Resultados

Adeuvaldo Neto Fernandes Paiva

--	02:36
Ponto possível(eis)	Tempo para esta tentativa

Suas respostas:

1

Em qual aplicação do mundo real é comum o uso de uma estrutura de **fila** (FIFO)?

- ◯ Histórico de navegação em um navegador
- ◯ Impressão de documentos em uma impressora compartilhada
- ◯ Verificação de parênteses em expressões
- ◯ Chamadas de função recursiva

(resposta não exibida)

Feedback

Baseado na sua resposta

As impressoras utilizam filas para atender os pedidos na ordem em que foram enviados — comportamento FIFO.

2

Qual estrutura de dados segue a lógica "último a entrar, primeiro a sair" (LIFO)?

- ◯ Lista encadeada
- ◯ Fila
- ◯ Árvore

- ◯ Pilha

(resposta não exibida)

Feedback

Baseado na sua resposta

A pilha segue o comportamento LIFO (Last In, First Out), onde o último elemento inserido é o primeiro a ser removido.

3

O que caracteriza uma função recursiva?

- ◯ Chama várias funções externas
- ◯ É usada apenas para somar listas
- ◯ Chama a si mesma até uma condição de parada
- ◯ Executa somente estruturas de repetição `for`

(resposta não exibida)

Feedback

Baseado na sua resposta

A recursão ocorre quando uma função chama a si mesma, sendo necessário definir um **caso base** para que a repetição pare.

4

O que acontece se uma função recursiva não tiver um caso base definido?

- ◯ Ela termina imediatamente
- ◯ Gera um erro de sintaxe
- ◯ Entra em loop infinito ou causa "estouro de pilha"
- ◯ Executa somente uma vez

(resposta não exibida)

Feedback

Baseado na sua resposta

Sem caso base, a função continua chamando a si mesma indefinidamente, o que leva a um **stack overflow**.

5

Qual estrutura é mais adequada para implementar a funcionalidade "desfazer" (undo) em um editor de texto?

- ◯ Vetor
- ◯ Pilha
- ◯ Fila
- ◯ Hash Table

(resposta não exibida)

Feedback

Baseado na sua resposta

A pilha permite desfazer a última ação executada, pois sempre acessa o último item inserido — comportamento LIFO.

6

A recursividade pode ser substituída por:

- ◯ Pilhas dinâmicas
- ◯ Funções aninhadas
- ◯ Iterações com estruturas como `for` e `while`
- ◯ Múltiplas chamadas de outras funções

(resposta não exibida)

Feedback

Baseado na sua resposta

Quase toda função recursiva pode ser reescrita usando estruturas de repetição (`for`, `while`), embora a forma recursiva possa ser mais elegante para certos problemas.

7

Qual das alternativas abaixo é um exemplo clássico de problema recursivo?

- ◯ Soma de dois números
- ◯ Cálculo de fatorial
- ◯ Impressão de uma lista
- ◯ Troca de valores entre variáveis

(resposta não exibida)

Feedback

Baseado na sua resposta

O cálculo de fatorial é o exemplo mais tradicional de recursão: `n! = n * (n-1)!`

8

O que ocorre internamente na memória quando uma função recursiva é chamada várias vezes?

- ◯ Todas as chamadas são executadas simultaneamente
- ◯ Cada chamada é armazenada em uma pilha de execução (call stack)
- ◯ As chamadas são armazenadas em uma fila de prioridade
- ◯ A função sobrescreve a anterior até o fim do programa

(resposta não exibida)

Feedback

Baseado na sua resposta

Cada chamada recursiva é empilhada na **call stack** (pilha de chamadas), e o programa só "desempilha" quando chega ao caso base, executando as funções na ordem inversa das chamadas.

9

Qual é o comportamento principal de uma **fila de prioridade** em relação a uma fila comum?

- ◯ Os elementos são removidos em ordem aleatória
- ◯ Os elementos são removidos pela ordem de chegada
- ◯ Os elementos são atendidos conforme um critério de importância
- ◯ Os elementos são removidos apenas do final da fila

(resposta não exibida)

Feedback

Baseado na sua resposta

Em uma **fila de prioridade**, a remoção é feita conforme a prioridade associada ao elemento, e não necessariamente pela ordem de chegada (FIFO).

10

Considere a função recursiva que resolve a Torre de Hanói:

```
def hanoi(n, origem, destino, auxiliar):
    if n == 1:
        print(f"Mover disco de {origem} para {destino}")
    else:
        hanoi(n - 1, origem, auxiliar, destino)
        print(f"Mover disco de {origem} para {destino}")
        hanoi(n - 1, auxiliar, destino, origem)
```

Se chamarmos `hanoi(3, 'A', 'C', 'B')`, quantas vezes a função `hanoi` será chamada no total (incluindo a primeira)?

- ◯ 5
- ◯ 7
- ◯ 9
- ◯ 15

(resposta não exibida)

Feedback

Baseado na sua resposta

O padrão de chamadas dessa função é:

```
T(n) = 2 * T(n - 1) + 1
```

onde `T(n)` representa o número total de chamadas da função (incluindo a inicial).

Vamos calcular:

- `T(1) = 1`
- `T(2) = 2*T(1) + 1 = 2*1 + 1 = 3`
- `T(3) = 2*T(2) + 1 = 2*3 + 1 = 7`

Portanto, **hanoi(3)** faz **7 chamadas** no total.

A função da Torre de Hanói é outro exemplo de **recursão com duas chamadas internas**, mas diferente do Fibonacci, seu crescimento segue o padrão $2^n - 1$ — **exponencial**, porém previsível e exato.

Para comparar:

- `hanoi(3)` → 7 chamadas

- `hanoi(4)` → 15 chamadas
- `hanoi(5)` → 31 chamadas