**UNIVERSIDADE DE VASSOURAS**

**PRÓ-REITORIA DE CIÊNCIAS TECNOLÓGICAS E EXATAS**

**CURSO DE ENGENHARIA DE SOFTWARE**

# Trabalho P2 – Estrutura de Dados

**Professor: Caio Jannuzzi**

**Aluno: Jeziel Luiz Monteiro Farani | Matrícula: 202313146**

**Aluno: Luiz Henrique Souza do Nascimento | Matrícula: 202310788**

**Vassouras**

**2024**

# Pilhas

São estruturas de dados do tipo LIFO (Last-In First-Out / último a entrar primeiro a sair), onde o último elemento a ser inserido, será o primeiro a ser retirado. Assim, uma pilha permite acesso a apenas um item de dados, o último inserido. Para processar o penúltimo item inserido, deve-se remover o último item.

## Características

**I.** os itens são organizados em uma sequência linear;
**II.** apenas o topo da pilha é acessível;
**III.** inserções e remoções ocorrem apenas no topo;

## Operações Básicas

**empilhar -** inserir um item no topo da pilha;

**desempilhar -** remover um item do topo da pilha;

**ver o topo (top) -** exibir o item que está no topo da pilha;

## Implementação Prática

```python
import numpy as np

class Pilha:
    def __init__(self, capacidade):
        self.__capacidade = capacidade #Define a capacidade da pilha
        self.__topo = -1  #Valor do topo da pilha
        self.__valores = np.empty(self.__capacidade, dtype=int)  #Array de valores

    def __pilha_cheia(self): #Retorna "True" se a pilha estiver cheia
        return self.__topo == self.__capacidade - 1

    def __pilha_vazia(self): #Retorna "True" se a pilha estiver vazia
        return self.__topo == -1

    def empilhar(self, valor): #Insere um valor ao topo da pilha
        if self.__pilha_cheia():
            print("A pilha está cheia")
        else:
            self.__topo += 1
            self.__valores[self.__topo] = valor

    def desempilhar(self): #Desempilha um item da pilha e retorna o valor que agora está no topo da pilha
        if self.__pilha_vazia():
            print("A pilha está vazia")
            return -1
        else:
            valor = self.__valores[self.__topo]
            self.__topo -= 1
            return valor

    def verTopo(self): #Retorna o valor do topo da pilha
        if self.__topo != -1:
            return self.__valores[self.__topo]
        else:
            return -1
```

```
36
37  pilha = Pilha(6)
38
39  print(pilha.verTopo())
40  pilha.empilhar(1)
41  print(pilha.verTopo())
42  pilha.desempilhar()
43  pilha.empilhar(5)
44  pilha.empilhar(6)
45  pilha.empilhar(7)
46  pilha.empilhar(3)
47  pilha.empilhar(4)
48  pilha.empilhar(2)
49  pilha.empilhar(8)
50  print(pilha.verTopo())
51  pilha.desempilhar()
52  print(pilha.verTopo())
53
```

```
PS C:\Users\jezie\Desktop\bagunça\faculdade\programação\python codes\estrutura de dados\P2>  c:; cd 'c:\Users\jezie\Desktop\bagunça\faculdade\programação\python c
odes\estrutura de dados\P2'; & 'c:\Users\jezie\AppData\Local\Microsoft\WindowsApps\python3.12.exe' 'c:\Users\jezie\.vscode\extensions\ms-python.debugpy-2024.6.0-w
in32-x64\bundled\libs\debugpy\adapter/../..\debugpy\launcher' '65338' '--' 'C:\Users\jezie\Desktop\bagunça\faculdade\programação\python codes\estrutura de dados\P
2\pilha.py'
-1
1
A pilha está cheia
2
4
```

## Caso de Aplicações

**I. Gerenciamento de chamadas de funções (recursão) -** quando uma função é chamada, seu estado é empilhado, e quando a função retorna, seu estado é desempilhado; isso permite que as funções sejam aninhadas e controla a ordem de execução;

**II. Resolução de problemas envolvendo parênteses e expressões aritméticas:** ao avaliar expressões matemáticas, as pilhas ajudam a manter a ordem correta das operações;

# Filas

São estruturas de dados do tipo FIFO (First-In First-Out / primeiro a entrar primeiro a sair), onde o primeiro elemento inserido, é o primeiro a ser removido.

## Características

**I.** os itens são organizados em uma sequência linear;
**II.** inserções ocorrem no final da fila;
**III.** remoções ocorrem no início da fila;

## Operações Básicas

**enfileirar -** inserir um item no final da fila;

**desenfileirar -** remover um item do início da fila;

**ver início da fila -** exibir o item que está no início da fila;

## Implementação Prática

```python
import numpy as np

class FilaCircular:
    def __init__(self, capacidade):
        self.capacidade = capacidade #Define a capacidade da fila circular
        self.inicio = 0
        self.final = -1
        self.numero_elementos = 0
        self.valores = np.empty(self.capacidade, dtype=int)

    def enfileirar(self, valor):
        if (self.__fila_cheia()):   #Adiciona um valor ao final da fila circular
            print("A fila está cheia")
            return

        if (self.final == self.capacidade - 1):
            self.final = -1
        self.final += 1
        self.valores[self.final] = valor
        self.numero_elementos += 1

    def desenfileirar(self):  #Remove o primeiro elemento da fila e retorna o segundo valor que agora está no início da fila
        if self.__fila_vazia():
            print("A fila já está vazia")
            return

        temp = self.valores[self.inicio]
        self.inicio += 1
        if (
            self.inicio == self.capacidade
        ):
            self.inicio = 0
        self.numero_elementos -= 1
        return temp

    def __fila_vazia(self): #Retorna "True" se a fila estiver vazia
        return self.numero_elementos == 0

    def __fila_cheia(self): #Retorna "True" se a fila estiver cheia
        return (self.numero_elementos == self.capacidade)

    def primeiro(self):  #Retorna o primeiro valor da fila
        if (self.__fila_vazia()):
            return -1
        return self.valores[self.inicio]
```

```
46
47    fila = FilaCircular(6)
48
49    print(fila.primeiro())
50    fila.enfileirar(5)
51    fila.enfileirar(6)
52    fila.enfileirar(7)
53    fila.enfileirar(3)
54    print(fila.primeiro())
55    fila.enfileirar(4)
56    fila.enfileirar(2)
57    fila.enfileirar(8)
58    print(fila.primeiro())
59    fila.desenfileirar()
60    print(fila.primeiro())
61    fila.desenfileirar()
62    print(fila.primeiro())
63
```

```
PS C:\Users\jezie\Desktop\bagunça\faculdade\programação\python codes\estrutura de dados\P2>  c:; cd 'c:\Users\jezie\Desktop\bagunça\faculdade\programação\python c
odes\estrutura de dados\P2'; & 'c:\Users\jezie\AppData\Local\Microsoft\WindowsApps\python3.12.exe' 'c:\Users\jezie\.vscode\extensions\ms-python.debugpy-2024.6.0-w
in32-x64\bundled\libs\debugpy\adapter/../..\debugpy\launcher' '49952' '--' 'C:\Users\jezie\Desktop\bagunça\faculdade\programação\python codes\estrutura de dados\P
2\fila.py'
-1
5
A fila está cheia
5
6
7
```

## Caso de Aplicações

**I. Agendamento de tarefas em sistemas operacionais:** filas são usadas para gerenciar processos, garantindo que eles sejam executados na ordem em que foram recebidos;

**II. Implementação de algoritmos de busca como o BFS (Breadth-First Search):** o BFS explora os nós em camadas, visitando primeiro os vizinhos mais próximos antes de se mover para os mais distantes;

# Pilhas e Filas: diferenças fundamentais

## Estrutura

**Pilha:** os itens são organizados em uma estrutura LIFO (Last-In First-Out / último a entrar primeiro a sair);

**Fila:** os itens são organizados em uma estrutura FIFO (First-In First-Out / primeiro a entrar primeiro a sair);

## Operações

**Pilha:** suporta operações como "empilhar" para adicionar um elemento ao topo da pilha e "desempilhar" para remover e retornar o elemento do topo da pilha;

**Fila:** suporta operações como "enfileirar" para adicionar um elemento ao final da fila e "desenfileirar" para remover e retornar o elemento do início da fila;