**ATIVIDADE 1**

|  |  |
| --- | --- |
| **Nome: Laura Fantini Souza Ferreira** | **RA: 24183933-5** |
| **Curso: Análise e Desenvolvimento de Sistemas** | |
| **Disciplina: Estruturas de Dados** | |

**Instruções para Realização da Atividade**

1. Todos os campos acima deverão ser devidamente preenchidos.
2. É obrigatória a utilização deste formulário para a realização da MAPA.
3. Esta é uma atividade individual. Caso identificado cópia de colegas, o trabalho de ambos sofrerá decréscimo de nota.
4. Utilizando este formulário, realize sua atividade, salve em seu computador, renomeie e envie em forma de anexo. Antes de selecionar a opção de 'Finalizar' a atividade no sistema, verifique o arquivo anexado, pois arquivos em branco ou incorretos **não** poderão ser substituídos após a finalização.
5. Formatação exigida para esta atividade: documento Word, Fonte Arial ou Times New Roman tamanho 12, Espaçamento entre linhas 1,5, texto justificado.
6. Ao utilizar quaisquer materiais de pesquisa referencie conforme as normas da ABNT.
7. Critérios de avaliação: Utilização do template (Formulário Padrão); Atendimento ao Tema; Constituição dos argumentos e organização das Ideias; Correção Gramatical e atendimento às normas ABNT.
8. Procure argumentar de forma clara e objetiva, de acordo com o conteúdo da disciplina.

**Em caso de dúvidas, entre em contato com seu Professor Mediador.**

**Bons estudos!**

**AGORA É COM VOCÊ!**

As estruturas de dados são fundamentais no desenvolvimento de sistemas computacionais, pois permitem a organização, manipulação e armazenamento eficiente de informações em programas. Segundo Cormen et al. (2012), “estruturas de dados são

formas particulares de organizar dados em um computador para que possam ser utilizados de maneira eficiente”.

Entre as estruturas de dados lineares mais conhecidas estão as Pilhas (Stacks) e as Filas (Queues), ambas largamente utilizadas em linguagens de programação. A Pilha opera sob o princípio LIFO (Last-In, First-Out), ou seja, o último elemento que entra é o

primeiro a sair. Já a Fila segue o princípio FIFO (First-In, First-Out), garantindo que o primeiro elemento inserido seja o primeiro a ser removido (GOODRICH; TAMASSIA; GOLDWASSER, 2014).

Essas estruturas são aplicadas em diversas situações práticas:

Pilhas → Controle de chamadas de funções (recursão), navegadores de internet (histórico), algoritmos de reversão.

Filas → Impressoras, gerenciamento de processos, controle de tarefas e buffers de dados.

O domínio dessas estruturas possibilita ao programador construir algoritmos mais eficientes e adequados às necessidades do problema a ser resolvido. Em ambiente acadêmico, é comum que estudantes comecem a praticar essas estruturas em linguagens

como Python, C, Java, entre outras.

CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. Introduction to Algorithms. 3. ed. The MIT Press, 2012.

GOODRICH, M. T.; TAMASSIA, R.; GOLDWASSER, M. H. Estruturas de Dados e Algoritmos em Python. Rio de Janeiro: LTC, 2014.

Durante os estudos da disciplina de Estrutura de Dados Aplicadas, o aluno Murilo Luz, estudante de um dos cursos de TI EAD Unicesumar, estava estudando as estruturas de Pilha e Fila com base no livro-texto da faculdade.

No entanto, ao tentar copiar os exemplos do livro e montar os códigos em Python no seu computador pessoal, Murilo acabou se confundindo e misturou trechos dos códigos de Pilha e de Fila em uma única classe chamada Estrutura.

Ele enviou esse código misturado para você, pedindo ajuda para reorganizá-lo.

< -- Código misturado -- >

from collections import deque

from typing import Deque

class Estrutura(object):

def \_\_init\_\_(self):

self.elementos = []

self.filaElementos: Deque[int] = deque()

def empilha(self, elemento):

self.elementos.append(elemento)

def desempilha(self):

return self.elementos.pop()

def vazio(self):

return len(self.elementos) == 0

def incluirNaFila(self, elemento):

self.filaElementos.append(elemento)

def incluirMuitosNaFila(self, variosElementos: list):

for elemento in variosElementos:

self.filaElementos.append(elemento)

def retirarDaFila(self):

return self.filaElementos.popleft()

def \_\_str\_\_(self):

retorno = "\nEstrutura Pilha (Topo para Base)\n"

for elemento in self.elementos[::-1]:

indice = self.elementos.index(elemento)

retorno += str(indice) + " - " + str(elemento) + "\n"

retorno += "\nEstrutura Fila (Frente para Trás)\n"

for elemento in self.filaElementos:

indice = self.filaElementos.index(elemento)

retorno += str(indice) + " - " + str(elemento) + "\n"

return retorno

< -- Código Misturado -- >

**Seu desafio como desenvolvedor experiente é ajudar o Murilo**:

a) Identificar no código quais métodos pertencem à Estrutura de Pilha.

b) Identificar no código quais métodos pertencem à Estrutura de Fila.

c) Separar o código em duas classes distintas:

- Classe Pilha (com seus métodos corretos).

- Classe Fila (com seus métodos corretos).

**Orientações Gerais**:

Acesse o link com um vídeo tutorial para ajudá-lo nesse processo de criação e desenvolvimento. O acesso deverá ser realizado no **fórum de Explicações de Atividades.**

**A entrega deve ser feita exclusivamente** por meio do Template de entrega da Atividade de Estudo 1 (AE1) disponível no **Material da Disciplina**;

Lembre-se que evidências de cópias de materiais, incluindo de outros acadêmicos, sem devidas referências, serão inquestionavelmente zeradas. As citações e referências, mesmo que do livro da disciplina, devem ser realizadas conforme normas da

Instituição de Ensino.

Não são permitidas correções parciais no decorrer do módulo, ou seja, o famoso: “professor, veja se minha atividade está certa?”. Isso invalida seu processo avaliativo. Lembre-se que a interpretação da atividade também faz parte da avaliação.

Procure sanar suas dúvidas junto à mediação em tempo hábil sobre o conteúdo exigido na atividade, de modo que consiga realizar sua participação.

Atenção ao prazo de entrega, evite envio de atividade em cima do prazo. Você pode ter algum problema com internet, computador, software etc., e **os prazos não serão flexibilizados**, mesmo em caso de comprovação.

Bons Estudos.

**RESPOSTA:**

A) Na **estrutura de pilha**, (com comportamento LIFO) os métodos presentes são os responsáveis por adicionar, remover elementos e verificar se a pilha está vazia. Os métodos utilizados são:

* **empilha(self, elemento):** adiciona elemento ao topo da pilha.
* **desempilha(self):** remove e retorna o elemento do topo da pilha.
* **vazio(self):** verifica se a pilha está vazia.

Além disso, há um trecho do método \_\_str\_\_ que pertence à **pilha**:

retorno = "\nEstrutura Pilha (Topo para Base)\n"

for elemento in self.elementos[::-1]:

indice = self.elementos.index(elemento)

retorno += str(indice) + " - " + str(elemento) + "\n"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

B) A **estrutura fila** (com comportamento FIFO) contém métodos que permitem inserir elementos no final da fila, inserir vários elementos ao mesmo tempo e remover elementos do início. A fila é como uma fila de supermercado: quem chega primeiro, sai primeiro. Usamos os seguintes métodos:

* **incluirNaFila(self, elemento):** adiciona um elemento ao final da fila.
* **incluirMuitosNaFila(self, variosElementos: list):** adiciona vários elementos ao final da fila.
* **retirarDaFila(self):** remove e retorna o elemento da frente da fila.

Além disso, há um trecho do método \_\_str\_\_ que pertence à **fila**:

retorno += "\nEstrutura Fila (Frente para Trás)\n"

for elemento in self.filaElementos:

indice = self.filaElementos.index(elemento)

retorno += str(indice) + " - " + str(elemento) + "\n"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

C) Separação do código em duas classes distintas:

* **Pilha**

class Pilha:

def \_\_init\_\_(self):

self.elementos = []

def empilhar(self, valor):

"""Insere um novo valor no topo da pilha."""

self.elementos.append(valor)

def retirar(self):

"""Elimina e retorna o elemento do topo da pilha."""

return self.elementos.pop()

def esta\_vazia(self):

"""Confere se a pilha não possui elementos."""

return len(self.elementos) == 0

def \_\_str\_\_(self):

"""Mostra os elementos da pilha, do topo à base."""

representacao = "\nVisualização da Pilha (Topo → Base)\n"

for elemento in self.elementos[::-1]:

indice = self.elementos.index(elemento)

representacao += f"{indice} - {elemento}\n"

return representacao

* **Fila**

class Fila:

def \_\_init\_\_(self):

self.itens: Deque[int] = deque()

def adicionar(self, valor):

"""Coloca um valor ao final da fila."""

self.itens.append(valor)

def adicionar\_varios(self, lista\_valores: list):

"""Inclui múltiplos valores ao final da fila."""

for valor in lista\_valores:

self.itens.append(valor)

def remover(self):

"""Elimina e retorna o valor que está no início da fila."""

return self.itens.popleft()

def \_\_str\_\_(self):

"""Exibe os elementos da fila na ordem: frente → trás."""

representacao = "\nVisualização da Fila (Frente → Trás)\n"

for elemento in self.itens:

indice = self.itens.index(elemento)

representacao += f"{indice} - {elemento}\n"

return representacao