

PERTEMUAN 6

STRUKTUR REKURSIF

STRUKTUR REKURSIF

Rekursif adalah suatu proses yang bisa memanggil dirinya sendiri.

Contoh konsep penggunaan Rekursif

Masalah : Memotong Roti tawar tipis-tipis sampai habis

Algoritma :

1. Jika roti sudah habis atau potongannya sudah paling tipis maka pemotongan roti selesai.
2. Jika roti masih bisa dipotong, potong tipis dari tepi roti tersebut, lalu lakukan prosedur 1 dan 2 untuk sisa potongannya.

Contoh Fungsi Rekursif

- a. Fungsi pangkat
- b. Faktorial
- c. Fibonancy
- d. Menara Hanoi

Fungsi Pangkat

Menghitung 10 pangkat n dengan menggunakan konsep rekursif.

Secara Notasi pemrograman dapat ditulis :

$$10^0 = 1 \quad \text{.....(1)}$$

$$10^n = 10 * 10^{n-1} \quad \text{.....(2)}$$

Contoh :

$$10^3 = 10 * 10^2$$

$$10^2 = 10 * 10^1$$

$$10^1 = 10 * 10^0$$

$$10^0 = 1$$

Fungsi Pangkat

```
#Fungsi Pangkat secara Rekursif
def pangkat(x,y):
    if y == 0:
        return 1
    else:
        return x * pangkat(x,y-1)

x = int(input("Masukan Nilai X : "))
y = int(input("Masukan Nilai Y : "))

print("%d dipangkatkan %d = %d"
      % (x,y,pangkat(x,y)))
```

Output Program:

Masukan Nilai X : 10

Masukan Nilai Y : 3

10 dipangkatkan 3 = 1000

Fungsi Pangkat

- Fungsi pangkat akan memanggil dirinya sendiri, yaitu setiap nilai x dan y di input akan dikirim ke fungsi pangkat() melalui parameter variabel x dan y .
- Selama nilai y bukan 0 maka fungsi pangkat() akan terus memanggil dirinya sendiri, dan nilai y akan selalu berkurang 1 ($y-1$) sampai kondisi terpenuhi dan perulangan dihentikan.

Faktorial

$$0! = 1$$

$$N! = N \times (N-1)! \quad \text{Untuk } N > 0$$

Scr notasi pemrograman dapat ditulis sebagai :

$$\text{FAKT}(0) = 1 \quad \dots\dots\dots (1)$$

$$\text{FAKT}(N) = N * \text{FAKT}(N-1) \dots\dots\dots (2)$$

Contoh :

$$\text{FAKT}(5) = 5 * \text{FAKT}(4)$$

$$\text{FAKT}(4) = 4 * \text{FAKT}(3)$$

$$\text{FAKT}(3) = 3 * \text{FAKT}(2)$$

$$\text{FAKT}(2) = 2 * \text{FAKT}(1)$$

$$\text{FAKT}(1) = 1 * \text{FAKT}(0)$$

Nilai Awal

Misal :

hitung $5!$, maka dapat dilakukan secara rekursif dgn cara :

$$5! = 5 * 4!$$

Scr rekursif nilai dr $4!$ Dpt dihitung kembali dgn $4 * 3!$,

$$\text{shg } 5! \text{ Menjadi : } 5! = 5 * 4 * 3!$$

Scr rekursif nilai dr $3!$ Dpt dihitung kembali dgn $3 * 2!$, shg $5!$ Menjadi : $5! = 5 * 4 * 3 * 2!$

Scr rekursif nilai dr $2!$ Dpt dihitung kembali dgn $2 * 1$, shg $5!$ Menjadi : $5! = 5 * 4 * 3 * 2 * 1 = 120$.

Program Faktorial

```
#Fungsi Pangkat secara Rekursif
def faktorial(a):
    if a == 1:
        return (a)
    else:
        return (a*faktorial(a-1))

bil = int(input("Masukan Bilangan :
"))

print("%d! = %d" % (bil,
faktorial(bil)))
```

Output Program:
Masukan Bilangan : 5
5! = 120

Masukan Bilangan : 6
6! = 720

Fungsi Faktorial

- Fungsi Faktorial adalah fungsi rekursif karena memanggil fungsinya sendiri.
- Pada saat dijalankan program akan meminta “memasukkan bilangan” pada variabel bil, kemudian bilangan tersebut akan dikirim ke fungsi faktorial() lewat parameter a.
- Selama nilai a tidak sama dengan 1 maka fungsi faktorial akan terus memanggil dirinya sendiri. Perulangan akan berhenti ketika nilai =1

Fibonancy

Deret Fibonancy : 0,1,1,2,3,5,8,13,.....

Secara notasi pemrograman dapat ditulis sebagai :

Fibo (1) = 0 & Fibo (2) = 1 (1)

Fibo (N) = Fibo (N-1) + Fibo (N-2) (2)

Contoh :

Fibo(5) = Fibo(4) + Fibo(3)

Fibo(4) = Fibo(3) + Fibo(2)

Fibo(3) = Fibo(2) + Fibo(1)

└──────────┘

Nilai Awal

Program Deret Fibonancy

```
#Fibonacci Secara Rekursif
def fibonacci(n):
    if n == 0 or n == 1:
        return n
    else:
        return (fibonacci(n-1) +
fibonacci(n-2))

x = int(input("Masukan Batas
Deret Bilangan Fibonacci : "))
print("Deret Fibonacci")
for i in range(x):
    print(fibonacci(i),end=' ')
```

Output Program:

Masukan Batas Deret Bilangan
Fibonacci : 5

Deret Fibonacci
0 1 1 2 3

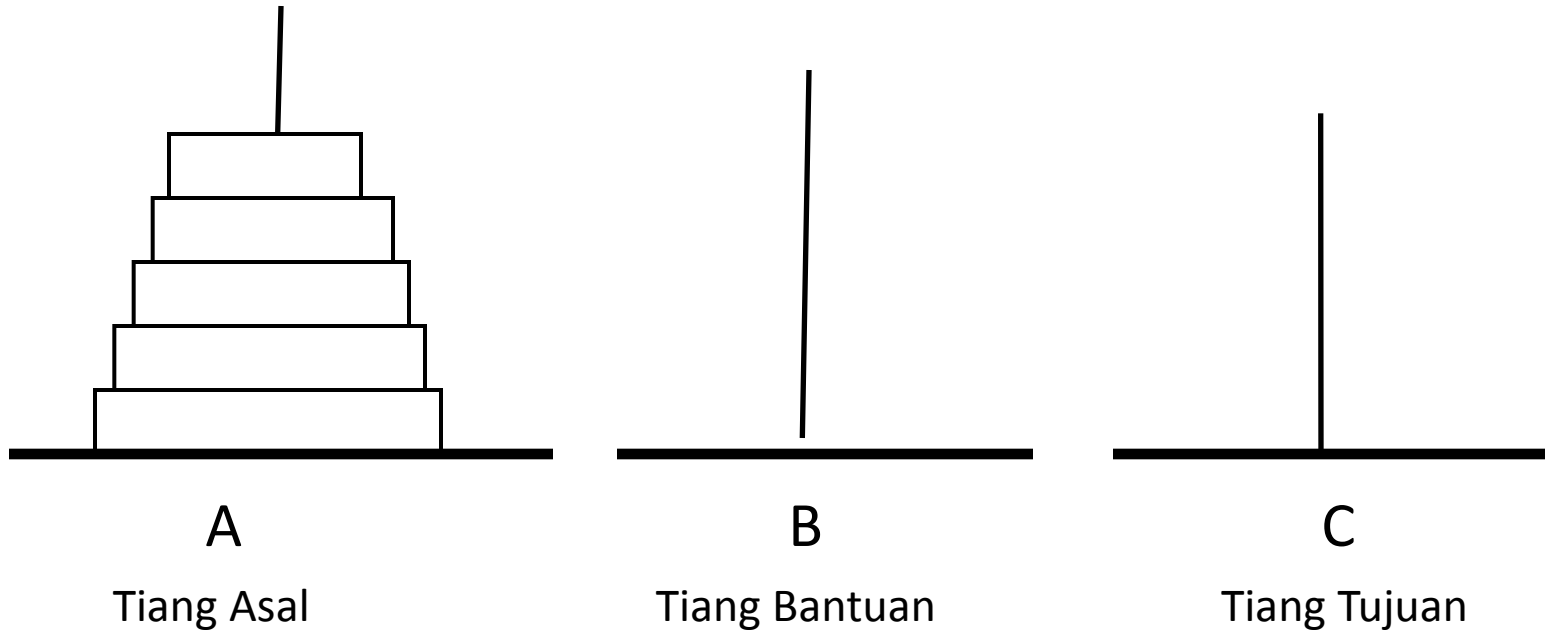
Masukan Batas Deret Bilangan
Fibonacci : 8

Deret Fibonacci
0 1 1 2 3 5 8 13

Fungsi Fibonancy

- Fungsi fibonancy merupakan fungsi rekursif yang memanggil dirinya sendiri.
- Bilangan fibonancy adalah bilangan yang memiliki suku awal 0 dan 1, dan suku berikutnya adalah penjumlahan dari dua suku sebelumnya.
- Fungsi fibonancy akan terus memanggil dirinya ketika (nilai n) bukan bernilai 0 atau 1 dengan melakukan proses penjumlahan ($\text{fibonacci}(n-1) + \text{fibonacci}(n-2)$)

Konsep Menara Hanoi



- ❖ Jika $n=1$, maka langsung pindahkan saja piringan dr tiang A ke tiang C & selesai.
- ❖ Pindahkan $n-1$ piringan yg paling atas dr tiang A ke tiang B.
- ❖ Pindahkan piringan ke n (piringan terakhir) dr tiang A ketiang C
- ❖ Pindahkan $n-1$ piringan dari tiang B ke tiang C.

Langkah pemindahan tsb diatas dpt diubah dengan notasi sbb:

Menara (n,asal,bantu,tujuan)

- Utk jml piringan $n > 1$ dpt dibagi menjadi 3 notasi penyelesaian
- Menara ($n-1$, Asal, Tujuan, Bantu);
- Menara (n , Asal, Bantu, Tujuan); atau Asal \rightarrow Tujuan;
- Menara ($n-1$, Bantu, Asal, Tujuan);

Langkah Pemindahan Piringan

MENARA(1,A,C,B) A
 → B
 MENARA(2,A,B,C) A → C A
 → C
 MENARA(1,B,A,C)B
 → C
 MENARA(3,A,C,B) A→B A
 → B
 MENARA(1,C,B,A)C → A
 MENARA(2,C,A,B)C → B C → B
 MENARA(1,A,C,B) A
 → B
 MENARAA → C
 A → C
 (4,A,B,C)
 MENARA(1,B,A,C) B → C
 MENARA(2,B,C,A) B → AB →
 A
 MENARA(1,C,B,A) C → A
 MENARA(3,B,A,C) B →C B → C
 MENARA(1,A,C,B) A →
 B
 MENARA(2,A,B,C) A → C A →
 C
 MENARA(1,B,A,C) B → C

Lanjutan

Ilustrasi diatas menghasilkan 15 langkah penyelesaian dari permasalahan konsep menara Hanoi dgn jumlah piringan sebanyak 4 buah

Untuk Video konsep menara hanoi dapat dilihat pada:
<https://www.mathsisfun.com/games/towerofhanoi.html>

Rumus Langkah Pemindahan :

$$2^N - 1$$

N = Jumlah Piringan