IFCE - Campus Morada Nova
Tecnologia em Análise e Desenvolvimento de Sistemas
Estruturas de Dados
Recursividade

Nível Fácil: Potência de um Número

Crie uma função recursiva `potencia(base, expoente)` que calcule a base elevada ao expoente. Lembre-se, o caso base é quando o expoente é 0, pois qualquer número elevado a 0 é 1.

Esqueleto de Código:

```cpp
#include <iostream>

using namespace std;

int potencia(int base, int expoente) {

    // Caso Base: Complete a condição de parada.

    if (______) {

        return 1;

    }

    // Chamada Recursiva: Complete a linha para calcular a potência.

    return base * ______;

}

int main() {

    int resultado = potencia(2, 3);

    cout << "2^3 = " << resultado << endl; // Espera-se 8

    return 0;

}
```

Nível Intermediário: Busca Binária Recursiva

Implemente uma função recursiva `buscaBinaria(arr[], esquerda, direita, valor)` para encontrar um valor em um array ordenado. A lógica é "dividir e conquistar", explorando apenas a metade correta do array a cada chamada.

Esqueleto de Código:

```cpp
#include <iostream>

using namespace std;

bool buscaBinaria(int arr[], int esquerda, int direita, int valor) {

    // Caso Base de Falha: Complete a condição para quando o valor não for encontrado.

    if (______) {

        return false;

    }

    int meio = esquerda + (direita - esquerda) / 2;

    // Caso Base de Sucesso: O valor foi encontrado.

    if (arr[meio] == valor) {

        return true;

    }

    // Chamadas Recursivas: Complete as condições para buscar na metade correta.

    if (______) {

        return buscaBinaria(arr, esquerda, meio - 1, valor);

    }

    else {

        return buscaBinaria(arr, ______, ______, valor);

    }

}

int main() {

    int arr[] = {2, 5, 8, 12, 16, 23, 38, 56, 72, 91};

    int n = 10;

    if (buscaBinaria(arr, 0, n - 1, 23)) {

        cout << "Valor encontrado!" << endl;

    } else {

        cout << "Valor nao encontrado." << endl;

    }
```

```
    return 0;
}
```

Nível Difícil: Torre de Hanoi

Escreva uma função recursiva que imprima os passos para mover n discos de um pino de origem para um pino de destino. A lógica recursiva para n discos é: (1) Mover n-1 discos da origem para o auxiliar; (2) Mover o disco n da origem para o destino; (3) Mover n-1 discos do auxiliar para o destino.

Esqueleto de Código:

```
#include <iostream>
#include <string>
using namespace std;
void torreDeHanoi(int n, string origem, string destino, string auxiliar) {
    // Caso Base: Complete a condição de parada.
    if (______) {
        return;
    }
    // Chamada Recursiva 1: Mova n-1 discos da origem para o auxiliar.
    torreDeHanoi(______, ______, ______, ______ );
    // Passo 2: Imprima o movimento do disco n.
    cout << "Mova o disco " << n << " da " << origem << " para a " << destino << endl;
    // Chamada Recursiva 2: Mova n-1 discos do auxiliar para o destino.
    torreDeHanoi(______, ______, ______, ______);
}
int main() {
    int nDiscos = 3;
    cout << "Passos para resolver a Torre de Hanoi com " << nDiscos << " discos:\n";
    torreDeHanoi(nDiscos, "Pino A", "Pino C", "Pino B");
    return 0;
}
```