Estrutura de Dados: (Paradigma Imperativo)

Foi utilizada a estrutura de dados Nativa do python {lista}, para implementar o tipo abstrato de conjunto, sejam eles inteiros, strings, listas ....

Algoritmos:

- ehVazio(A): Verifica se o comprimento da lista é zero ("len(A) == 0")
- uniaoConj(A,B): Utiliza a operação de conjunto "set" para realizar a união e converter de volta para lista
- interConj(A,B): Utiliza a operação do conjunto "set" para calcular a intersecção e converter de volta para lista
- diffConj(A,B): Utiliza a operação do conjunto "set" para calcular a diferença e converte de volta para lista
- diffSimConj(A,B): Utiliza a operação de conjunto "set" para calcular a diferença simétrica e converte de volta para lista
- addConj(A, x): Adiciona o elemento x a lista A se não estiver presente
- remConj(A, x): Remove o elemento x da lista A se estiver presente
- tamConj(A): Retorna o comprimento da lista A pós alteração
- ehElemConj(A,x): Verifica se x está presente na lista

. Utilização do Código

```python
# Criando conjuntos A e B a partir da entrada do usuário
entrada_A = input("Digite os elementos do conjunto A separados por espaço: ")
conjunto_A = [converte_elemento(elemento) for elemento in entrada_A.split()]

entrada_B = input("Digite os elementos do conjunto B separados por espaço: ")
conjunto_B = [converte_elemento(elemento) for elemento in entrada_B.split()]

# Exibindo os conjuntos A e B
print("A:", conjunto_A)
print("B:", conjunto_B)

# Realizando as operações automaticamente
print("É vazio (A):", ehVazio(conjunto_A))
print("União de A e B:", uniaoConj(conjunto_A, conjunto_B))
print("Interseção de A e B:", intersConj(conjunto_A, conjunto_B))
print("Diferença A - B:", diffConj(conjunto_A, conjunto_B))
print("Diferença Simétrica A ⊕ B:", diffSimConj(conjunto_A, conjunto_B))

elemento_add = converte_elemento(input("Digite um elemento para adicionar ao conjunto A: "))
addConj(conjunto_A, elemento_add)
print("A após adição:", conjunto_A)

elemento_rem = converte_elemento(input("Digite um elemento para remover do conjunto A: "))
remConj(conjunto_A, elemento_rem)
print("A após remoção:", conjunto_A)

print("Tamanho de A:", tamConj(conjunto_A))

elemento_verificar = converte_elemento(input("Digite um elemento para verificar se está em A: "))
print(f"{elemento_verificar} pertence a A:", ehElemConj(conjunto_A, elemento_verificar))

```

Aqui um exemplo genérico (Depende exclusivamente das alterações do professor) para inserção dos conjuntos sejam eles quaisquer tipos