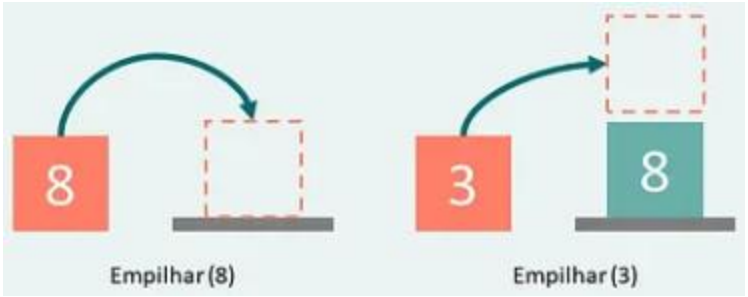


ESTRUTURA DE DADOS ESTÁTICAS

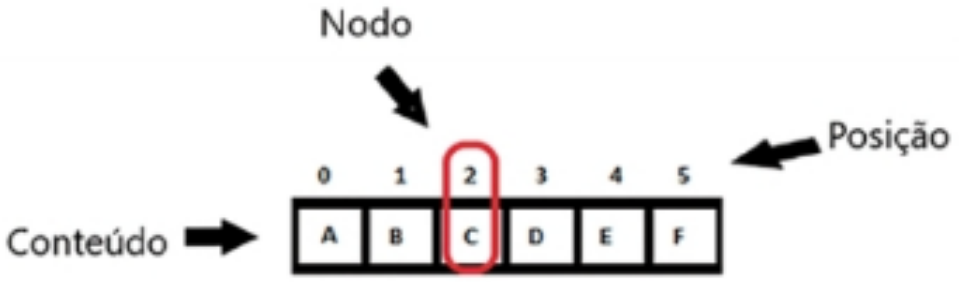
Prof. Dr. Fabio Fernando Kobs

Estruturas Estáticas - Agenda

- Definição
- Vantagens e desvantagens
- Pilha
- Fila
- Lista

Estruturas Estáticas - Definição

- Estrutura de dados primariamente unidimensional que armazena uma quantidade de dados pré determinados em nodos do mesmo tipo, em uma mesma variável, de forma sequencial em memória.
- O vetor é um exemplo de estrutura estática, em que cada posição tem um índice referente à sua localização, localizador do nodo, no qual cada posição é capaz de armazenar um determinado dado.



Estruturas Estáticas - Vantagens

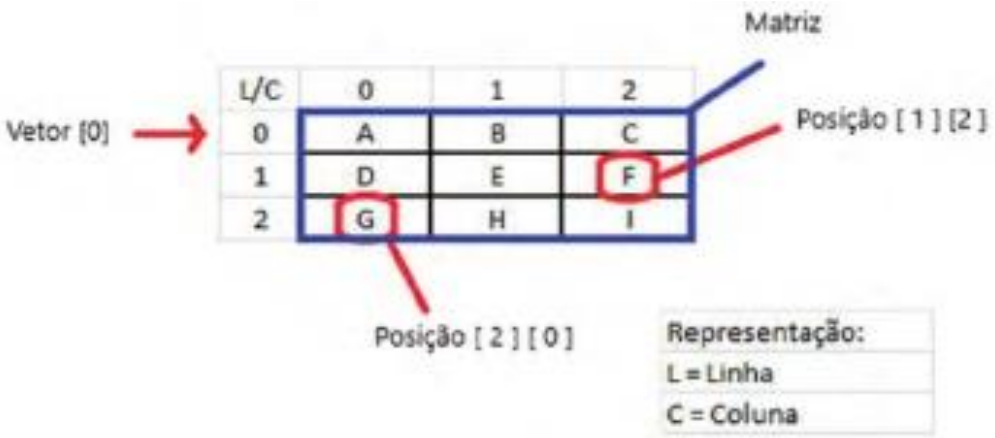
- Fácil implementação comparada a listas encadeadas.
- Redução da quantidade de memória utilizada, pois não requer armazenar a localização do próximo nodo.
- A sua dimensão está disponível sem a necessidade de percorrer todos os nodos.

Estruturas Estáticas - Desvantagens

- Dificuldade na ordenação dos seus elementos.
- O aumento do seu tamanho poderá requerer o deslocamento do vetor em memória, caso não haja espaço de forma linear.
- A impossibilidade de adicionar um elemento no meio do vetor sem realizar o deslocamento de todos os demais nodos, logo, consumindo alto processamento.

Estruturas Estáticas - Matrizes

- Ao utilizar vetores, indiretamente estamos compreendendo a estrutura de uma matriz, muito utilizada no ensino de matemática e aplicada de forma extensiva em análise e controle de equipamentos nos cursos de engenharia, bem como na área de inteligência artificial no processamento de imagens em algoritmos de identificação de objetos.
- Não muito diferente do vetor, a matriz nada mais é que uma lista de vetores, na qual cada linha representa um respectivo vetor e cada posição da matriz tem uma identificação única.



Estruturas Estáticas - Exemplo

- Exemplo de declaração de um vetor, atribuição de conteúdo e impressão em Python:

```
# Inicio
# Declaração de variável - Vetor
funcionario = ['nome','idade','salario']
joao = ['joao', 42, 3000]
maria = ['Maria', 45, 4000]
media = i = 0
# impressão de conteúdo em posição específica
print('alunos',joao[0], 'e', maria[0])
print('Idade', joao[1], 'e', maria[1])
print('Renda', joao[2], 'e', maria[2])
# Impressão do vetor completo
print(funcionario)
# Atribuição de conteúdo em posição específica do vetor
funcionario[0] = 'Lucas'
funcionario[1] = 18
funcionario[2] = 5000
# Cálculo da média de salários
media = (funcionario [2] + joao [2] + maria [2] )/ 3
# Impressão da média de salários
print('A media dos salários e de :', media)
```

Estruturas Estáticas - Exemplo

- Exemplo de declaração de uma matriz, atribuição de conteúdo e impressão em Python:

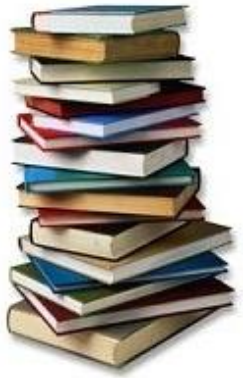
```
# Inicio
# Declaração de uma Matriz
matriz = [['A', 'B', 'C', 'D'],
          ['E', 'F', 'G', 'H'],
          ['i', 'j', 'k', 'l']]
# Impressão de todo o vetor
print("Matriz =", matriz)
# Impressão de cada vetor pertencente a matriz
print("vetor[0] =", matriz[0])
print("vetor[1] =", matriz[1])
print("vetor[2] =", matriz[2])
# Impressão de uma posições específicas de uma matriz
print("posicao[1][2] =", matriz[1][2])
print("posicao[0][-1] =", matriz[0][-1])
```

Estruturas Estáticas - Exemplo

- Exemplo de utilização da biblioteca de lista em Python:

```
# inicio
# Uma nova lista
generos = ['policial', 'indefinido', 'terror', 'suspense', 'amor']
# Varrendo a lista
for prog in generos:
    print (prog)
# Trocando o último elemento
generos[-1] = 'romantico'
# Incluindo
generos.append('drama')
# Removendo
generos.remove('indefinido')
# Imprime numerado
for i, prog in enumerate(generos):
    print (i + 1, '=>', prog)
print (generos)
```

PILHAS - Definição



- Pilhas são coleções lineares em que o acesso é completamente restrito a apenas uma extremidade, chamada topo.
- Ou seja, os novos itens só podem ser adicionados no topo da pilha e também só podem ser removidos os itens do seu topo.
- Tem amplo uso na ciência da computação. A pilha é a coleção mais simples de descrever e implementar.

PILHAS - Definição

- Por esse motivo (adição e remoção de itens no topo), uma pilha é conhecida como uma estrutura de dados *último a entrar, primeiro a sair (last-in, first-out*, LIFO ou último a entrar é o primeiro a sair, UEPS).
- É um tipo abstrato de dados (TAD) – especificação de um conjunto de dados e operações. Características:
  - caracterizado pela separação entre a sua definição conceitual e a implementação propriamente dita;
  - acessado pelo programa por meio de suas operações, pois a sua estrutura de dados nunca é acessada diretamente;
  - programador tem acesso somente a uma descrição dos valores e das operações que são suportados pelo TAD;
  - programador não tem acesso à implementação do TAD, pois ela é invisível ou inacessível, ou seja, está encapsulada.
- Na prática, o ADT é implementado usando-se um tipo composto (estrutura/registro/classe), e por funções que operam esta estrutura.

PILHAS - TAD

Vantagens na utilização dos TAD:

- Reutilização: uma vez que o TAD tenha sido definido, implementado e testado, verificando-se que funciona corretamente e do modo esperado, ele pode ser acessado de diferentes partes do mesmo programa ou de programas diferentes.
- Manutenção facilitada: uma vez que mudanças feitas na implementação do tipo TAD não vão influenciar o código dos programas que o utilizam.
- Código do programa: utiliza os detalhes da implementação do TAD e é minimizado, o que dá maior liberdade para que o programa seja alterado com o mínimo de custo e esforço.
- Segurança: o programador não tem acesso direto aos dados, o que evita que eles sejam manipulados de maneira incorreta.
- Encapsulamento: o conjunto de operações possíveis para o TAD é tudo o que um programador precisa saber sobre ele, sem que seja necessário nenhum conhecimento técnico para que ele seja implementado.

PILHAS - TAD

Um TAD pode, então, ser definido como a união de uma estrutura de dados e das operações definidas para serem feitas com esses dados (Figura 1).

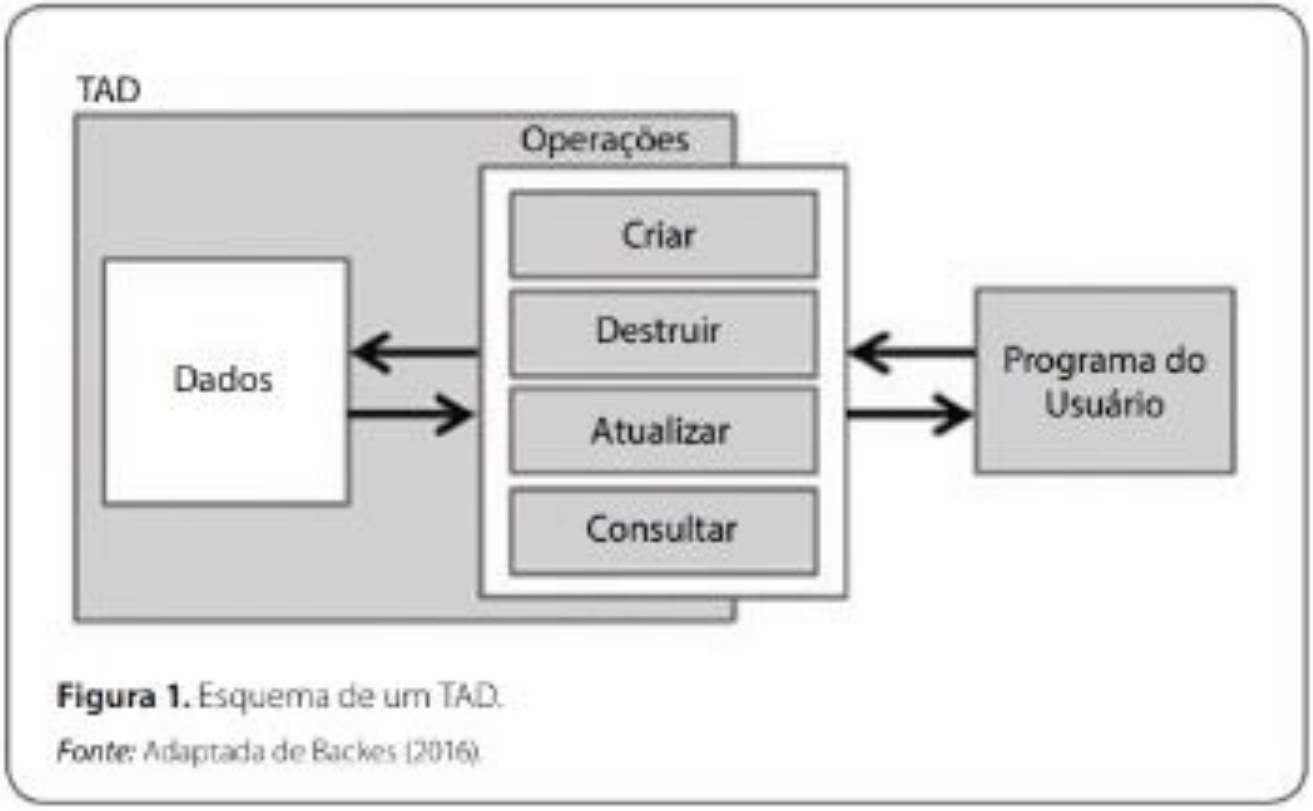
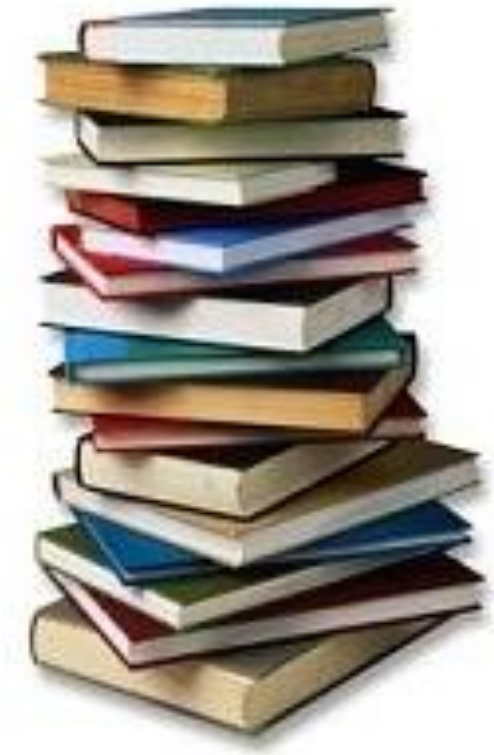


Figura 1. Esquema de um TAD.

Fonte: Adaptada de Backes (2016).

PILHAS - **Exemplo da analogia postal**

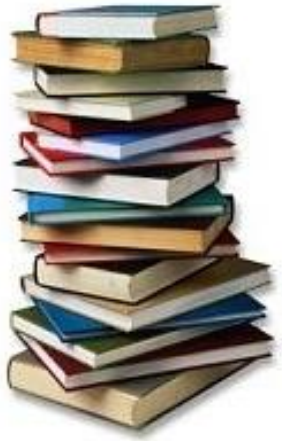


Para entender a ideia de uma pilha, considere uma analogia fornecida pelo serviço postal:

- Muitas pessoas, quando pegam sua correspondência, jogam-na em uma pilha na mesa.
- Então, elas processam a correspondência acumulada de cima para baixo.
- Primeiro, abrem a carta no topo da pilha e tomam a devida ação.
- Depois da primeira carta ter sido liberada, elas examinam a próxima, que é agora o topo da pilha e lidam com ela.
- Finalmente, chegam até a carta do fundo da pilha (que é agora o topo).

PILHAS - **Exemplo da analogia postal**

- Esta abordagem “trabalhe no topo primeiro” funciona bem, contanto que se possa processar facilmente toda a correspondência em tempo razoável.
- Se não puder, há o risco das cartas no fundo da pilha não serem examinadas por meses e as demandas que elas contém ficarem sem atendimento.



PILHAS - Aplicação

- Exemplo 1: Uma pilha é uma ajuda útil para algoritmos aplicados a algumas estruturas de dados complexas, como por exemplo, para percorrer os nós de uma árvore binária.
- Exemplo 2: A maioria dos microprocessadores usa uma arquitetura baseada em pilhas. Quando um método é chamado, seu endereço de retorno e seus argumentos são colocados (empilhados) em uma pilha, e quando ele retorna, eles são retirados (desempilhados), ou seja, utilizado para o controle de sequências de chamadas de subprogramas. As operações de pilha estão embutidas no microprocessador.

PILHAS - Aplicação

- Exemplo 3: Mecanismo de fazer/desfazer de um aplicativo.
- Exemplo 4: As pilhas ocorrem em estruturas de natureza recursiva (como árvores). Ou seja, elas são utilizadas para implementar a recursividade.
- Exemplo 5: Conversão de número decimal para binário.
- Exemplo 6: Mecanismo de navegação de páginas na Internet (avançar e retornar).

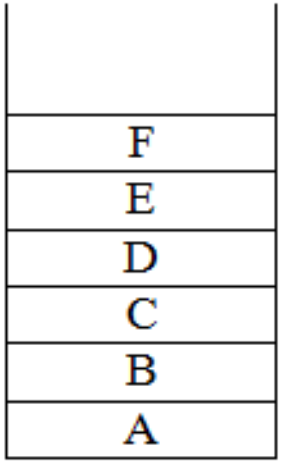
PILHAS - Aplicação

- Exemplo 7: Algumas calculadoras usam uma arquitetura baseada em pilhas. Ao invés de fornecer expressões aritméticas usando parênteses, a calculadora empilha resultados intermediários em uma pilha. Como na HP 12c, se simular: 12+10+8, tem-se:

	10 +		8 +	
12 enter	12	22	22	30

PILHAS - **Representação de uma pilha**

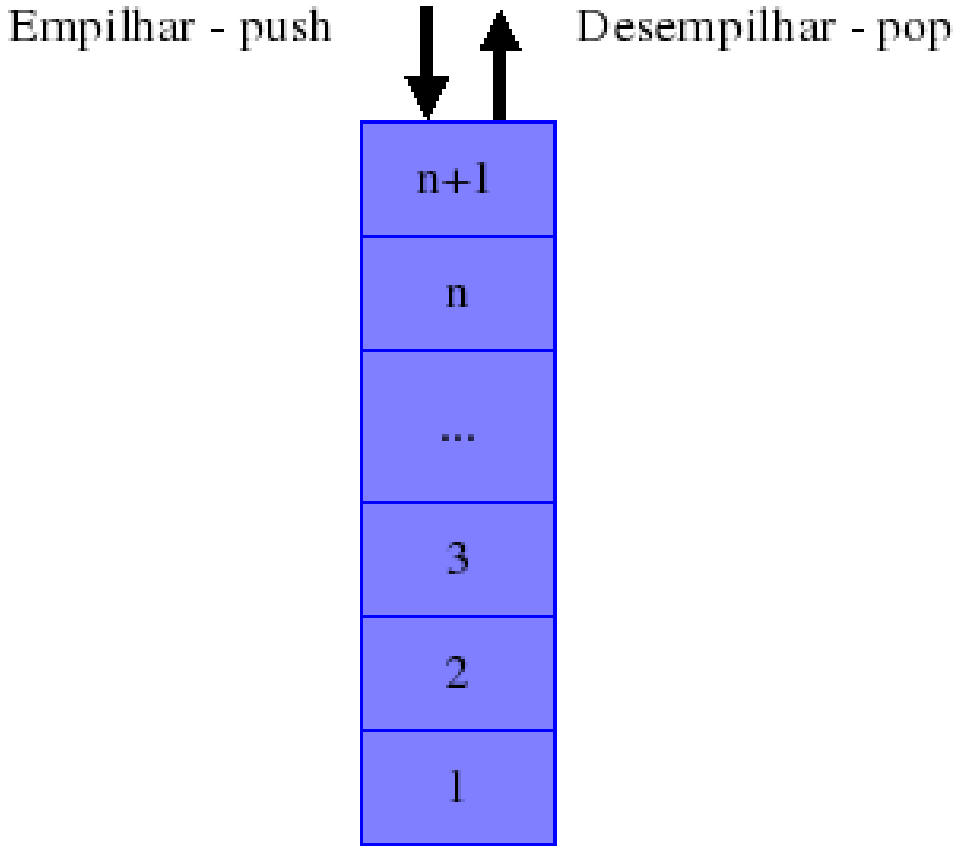
- **Somente uma extremidade da pilha é designada como o topo da pilha**.
- Observando a Figura ao lado, verifica-se que *F* está fisicamente em cima, em relação aos outros itens da pilha.
- Se forem incluídos novos itens na pilha, eles serão colocados acima de *F* e, se forem eliminados alguns itens, *F* será o primeiro a ser eliminado.



PILHAS - **Operações primitivas**

As duas **alterações** que podem ser introduzidas numa pilha recebem nomes especiais:

- Quando um item é incluído numa pilha, ele é **empilhado** sobre a pilha e, quando um item é removido, ele é **desempilhado**.

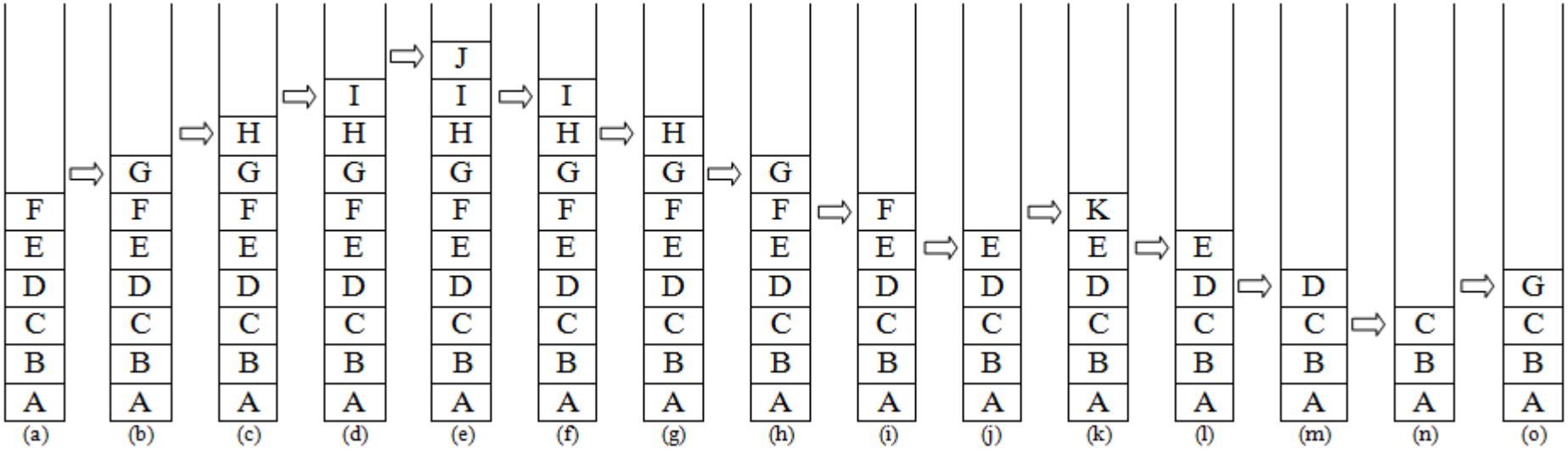


PILHAS - **Operações primitivas**

- Em função de uma pilha *s* e de um item *i*, executar a operação ***push****(s,i)* incluirá o item *i* no topo da pilha *s*.
- De modo semelhante, a operação ***pop****(s)* removerá o elemento superior e o retornará como valor da função. Assim, a operação de atribuição *i=pop(s);* remove o elemento posicionado no topo de *s* e atribui seu valor a *i*.

PILHAS - **Filme de uma pilha**

- Conforme ela se expande e se reduz com o passar do tempo:



- Por exemplo, se *s* for a pilha da Figura acima, executa-se a operação *push(s,G)*, ao passar do quadro **a** para o **b**. Outro exemplo, ao passar do quadro **e** para o **f**, executa-se a operação *pop(s)*.

PILHAS - **Operações primitivas**

- Não há limite máximo para o número de itens que podem ser mantidos numa pilha.
- Se uma pilha contiver um só item e ele for desempilhado, a pilha resultante não conterá itens e será chamada de *pilha vazia*.
  - Neste caso, a operação *pop* não pode ser aplicada à pilha vazia.
  - Portanto, antes de aplicar o operador *pop* a uma pilha, deve-se verificar se ela não está vazia.

PILHAS - **Operação Topo Pilha**

- Outra operação que pode ser executada sobre uma pilha é a determinação do item superior da pilha, sem removê-lo. Essa operação retorna o elemento superior da pilha.
- Também não deve ser executado sob uma pilha vazia.

PILHAS – Usando no Python

- Um tipo de pilha não é integrado ao Python. Se necessário, os programadores Python podem usar uma lista para emular uma pilha baseada em array.
- Se você vir o final de uma lista como o topo de uma pilha, o método *append* de *list* insere um elemento nessa pilha, enquanto o método *pop* de *list* remove e retorna o elemento para o topo.
- A principal desvantagem dessa opção é que todas as outras operações de lista também podem manipular a pilha. Isso inclui inserção, substituição e remoção de um elemento em qualquer posição. Essas operações extras violam o conceito de uma pilha como um tipo de dados abstrato, portanto, usaremos uma interface mais restrita para qualquer implementação de pilha autêntica.

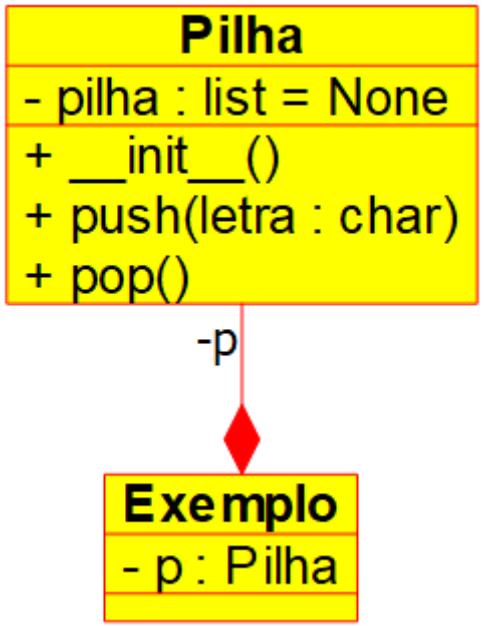
PILHAS – **Exemplo 1 – Python**

```
class Pilha:
    __pilha = None

    def __init__(self):
        self.__pilha=[]

    def push(self, letra):
        self.__pilha.append(letra)

    def pop(self):
        if self.__pilha:
            del self.__pilha[-1] # ou self.__pilha.pop()
        else:
            print("Pilha vazia")
##########################################################################
p=Pilha()
while True:
    op = int(input("\n1- Empilhar\n2- Desempilhar\n3- Sair\n\nOpção: "))
    if op == 1:
        p.push(input("Digite uma letra para empilhar: "))
    elif op == 2:
        p.pop()
    else:
        break
```



PILHAS – EXERCÍCIOS

01) De acordo com o exemplo 1, implemente uma função para imprimir todos os itens da pilha; e outra função para imprimir o topo da pilha. Lembre-se, o acesso se dá somente ao elemento topo da pilha.

02) Construa um programa com as seguintes opções no menu:

1 – Incluir novo nome na pilha

2 – Excluir o nome do topo da pilha

3 – Imprimir o nome do topo da pilha

4 – Imprimir todos os nomes da pilha

5 – Excluir todos os nomes da pilha

PILHAS – **EXERCÍCIOS**

03) Construa um programa para ler uma palavra, imprimindo-a em ordem inversa. Uma pilha de letras será usada para inverter a palavra. Primeiro, os caracteres são extraídos um por um da cadeia de entrada e empilhados. Então eles são desempilhados e exibidos. Por causa da característica LIFO, a pilha inverte a ordem dos caracteres.

04) Escreva um programa usando pilhas para determinar se uma *string* é um palíndromo.

PILHAS
REFERÊNCIAS

- DEITEL, H. M.; DEITEL, P. J. **Como programar em C**. 2. ed. Rio de Janeiro: LTC – Livros Técnicos e Científicos Editora S.A., 1999. ISBN 85-216-1191-9.
- LAFORE, R. **Estruturas de dados & algoritmos em Java**. Rio de Janeiro: Editora Ciência Moderna Ltda., 2004. ISBN 85-7393-375-5.
- LAMBERT, Kenneth A. **Fundamentos de Python**: estruturas de dados. [Digite o Local da Editora]: Cengage Learning Brasil, 2022. E-book. ISBN 9786555584288. Disponível em: https://app.minhabiblioteca.com.br/#/books/9786555584288/. Acesso em: 07 mar. 2024.
- PINTO, Rafael A.; PRESTES, Lucas P.; SERPA, Matheus da S.; et al. **Estrutura de dados**. [Digite o Local da Editora]: Grupo A, 2020. E-book. ISBN 9786581492953. Disponível em: https://app.minhabiblioteca.com.br/#/books/9786581492953/. Acesso em: 06 mar. 2024.
- TENENBAUM, A. M.; LANGSAM, Y.; AUGENSTEIN, M. J. **Estruturas de dados usando C**. Trad. Teresa Cristina Félix de Souza. São Paulo: Makron Books, 1995. ISBN 85-346-0348-0.