memória alocado a variável principal.

Verifica-se se a pilha está vazia, caso sim, deverá copiar o registro passado pelo parâmetro nesse novo elemento alocado, indicar que o próximo endereço é NULL e apontar a cabeça da pilha para essa variável alocado.

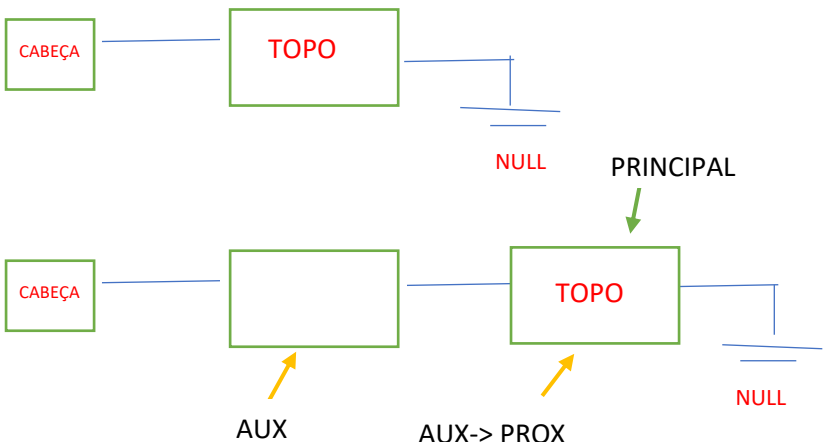
Caso contrário,(ou seja não está vazia), é necessário criar uma variável auxiliar com o mesmo tipo do elemento alocado anteriormente.

Essa variável auxiliar começa apontando para o topo , é inserida dentro de um laço até chegar no último elemento da pilha.

Ao chegar no último elemento , copia-se o registro passado por parâmetro para a variável principal e indica que o próximo endereço dessa variável deverá ser NULL.

Por fim, liga essa variável principal ao próximo endereço apontado pela variável auxiliar.

Com isso a variável auxiliar fica com um papel importante de ligar o novo elemento ao fim da pilha que está sendo formada.



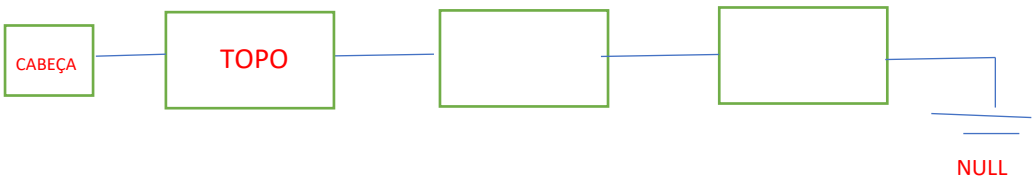
Na operação de remoção, a primeira forma é remover elementos do topo, nesse modelo o topo é sempre o primeiro elemento ,o que indica a operação de remover no início.

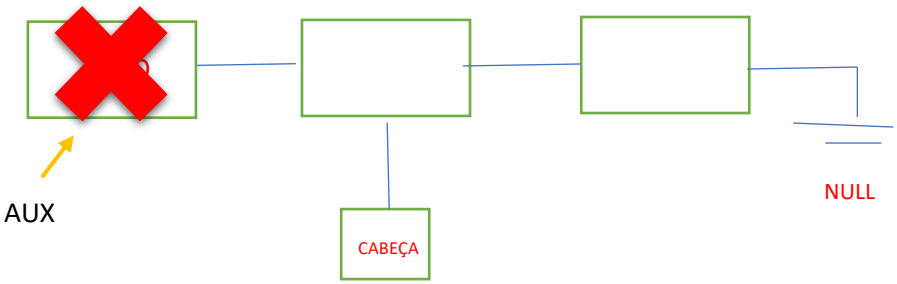
Imaginando isso dentro de uma subrotina, como parâmetros da função há o ponteiro para pilha.

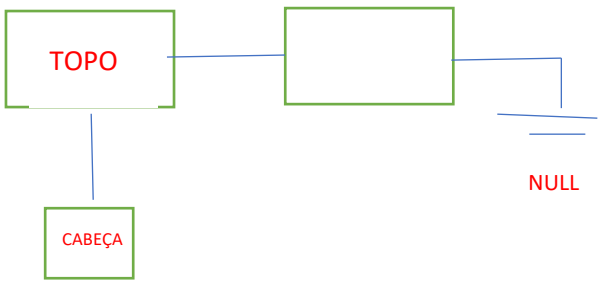
É necessário criar uma variável auxiliar para ajudar a compactar a lista após a retirada do elemento e essa variável já tem que ser iniciada recebendo o endereço de onde a cabeça está apontando, ou seja, o primeiro elemento da pilha.

Primeiro, verifica-se se a pilha está vazia, se estiver não é possível fazer operação de remoção e o ideal seria printar uma mensagem de erro e sair da função.

Se não estiver vazia, como o elemento que se quer retirar é sempre no topo e nesse modelo o topo é sempre o primeiro elemento, então é só apontar a cabeça da lista para o endereço do próximo elemento e com a variável auxiliar liberar a memória da região apontada anteriormente pela cabeça.







A segunda forma também consta de remover elementos do topo, mas agora o elemento que se quer retirar é sempre no fim ,o que indica a operação de remover no fim.

Imaginando isso dentro de uma subrotina, como parâmetros da função há o ponteiro para pilha.

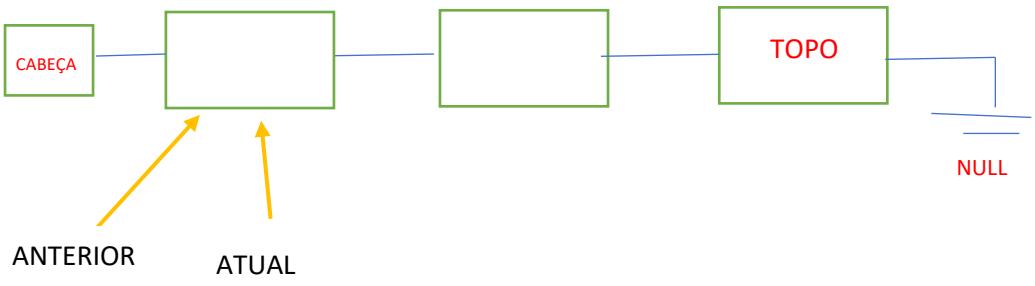
O ideal seria criar duas variáveis auxiliares para ajudar a compactar a lista após a retirada do elemento, um ponteiro (anterior) e outro (atual).

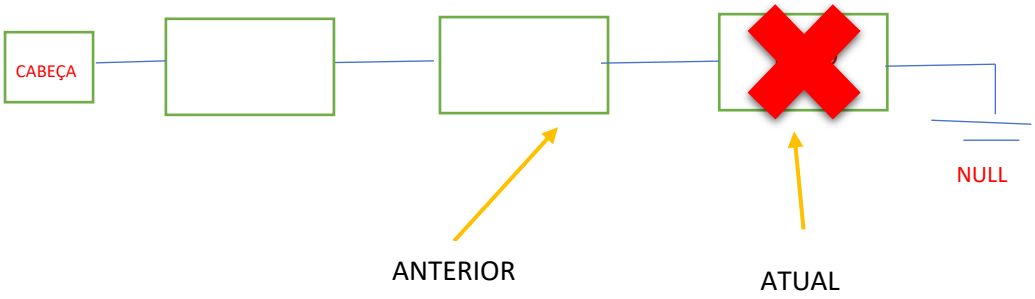
Primeiro, verifica-se se a pilha está vazia, se estiver não é possível fazer operação de remoção e o ideal seria printar uma mensagem de erro e sair da função.

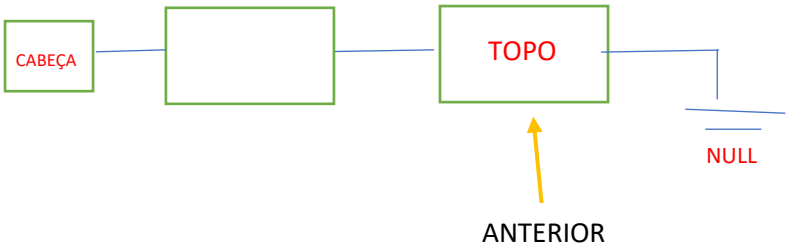
Se não estiver vazia, como o elemento que se quer retirar é sempre no fim, deve-se percorrer a lista utilizando os dois ponteiros, de forma que um ponteiro esteja apontando para um elemento à frente de outro

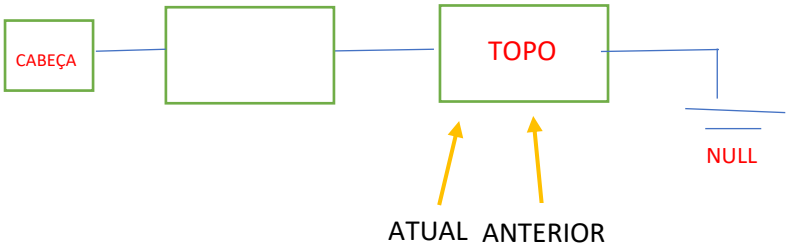
Então, a cada laço o ponteiro anterior recebe o endereço do ponteiro atual e o ponteiro atual avança um endereço.

Quando o ponteiro atual apontar pro endereço do último elemento, o ponteiro anterior estará apontando pro penúltimo elemento, então é só fazer com que o próximo endereço apontado pelo ponteiro anterior seja NULL e liberar a memória do endereço apontado para o ponteiro atual.









(após liberar memória, faz o atual apontar pro endereço de anterior)

Vantagens: qual forma de implementação tem uma aplicação melhor?

A primeira forma em que o topo estará sempre no início tem uma implementação bem mais fácil e prática do que a segunda em que o topo está no final, por não ter que percorrer toda a pilha primeiro, não precisará de tantas variáveis auxiliares.

A primeira forma só se preocupa com o endereço NULL quando inserir o primeiro elemento, caso contrário não precisa se preocupar em apontar elementos para esse endereço.