

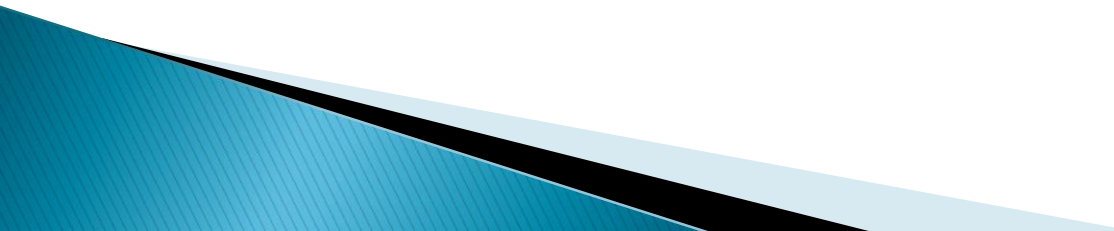
# **PERTEMUAN 6**

## Algoritma Rekursif

Hermanto, S.Kom. M.Kom.

# Apa itu fungsi rekursif?

---

- ❑ Alat/cara untuk memecahkan masalah dalam suatu fungsi atau procedure yang memanggil dirinya sendiri
  - ❑ Teknik pemecahan masalah yang powerful dan dapat digunakan ketika inti dari masalah terjadi berulang kali (for, while dan do-while)
  - ❑ Subrutin yang memanggil dirinya sendiri, baik langsung maupun tak langsung.
  - ❑ Subrutin rekursi bisa menyelesaikan tugas kompleks dalam beberapa baris perintah
  - ❑ Pada rekursif, method dapat memanggil dirinya sendiri.
- 

# Apa itu fungsi rekursif?

---

Permasalahan yang dapat diselesaikan oleh fungsi rekursif memiliki sifat

- Memiliki kasus sederhana yang dapat langsung diselesaikan **(base case)**. Contoh  $0! = 1$ .
- Kasus yang kompleks dapat diuraikan menjadi kasus yang identik dengan ukuran yang lebih kecil **(recursive cases)**. Contoh:  $n! = n * (n-1)!$
- Dengan menerapkan karakteristik 2 berulang-ulang, **recursive cases** akan mendekati dan sampai pada **base case**. Contoh:  $n! \rightarrow (n-1)! \rightarrow (n-2)! \rightarrow \dots 1!, 0!$ .

# Format fungsi rekursif

---

if this **base case**

    solve it

else

    redefine the problem using **recursion case**

# Format Fungsi Rekursif

---

- ❖ Cabang **if** berisi **base case**, sedangkan bagian **elsenya** berisi **recursive case**
- ❖ **Agar rekursi dapat berhenti** input recursive cases harus mendekati base case di setiap pemanggilan fungsi rekursif

# Format Fungsi Rekursif

---

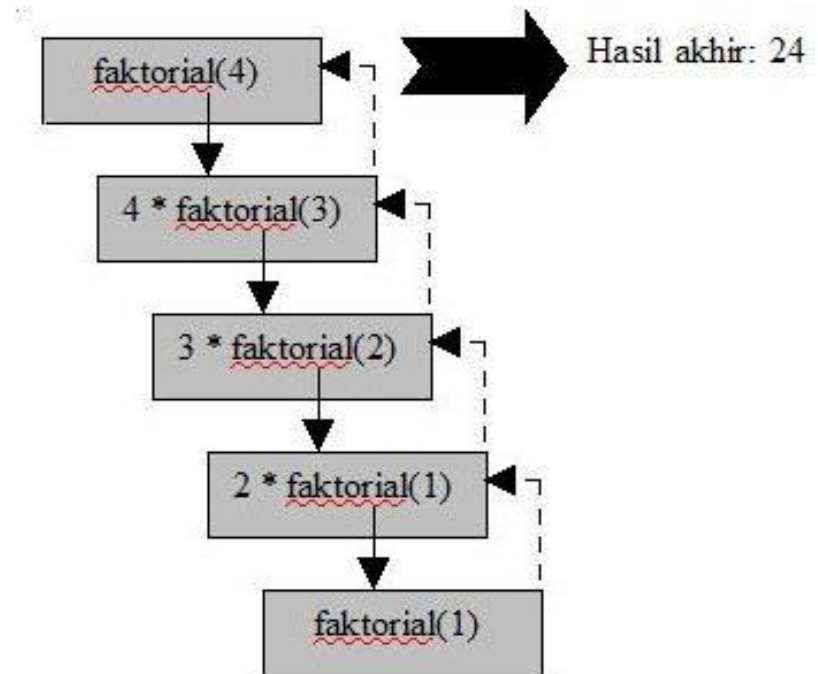
Contoh penerapan teknik rekursif

1. Perhitungan Nilai Faktorial
2. Pembentukan Barisan Fibonacci

# Faktorial

- Konsep Faktorial  
 $n! = n(n-1)(n-2)...1$
- Dapat diselesaikan dengan
  - Cara Biasa
  - Rekursif

$$F(n) = \begin{cases} 1 & \text{jika } n=0, n=1 \\ n * F(n) & \text{jika } n > 1 \end{cases}$$



# Faktorial : Cara Biasa

---

- Int Faktorial(int n)
- {
- if (n<0) return -1 ;
- else if (n>1)
- {
- S = 1 ;
- for(i=2 ;i<=n;i++) S = S \* n ;
- return S ;
- }
- else return 1 ;
- }



# Faktorial dengan Rekursif

---

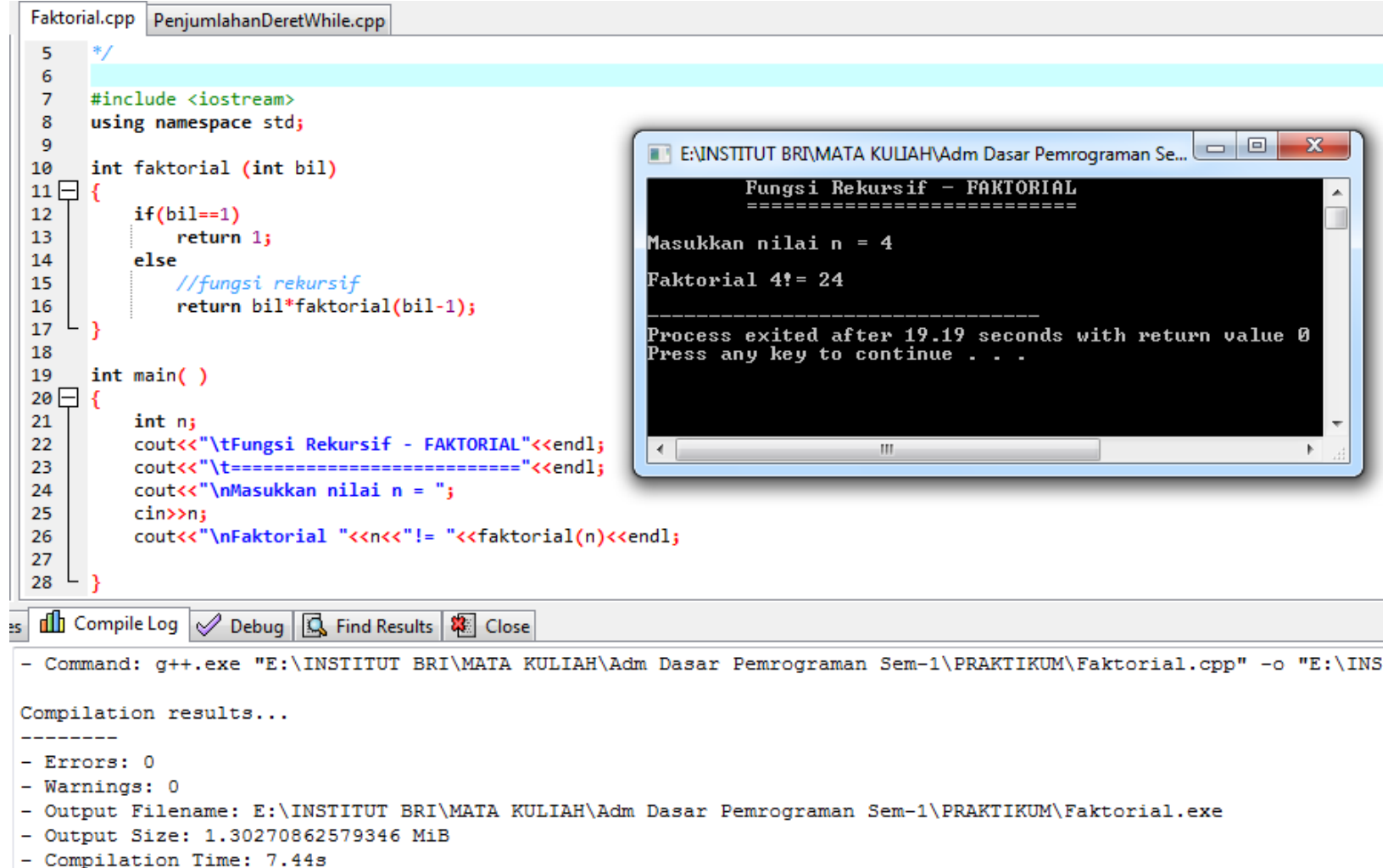
Faktorial (5)  
/  
Faktorial (4)\*5  
/  
Faktorial (3)\*4  
/  
Faktorial (2)\*3  
/  
Faktorial (1)\*2  
|  
1

## ▶ Algoritma:

- ▶  $n! = n \cdot (n-1)!$  , jika  $n > 1$   
 $n! = 1$  , jika  $n = 0, 1$

```
▶ int Faktorial(int n)
▶ {
    ▶ if ((n == 0) || (n == 1)) return (1);
    ▶ else
        ▶ return (n * Faktorial(n-1));
▶ }
```

# Faktorial dengan Rekursif



The image shows a C++ IDE with two tabs: 'Faktorial.cpp' and 'PenjumlahanDeretWhile.cpp'. The 'Faktorial.cpp' tab is active, displaying the following code:

```
5  /*
6
7  #include <iostream>
8  using namespace std;
9
10 int faktorial (int bil)
11 {
12     if(bil==1)
13         return 1;
14     else
15         //fungsi rekursif
16         return bil*faktorial(bil-1);
17 }
18
19 int main( )
20 {
21     int n;
22     cout<<"\tFungsi Rekursif - FAKTORIAL"<<endl;
23     cout<<"\t===== "<<endl;
24     cout<<"\nMasukkan nilai n = ";
25     cin>>n;
26     cout<<"\nFaktorial " <<n<<"!= " <<faktorial(n)<<endl;
27
28 }
```

Overlaid on the code editor is a console window titled 'E:\INSTITUT BRI\MATA KULIAH\Adm Dasar Pemrograman Se...'. It displays the program's output:

```
Fungsi Rekursif - FAKTORIAL
=====
Masukkan nilai n = 4
Faktorial 4!= 24

Process exited after 19.19 seconds with return value 0
Press any key to continue . . .
```

Below the code editor, the 'Compile Log' tab is active, showing the following information:

```
- Command: g++.exe "E:\INSTITUT BRI\MATA KULIAH\Adm Dasar Pemrograman Sem-1\PRAKTIKUM\Faktorial.cpp" -o "E:\INS
Compilation results...
-----
- Errors: 0
- Warnings: 0
- Output Filename: E:\INSTITUT BRI\MATA KULIAH\Adm Dasar Pemrograman Sem-1\PRAKTIKUM\Faktorial.exe
- Output Size: 1.30270862579346 MiB
- Compilation Time: 7.44s
```

# Bilangan Fibonacci

---

- Baris dari  $n=1$

1   1   2   3   5   8   13   21   34   55   89   144   233  
377   610   987   1597,...

- Algoritma (untuk  $n > 2$ ):

$$f_n = f_{n-1} + f_{n-2} \quad c/ \ n= 4$$

$$f_1 = 1$$

$$f_2 = 1$$

$$f_4 = f_3 + f_2$$

$$f_4 = (f_2 + f_1) + f_2$$

$$f_4 = (1+1) + 1$$

$$f_4 = 3$$

# Bilangan Fibonacci

---

## The Fibonacci Sequence

**1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377...**

$$1+1=2$$

$$1+2=3$$

$$2+3=5$$

$$3+5=8$$

$$5+8=13$$

$$8+13=21$$

$$13+21=34$$

$$21+34=55$$

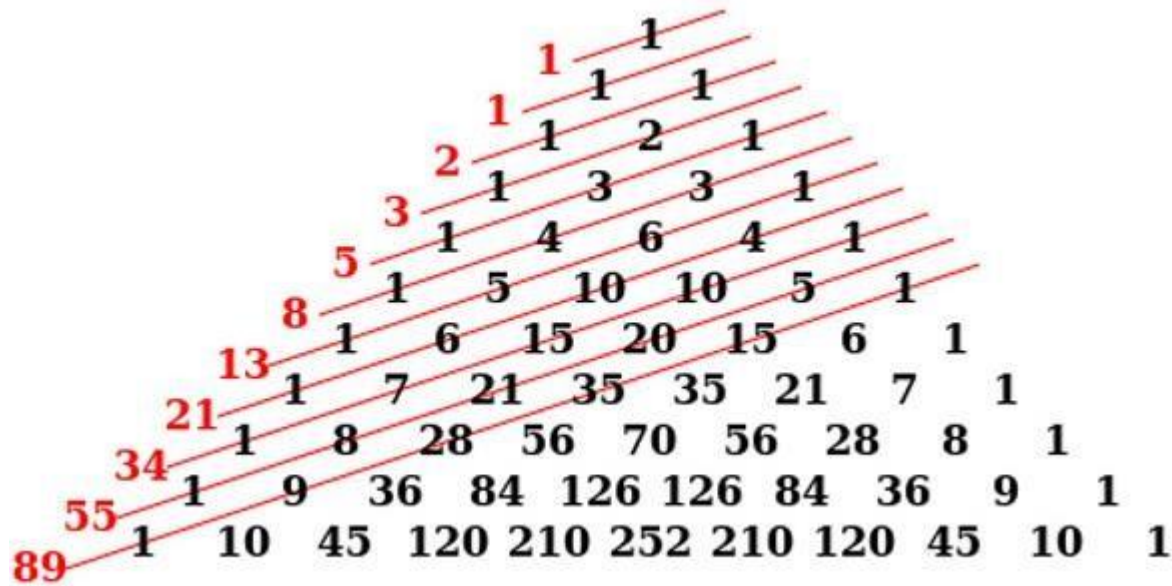
$$34+55=89$$

$$55+89=144$$

$$89+144=233$$

$$144+233=377$$

# Bilangan Fibonacci



```
int Fibonacci(int n){
    If (n ==1 || n==2) Then
        return (1)
    Else
        return (Fibonacci(n-1)+Fibonacci(n-2))
    Endif
}
```

# Deret Fibonacci

---

- procedure *fab*(n)
- 
- if  $n=1$  then  
return 1
- 
- if  $n=2$  then  
return 2
- return (*fab*(n-1) + *fab*(n-2))
- end

# Algoritma Rekursif

---

- ❑ Ciri masalah yang dapat diselesaikan secara rekursif adalah masalah itu dapat **di-reduksi** menjadi satu atau lebih **masalah-masalah serupa** yang **lebih kecil**.
- ❑ Secara umum, algoritme rekursif selalu mengandung dua macam kasus:
  - ❑ **Kasus induksi**: satu atau lebih kasus yang pemecahan masalahnya dilakukan dengan menyelesaikan masalah serupa yang lebih sederhana (yaitu menggunakan *recursive calls*)
  - ❑ **Kasus dasar** atau **kasus penyetop (base case)**: satu atau lebih kasus yang sudah sederhana sehingga pemecahan masalahnya tidak perlu lagi menggunakan recursive-calls.
- ❑ Supaya tidak terjadi rekursi yang tak berhingga, setiap langkah rekursif haruslah mengarah ke kasus penyetop (*base case*).

# Aturan Rekursif

---

1. Punya kasus dasar
  - ❖ Kasus yang sangat sederhana yang dapat memproses input tanpa perlu melakukan rekursif (memanggil method) lagi
2. Rekursif mengarah ke kasus dasar
3. “You gotta believe”. Asumsikan rekursif bekerja benar. Pada proses pemanggilan rekursif, asumsikan bahwa pemanggilan rekursif (untuk problem yang lebih kecil) adalah benar
  - ❖ Contoh: `pangkatRekursif(x, n)`
    - Asumsikan: `pangkatRekursif(x, n - 1)` menghasilkan nilai yang benar.
    - Nilai tersebut harus diapakan sehingga menghasilkan nilai `pangkatRekursif(x, n)` yang benar?
    - Jawabannya: dikalikan dengan `x`.
4. **Aturan penggabungan: Hindari duplikasi pemanggilan rekursif untuk sub-problem yang sama.**





Terima  
kasih