



Latihan Pertemuan Sebelumnya



```
1.  Uses Wincrt;
    Type mhs = record
        NIM : String[4];
        Nama : String[20];
        Prodi : String[20];
        IP : Real;
    End;

    Var data : mhs;
    Begin
        With data do
            Begin
                Write('NIM   : ');Readln(NIM);
                Write('Nama       : ');Readln(Nama);
                Write('Program Studi : ');Readln(Prodi);
                Write('IP       : ');Readln(IP);
            End;
            Writeln;
            Writeln;
            Writeln('NIM       : ',data.NIM);
            Writeln('Nama       : ',data.Nama);
            Writeln('Program Studi : ',data.Prodi);
            Writeln('IP         : ',data.IP:2:2);
        End.
```

2. Untuk program nomor 2, terlebih dahulu buatlah sebuah file dengan nama `test.txt`. Isilah file tersebut dengan kalimat: `This is a test text file`. Simpan file di folder yang sama dengan file program.

```
PROGRAM ReadMultipleStrings1 (OUTPUT, F);  
VAR  
F :TEXT;  
Str1, Str2, Str3 :STRING[5];  
BEGIN  
ASSIGN (F, 'test.txt');  
RESET (F);  
READLN (F, Str1, Str2, Str3);  
WRITELN ('Str1= ', Str1);  
WRITELN ('Str2= ', Str2);  
WRITELN ('Str3= ', Str3);  
CLOSE (F);  
WRITELN ('Press ENTER to continue..');  
READLN  
END.
```

3. Untuk program nomor 3, simpan ulang (save as) program nomor 2. Lalu ubah deklarasi `Str1, Str2, Str3` dengan menghapus panjang stringnya menjadi seperti berikut ini.

```
Str1, Str2, Str3 : STRING;
```

Amati dan pahami output yang dihasilkan.



POLITEKNIK STATISTIKA STIS

For Better Official Statistics

REKURSI

Farid Ridho, S.S.T., M.T., Nori Wilantika, S.S.T., M.T.I.



- Konsep Rekursi
- Penulisan Fungsi Rekusif
- Perbedaan antara rekursi dan iterasi serta kapan menggunakannya
- Teknik pemecahan masalah menggunakan rekursi
- Latihan



- Sub program dapat memanggil sub program lainnya
- Apabila sub program memanggil dirinya sendiri, inilah yang disebut dengan fungsi rekursif

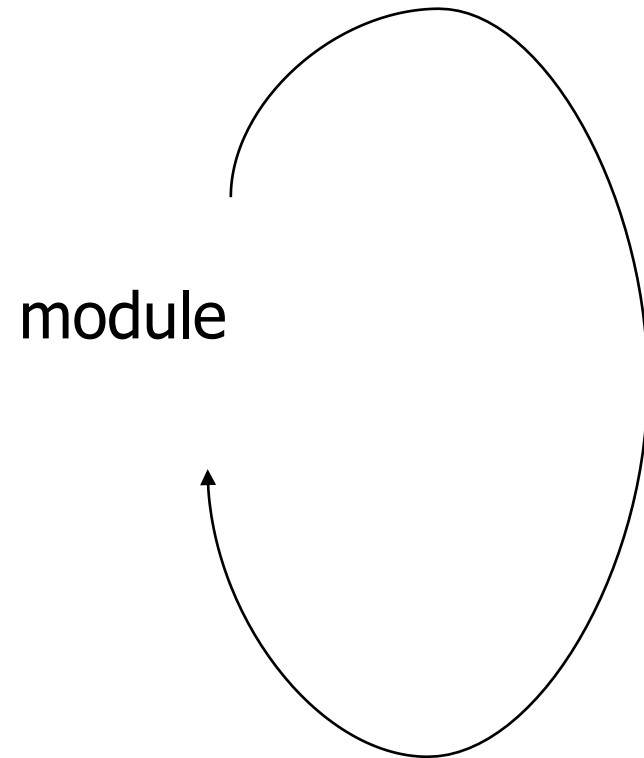
- Prosedur rekursif:

```
Procedure proc();  
Begin  
  proc();  
End;
```

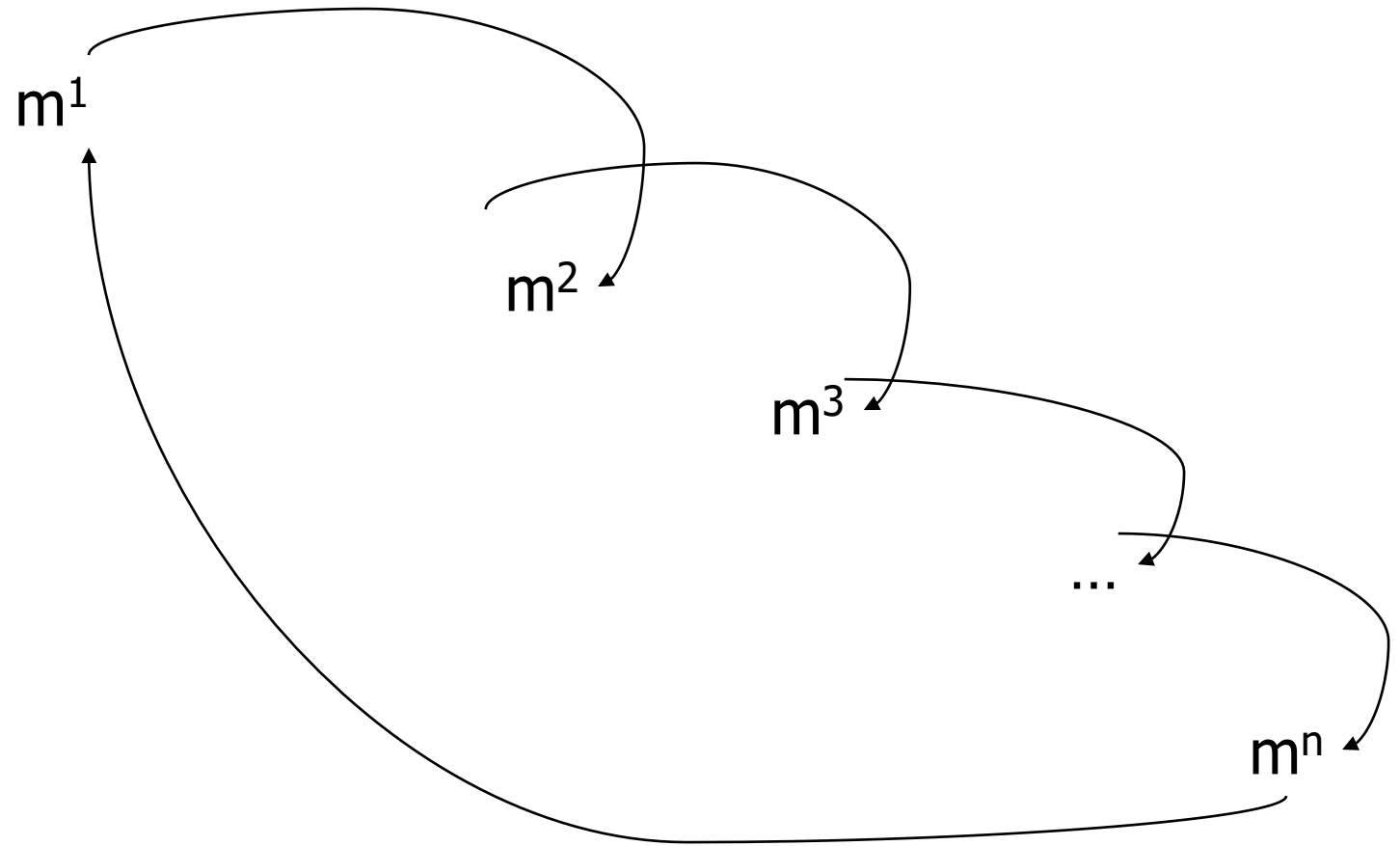
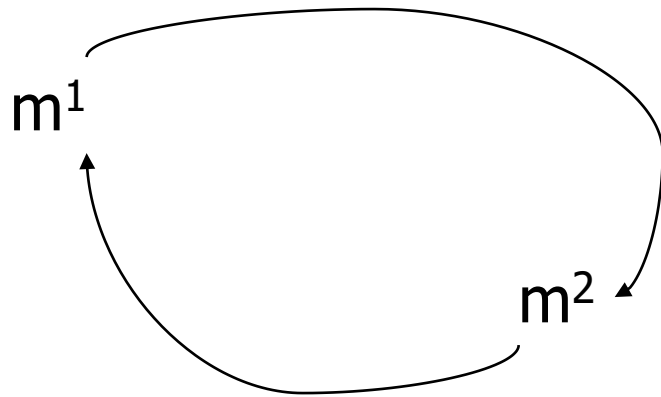
- Fungsi rekursif:

```
Function func();  
Begin  
  func();  
End;
```


- Pemanggilan secara langsung (direct call)
- Pemanggilan tak langsung (indirect call)



```
procedure  
proc;  
begin  
    :  
    proc ();  
    :  
end;
```

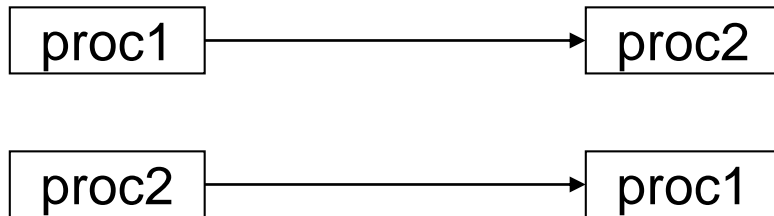


```
procedure proc1;  
begin  
  :  
  proc2;  
end;
```

```
procedure proc2;  
begin  
  :  
  proc3;  
end;
```

```
procedure proc3;  
begin  
  :  
  proc1;  
end;
```

Contoh scenario:



Procedure mana yang harus didefinisikan lebih dulu?

```
procedure proc1;  
begin  
  :  
  proc2;  
  :  
end;
```

```
procedure proc2;  
begin  
  :  
  proc1;  
  :  
end;
```

What is proc2?



```
procedure proc2;
```

```
begin
```

```
:
```

```
proc1;
```

```
:
```

```
end;
```

```
procedure proc1;
```

```
begin
```

```
:
```

```
proc2;
```

```
:
```

```
end;
```

What is proc1?



A "placeholder" for the compiler (definition comes later)

```
procedure proc2; FORWARD;
```

```
procedure proc1;
```

```
begin
```

```
  :
```

```
  proc2;
```

```
  :
```

```
end;
```

```
procedure proc2;
```

```
begin
```

```
  :
```

```
  proc1;
```

```
  :
```

```
end;
```


Konsep Faktorial

$$\begin{aligned}n! &= n * (n-1) ! \\&= n * (n-1) * (n-2) ! \\&\quad \dots \\&= n * (n-1) * (n-2) * (n-3) \dots 1\end{aligned}$$

Contoh:

Rekursif

```
faktorial(5)
-> 5 * faktorial(4)
-> 5 * (4 * faktorial(3))
-> 5 * (4 * (3 * faktorial(2)))
-> 5 * (4 * (3 * (2 * faktorial(1))))
-> 5 * (4 * (3 * (2 * 1)))
-> 5 * (4 * (3 * 2))
-> 5 * (4 * 6)
-> 5 * 24
-> 120
```

```
program factorialProgram;
```

```
function factorial (num :integer):integer;
var i,hasil: integer;
begin
    hasil := num;
    for i := num-1 downto 1 do
        hasil := hasil*i;
    factorial := hasil;
end;
```

```
var number, nFactorial :integer;
```

```
begin
write('Enter the number for factorial :');
readln(number);
nFactorial := factorial(number);
writeln(number, '! = ', nFactorial);
end.
```

```
program factorialProgram;
```

```
function factorial (num :integer):integer;
begin
if (num = 1) then
    factorial := 1
else
    factorial := num*factorial(num - 1);
end;
```

```
var number, nFactorial :integer;
```

```
begin
write('Enter the number for factorial :');
readln(number);
nFactorial := factorial(number);
writeln(number, '! = ', nFactorial);
end.
```

User Types in 3
nFactorial = factorial(3)

factorial (3)
if (3 = 1) then **false**
else
→ factorial := 3 * factorial(3 - 1);

factorial (2)
if (2 = 1) then **false**
else
→ factorial := 2 * factorial(2 - 1);

factorial (1)
if (1 = 1) then **true**
else
factorial := 1 * factorial(1 - 1);

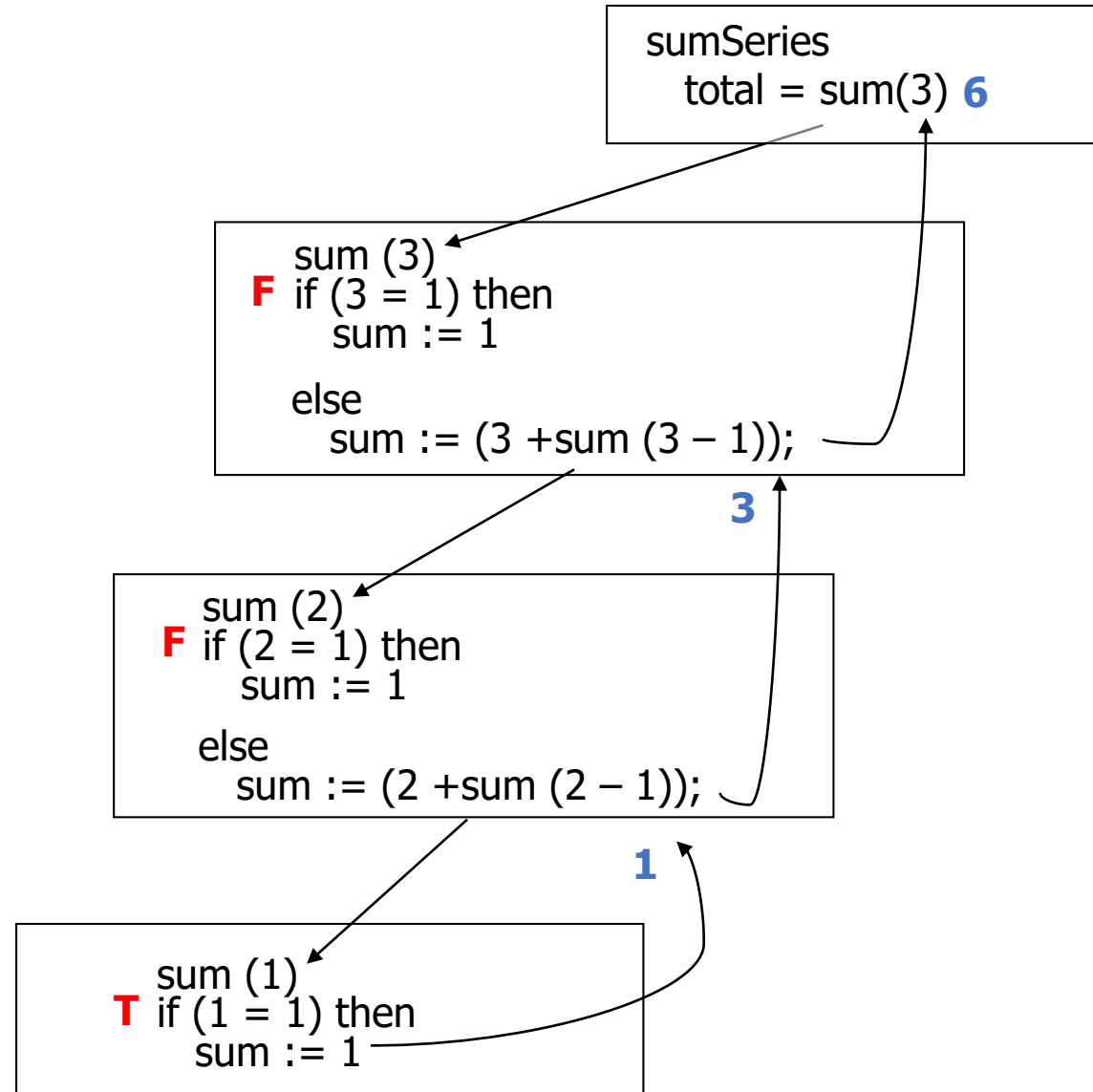
```
program sumSeries (input, output);

function sum (no : integer): integer;
begin
    if (no = 1) then
        sum := 1
    else
        sum := (no + sum (no - 1));
    end;

var lastNumber, total : integer;

begin
    write('Masukkan angka terakhir :');
    readln(lastNumber);
    total := sum(lastNumber);
    writeln('Jumlah dari 1 hingga angka tersebut ', total);
end.
```

```
function sum (no : integer): integer;
var i: integer;
begin
    if (no = 1) then
        sum := 1
    else
        sum := 0;
        for i := 1 to no do
            sum := sum + i;
        end;
    end;
```



- Dalam rekursi ada yang disebut sebagai *base case*,
- Dimana apabila kondisi ini tercapai maka suatu fungsi atau prosedur rekursi akan berakhir.
- Jika kondisi pada *base case* tidak tercapai (bernilai false) maka akan terjadi proses rekursi.

```
Function func () ;  
Begin  
if base_case then  
    return_value  
else  
    func () ;  
End;
```

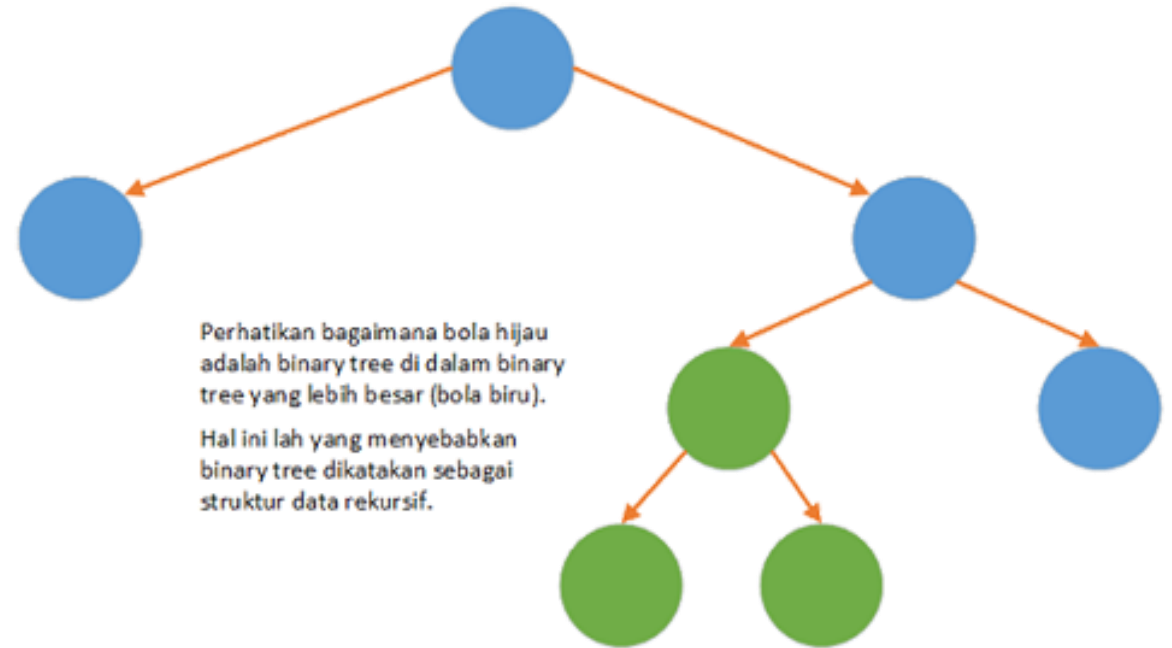
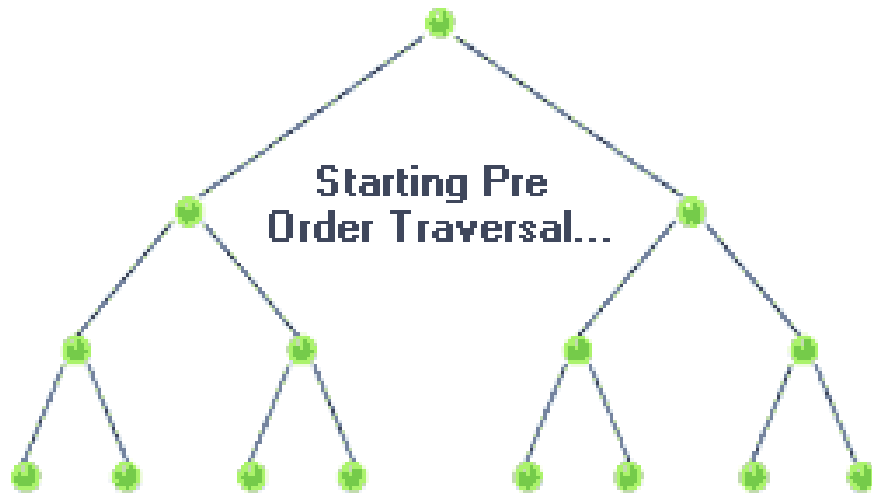
RECURSION	ITERATIONS
Fungsi rekursif adalah fungsi yang memanggil dirinya sendiri	Iterasi adalah pengulangan dari suatu proses menggunakan struktur loop
Rekursi menggunakan struktur pemilihan	Iterasi menggunakan struktur perulangan
Rekursi tak hingga terjadi apabila langkah rekursi tidak pernah memenuhi kondisi pada base case	Perulangan tak hingga terjadi apabila kondisi perulangan tidak pernah bernilai false
Rekursi berhenti apabila kondisi (base case) dikenali	Iterasi berhenti ketika kondisi perulangan bernilai false
Rekursi lebih banyak menggunakan memory daripada iterasi	Iterasi lebih hemat memory
Rekursi tak hingga dapat menyebabkan terjadinya crash pada sistem	Perulangan tak hingga akan mengakibatkan penggunaan Siklus CPU secara berulang
Rekursi membuat kode program menjadi lebih sederhana	Iterasi membuat kode program menjadi lebih panjang

- Kelebihan
 - solusi sangatlah efisien
 - dapat memecahkan masalah yang sulit dengan tahapan yang mudah dan singkat
- Kelemahan
 - sulit dipahami
 - perlu stack besar (stack overrun)

- Strategi yang biasa digunakan disebut dengan “Divide and Conquer”,
- yaitu dengan membagi masalah menjadi masalah dalam versi kecil kemudian menyelesaikanya.
 1. Bagi P (Problem) menjadi beberapa subproblems, P_1, P_2, \dots, P_n .
 2. Selesaikan setiap subproblem sampai mendapatkan solusi $S_1 \dots S_n$.
 3. Gunakan $S_1 \dots S_n$ untuk menyelesaikan masalah asal P .

- Amir akan memindahkan satu kardus buku yang berisi 50 buku dari dalam kantor menuju mobil.
- P (Problem): memindahkan satu kardus buku yang berisi 50 buku
- Karena amir tidak mampu mengangkat keseluruhan buku maka dia membagi menjadi beberapa tumpukan P1 berisi 10 buku, P2 berisi 15 buku, P3 berisi 25 buku.
- Strategi amir dengan memindahkan setiap tumpukan buku, kemudian menyusun kembali menjadi 1 kardus buku besar di mobil ini bisa disebut sebagai rekursi.

- Strategi yang umum digunakan adalah “Divide and Conquer”
- Beberapa pertanyaan yang harus dijawab:
 - Bagaimana kita dapat memecah suatu masalah menjadi beberapa masalah sama dalam versi yang lebih kecil.
 - Bagaimana setiap pemanggilan fungsi dapat membuat masalah menjadi versi kecil.
 - Apa *base case* dari masalah ini?
 - Apakah algoritma akan selalu mencapai kondisi *base case*?
- Membuat fungsi dan prosedur rekursif
 - Memahami masalah yang akan dipecahkan secara tepat. Langkah ini merupakan langkah awal dari seluruh permasalahan pemrograman.
 - Tentukan seberapa besar masalah yang akan dipecahkan menjadi beberapa subprogram.
 - Kenali dan tentukan base case dari masalah yang akan dikerjakan secara tidak rekursif.
 - Terakhir kenali dan tentukan kondisi umum (general case) dengan benar



Deret Fibonacci: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ...

```
program fibo_using_rekursif;
var x,i: integer;

function fib(n:integer):integer;
begin
    if(n=1) then fib:=1
    else if (n=2) then fib:=1
    else fib:= fib(n-1)+fib(n-2);
end;

begin
    writeln('Deret fibonacci');
    write('Jumlah Suku: ');
    readln(x);
    for i := 1 to x do
        write(fib(i), ' ');
    end.
```

$$\begin{aligned} \text{fib}(6) &= \text{fib}(4) + \text{fib}(5) \\ &= \text{fib}(2) + \text{fib}(3) + \text{fib}(5) \\ &= 1 + \text{fib}(3) + \text{fib}(5) \\ &= 1 + \text{fib}(1) + \text{fib}(2) + \text{fib}(5) \\ &= 1 + 1 + 1 + \text{fib}(3) + \text{fib}(4) \\ &= 3 + \text{fib}(1) + \text{fib}(2) + \text{fib}(4) \\ &= 3 + 1 + 1 + \text{fib}(2) + \text{fib}(3) \\ &= 5 + \text{fib}(2) + \text{fib}(3) \\ &= 5 + 1 + \text{fib}(1) + \text{fib}(2) \\ &= 6 + 1 + 1 \\ &= 8 \end{aligned}$$

```
program fibo_using_iteration;

function fib(n:integer):integer;
var i,j,k,l: integer;
begin
    k:=1;
    l:=0;

    for i:=1 to n do
        begin
            write(' ',k);
            j:=k+l;
            l:=k;
            k:=j;
        end;
    end;

var x: integer;
begin
    writeln('Deret fibonacci');
    write('Jumlah Suku: ');
    readln(x);
    fib(x);
end.
```



LATIHAN



1. Buat program untuk memangkatkan suatu bilangan berdasarkan input dari user. Program menggunakan fungsi atau prosedur yang bersifat rekursif.

Output program:

Masukkan bilangan: 5

Masukkan pangkat bilangan: 3

$5^3 = 125$

2. Buat segitiga pascal dengan menggunakan fungsi atau prosedur rekursif.
Dimana, tinggi segitiga pascal ditentukan oleh user.

```
      1
     1 1
    1 2 1
   1 3 3 1
  1 4 6 4 1
 1 5 10 10 5 1
1 6 15 20 15 6 1
```

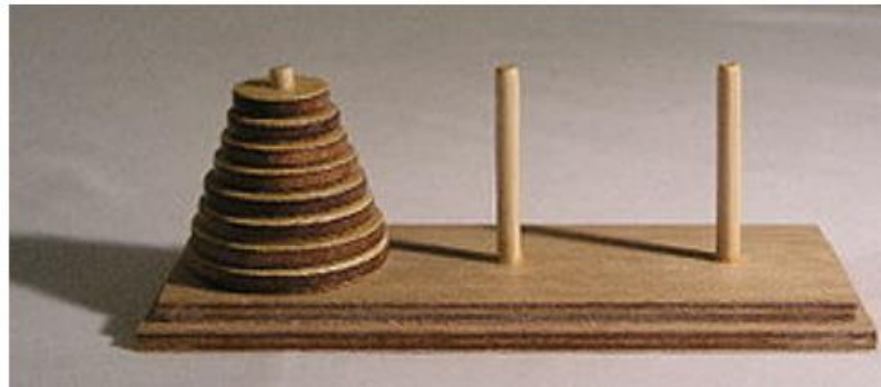
3. Buatlah program pascal untuk menyelesaikan kasus menara Hanoi.

Output program

```
Enter number of disks: 3
Move the plate from A to B
Move the plate from A to C
Move the plate from B to C
Move the plate from A to B
Move the plate from C to A
Move the plate from C to B
Move the plate from A to B
```

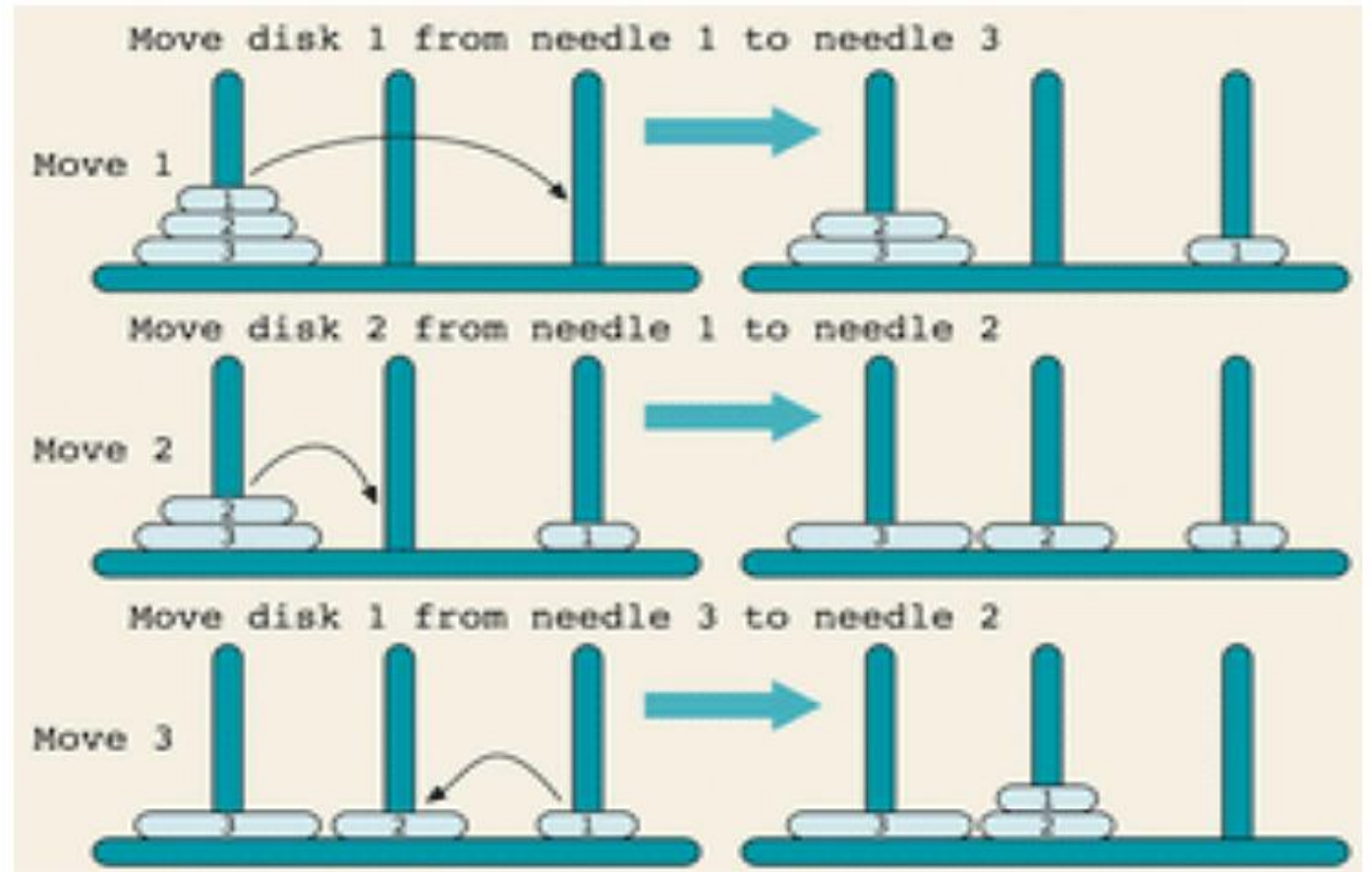
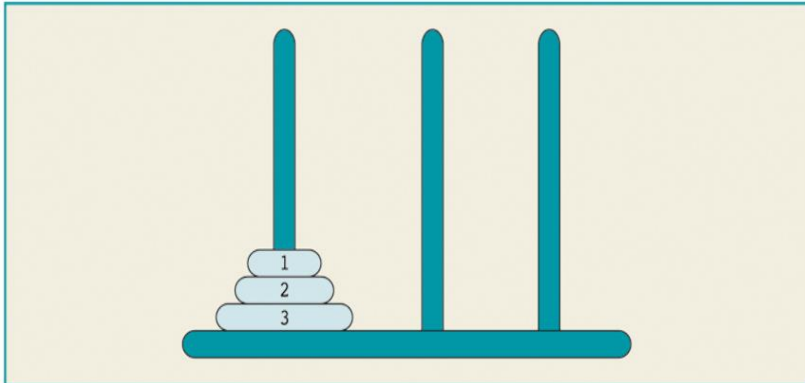
Menara Hanoi

- Legenda Kuil Brahma
- Memiliki 64 cakram emas dan 3 tiang
- Dunia akan kiamat apabila biarawan memindahkan keseluruhan cakram dari tiang 1 ke tiang terakhir (katanya)



Aturan

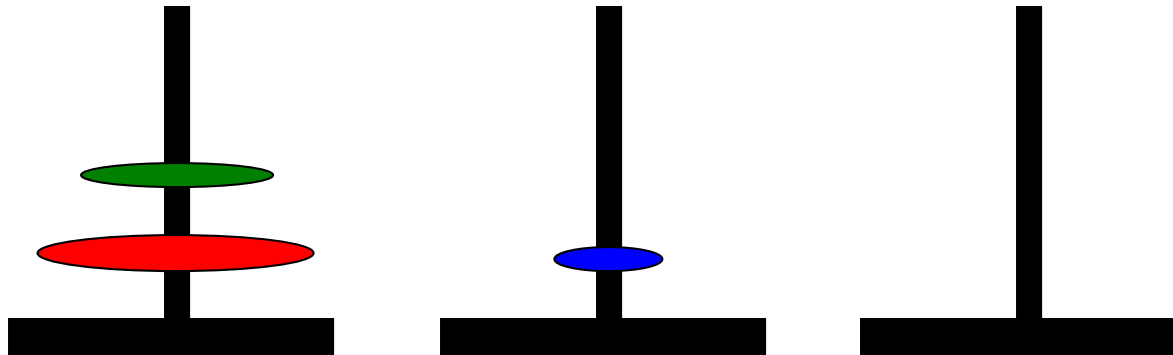
- Cakram hanya boleh dipindahkan satu per satu
- Cakram yang lebih kecil harus selalu berada di atas cakram yang lebih besar
- Dengan 64 cakram, apabila lama pemindahan 1 cakram membutuhkan waktu 1 detik, maka waktu yang diperlukan adalah 585Milyar tahun



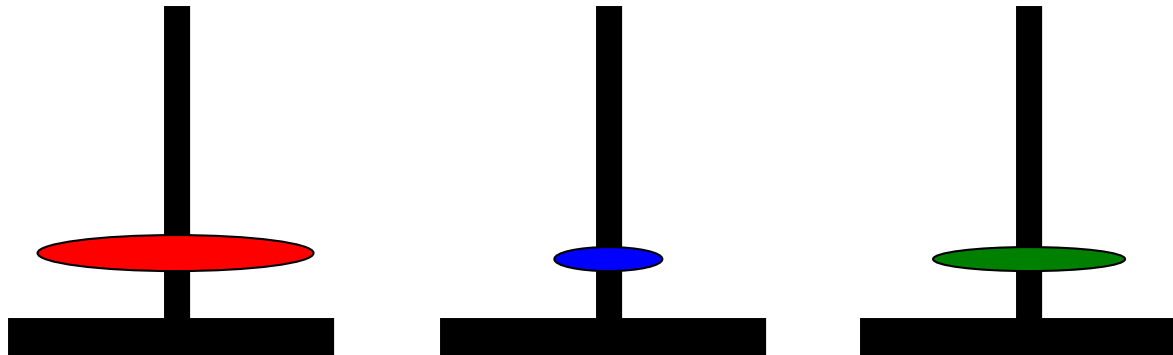
Tower of Hanoi



Tower of Hanoi



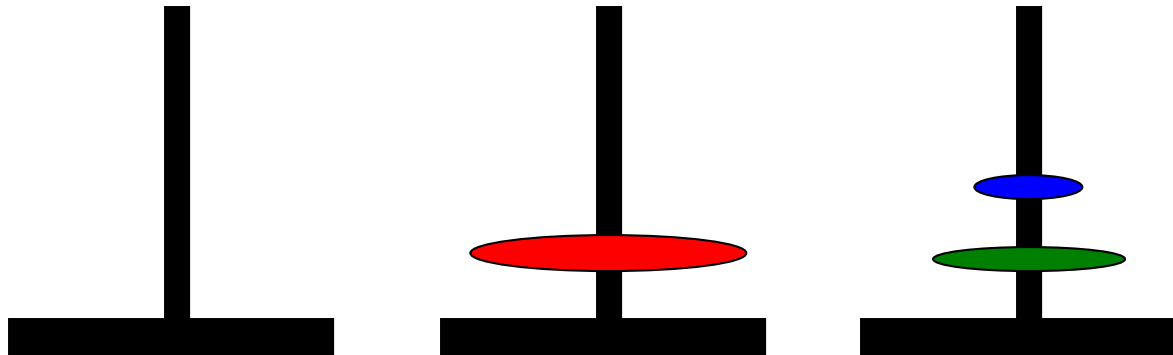
Tower of Hanoi



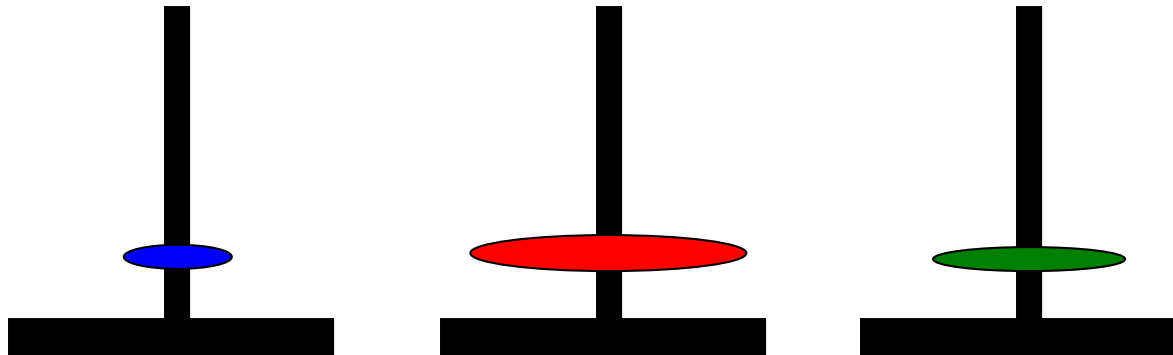
Tower of Hanoi



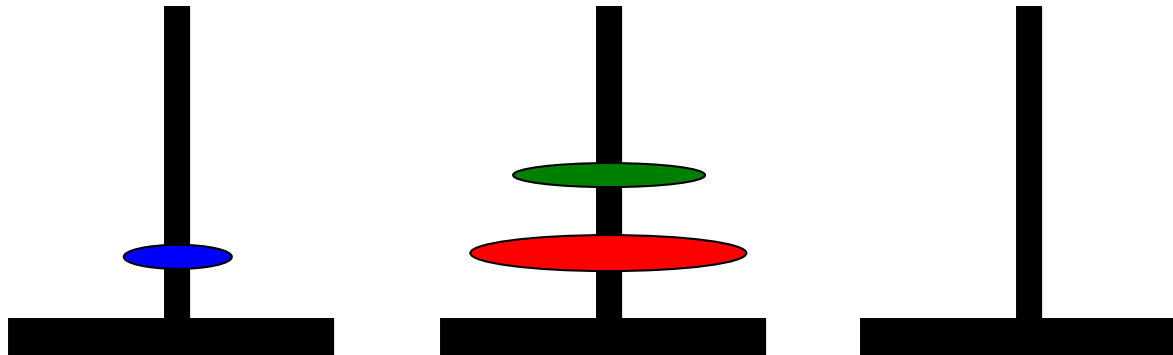
Tower of Hanoi



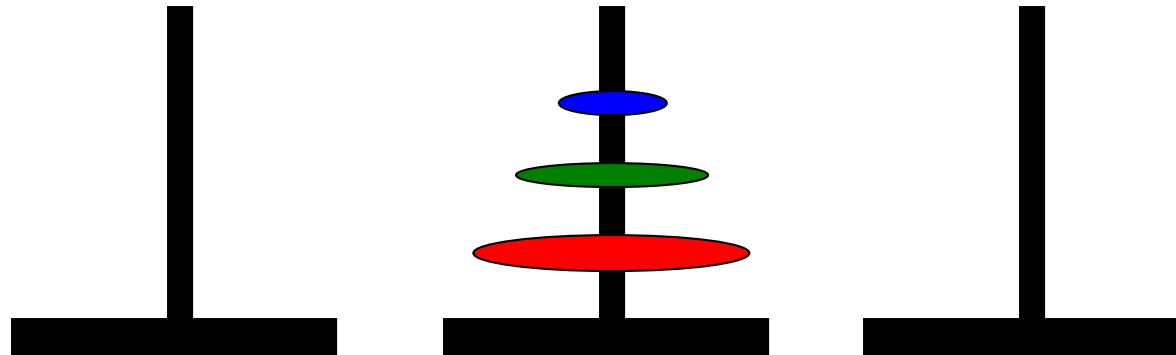
Tower of Hanoi



Tower of Hanoi



Tower of Hanoi





POLITEKNIK STATISTIKA STIS

For Better Official Statistics

TERIMA KASIH

