Lekcja – Rekurencja

Rekurencja to koncepcja programowania, która polega na tym, że funkcja wywołuje samą siebie, aż osiągnie określone kryterium zakończenia.
Można to porównać do rozwiązywania problemu za pomocą podobnych, ale mniejszych instancji tego samego problemu. Jest to użyteczne narzędzie w programowaniu do rozwiązywania pewnych typów problemów, takich jak obliczenia matematyczne, operacje na strukturach danych, analiza algorytmów itp.

Zadanie rekurencyjna zazwyczaj składa się z:

Definicja bazowa (base case): Każda rekurencyjna funkcja musi mieć definicję bazową, która jest prostym przypadkiem problemu, który można rozwiązać bez używania rekurencji. To jest punkt, w którym rekurencja kończy się i zaczyna zwracać wyniki. Bez tego funkcja mogłaby wywoływać same siebie w nieskończoność.

Krok rekurencyjny: Oprócz definicji bazowej, funkcja rekurencyjna zawiera również krok rekurencyjny. To jest miejsce, w którym funkcja wywołuje samą siebie, ale z mniejszym lub bardziej ograniczonym problemem. Ten krok rekurencyjny ma na celu rozwiązywanie coraz mniejszych części problemu, aż osiągnie definicję bazową.

Przykład - Obliczanie silni: Jednym z klasycznych przykładów rekurencji jest obliczanie silni liczby. Silnia n! jest zdefiniowana jako iloczyn wszystkich liczb całkowitych od 1 do n. Funkcję obliczającą silnię można zdefiniować rekurencyjnie w ten sposób:

Definicja bazowa: Silnia 0 jest równa 1: 0! = 1.
Krok rekurencyjny: Silnia n jest równa n pomnożonemu przez (n-1)!.
Dla przykładu, 5! = 5 * 4! = 5 * 4 * 3! = 5 * 4 * 3 * 2! = 5 * 4 * 3 * 2 * 1! = 5 * 4 * 3 * 2 * 1 * 0! = 120.

Widzisz, że funkcja obliczająca silnię wywołuje samą siebie, ale za każdym razem z mniejszym n, aż osiągnie definicję bazową (0!). Wtedy zaczyna zwracać wyniki i rekurencja się kończy.

Rekurencja jest używana w programowaniu, aby rozwiązywać problemy, które mają naturalne struktury podobieństwa lub podziały na mniejsze części, które można rozwiązać niezależnie. Ważne jest zrozumienie definicji bazowej (przypadku bazowego) i kroku rekurencyjnego, aby uniknąć błędów i nieskończonych pętli rekurencyjnych.

Zadanie 1

Ciąg Fibonacciego to sekwencja liczb, w której każdy następny element jest sumą dwóch poprzednich. Zadanie polega na napisaniu funkcji, która rekurencyjnie oblicza n-tego elementu tego ciągu.

Wzór:

F(0) = 0
F(1) = 1
F(n) = F(n-1) + F(n-2) dla n >= 2

Zadanie 2

Napisz funkcję, która rekurencyjnie oblicza sumę wszystkich liczb całkowitych od 1 do danej liczby całkowitej n.

Np.
n=5 -> 1+2+3+4+5 = 15

Zadanie 3

Stwórz funkcję, która przyjmuje dwie liczby całkowite jako argumenty, gdzie pierwsza liczba to podstawa, a druga to wykładnik. Funkcja ma obliczyć wynik podniesienia podstawy do potęgi wykładnika, korzystając z rekurencji.

2, 3 -> 2*2*2=8

Zadanie 4

Napisz funkcję, która przyjmuje liczbę całkowitą jako argument i oblicza sumę jej cyfr. Na przykład, dla liczby 12345 suma cyfr wynosi 15. Użyj rekurencji do rozwiązania tego zadania.

Np.
345 -> 3+4+5 =12

Zadanie 5

Napisz funkcję, która przyjmuje tekst i słowo jako argumenty i zlicza ilość wystąpień tego słowa w tekście. Wykorzystaj rekurencję do przeszukiwania tekstu.

Np.
„ala ma kota i psa i kota pies ala”, „ala”
Rezultat:
Slowo „ala” wystepapilo 2 razy.

Zadanie 6

Napisz funkcje rekurencyjną, która pozwoli nam odwrócić zawartość tablicy. Pamiętaj, że nie chcemy wykorzystywać pętli.

Np.
[4, 5, 3, 1] -> [1, 3, 5, 4]