# Algoritma dan Struktur Data

## Rekursi

Umi Sa'adah
Tita Karlita
Entin Martiana Kusumaningtyas
Arna Fariza

2021

Politeknik Elektronika Negeri Surabaya
Departemen Teknik Informatika dan Komputer

# Overview

- Pendahuluan
- Iterasi vs Rekursi
- Konsep Rekursi
- Syarat Proses Rekursi
- Rekursi Tail

# Pendahuluan

- Rekursi adalah konsep pengulangan yang penting dalam ilmu komputer
  - Digunakan untuk merumuskan solusi sederhana dari sebuah permasalahan yang sulit untuk diselesaikan secara iteratif menggunakan loop for, while, atau do while
  - Digunakan untuk mendefinisikan permasalahan dengan konsisten dan sederhana
- Rekursi membantu mengekspresikan algoritma menjadi lebih mudah untuk dianalisa

# Apa sih rekursi itu ???

# Pengertian Rekursi

suatu proses yang memanggil dirinya sendiri untuk menyelesaikan suatu permasalahan yang diberikan

# Iterasi Vs Rekursi

## Iterasi

- Kode program lebih panjang
- Membutuhkan alokasi memori yang  relatif kecil
- Hanya membutuhkan satu kali  pemanggilan fungsi
- Mudah untuk di trace
- Membutuhkan algoritma yang panjang

# Iterasi Vs Rekursi

## Rekursi

- Kode program lebih ringkas dan mudah dipahami
- Membutuhkan alokasi memori yang lebih besar
- Membutuhkan pemanggilan fungsi yang berulang kali, dan terkadang menyebabkan overhead
- Susah untuk di trace
- Membutuhkan algoritma yang lebih sederhana

# Iterasi Vs Rekursi

- Dalam permasalahan faktorial sebuah bilangan n (ditulis n!), adalah hasil kali bilangan tsb dengan bilangan-bilangan di bawahnya hingga bilangan 1.
- Misal : 4! = (4)*(3)*(2)*(1)
- Dengan cara iteratif, secara umum dapat didefinisikan sbb :

$$n! = (n)*(n-1)*(n-2)* \ldots * (1)$$

# Iterasi Vs Rekursi

- Dengan cara rekursi, dapat dijabarkan sbb :
  - *n!* adalah hasil kali dari n dengan *(n-1)!*
  - *(n-1)!* adalah *n-1* dikalikan dengan *(n-2)!*
  - *(n-2)!* adalah *n-2* dikalikan dengan *(n-3)!*
  - dst sampai dengan ketika *n = 1*, proses berhenti
- Cara rekursif untuk permasalahan ini, secara umum dapat kita detailkan sebagai berikut:

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

# Perlu Diingat

- **Suatu fungsi rekursi memiliki :**

1. **Fase awal : Awal dimulainya perhitungan, terjadi pemanggilan terhadap diri sendiri dan berhenti sampai kondisi terminal**
2. **Kondisi terminal : Suatu kondisi yang menghentikan proses pemanggilan diri sendiri**
3. **Fase balik : Perhitungan kembali hasil kembalian dari fase awal**

**Tidak semua perulangan dapat diselesaikan dengan rekursi**

# Fase Pada Rekursi

1 **FASE AWAL**

- Proses memanggil dirinya sendiri
- Fase berhenti saat telah mencapai kondisi terminal

**FASE BALIK** 2

- Fungsi mengembalikan nilai yang dihasilkan dari fungsi ke fungsi sebelumnya
- Fase berhenti saat fungsi sampai pada fungsi awal yang memanggil

# Fase Pada Rekursi

$F(3) = 3 \times F(2)$ — *Fase awal*

$F(2) = 2 \times F(1)$

$F(1) = 1$ — *Kondisi terminal*

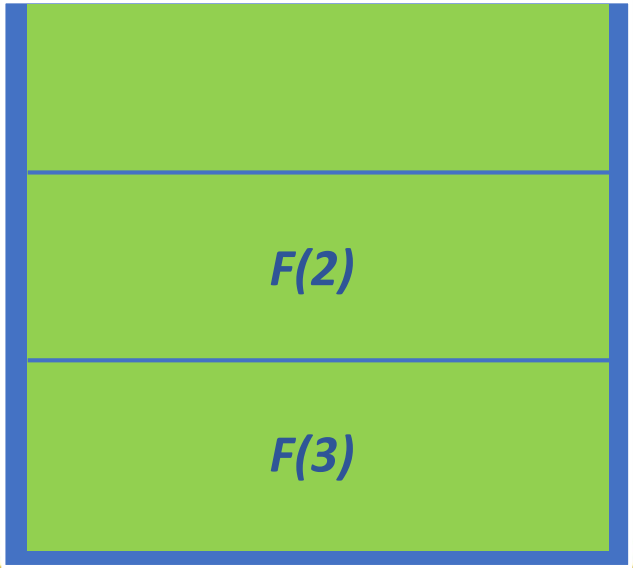
---

$F(2) = 2 \times 1$ — *Fase balik*

$F(3) = 3 \times 2$

$F(3) = 6$ — *Rekursi lengkap*

**Ilustrasi Fungsi Dalam Memori**



# Syarat-Syarat Rekursi

1. Permasalahan yang dihadapi dapat dipecah menjadi lebih sederhana
2. Ada fungsi yang memanggil dirinya sendiri
3. Karena fungsi rekursi memanggil dirinya sendiri secara terus menerus, maka dari itu diperlukan suatu kondisi untuk menghentikan pemanggilan ini, kondisi ini disebut dengan **kondisi terminal**
4. Fungsi rekursi minimal memiliki satu kondisi terminal

# Fungsi Rekursi Tanpa Batas Akhir

```
#include <stdio.h>
void tidak_Berhenti();

main(){
 tidak_Berhenti();
}

void tidak_Berhenti() {
 printf("Ctrl-Break untuk berhenti\n");
 tidak_Berhenti();
}
```

# Berhenti setelah n kali

```
#include <stdio.h>
#include <stdlib.h>

void berhenti_n_kali(int);
main(){
  int n;

  printf("Berapa kali rekursi : ");
  scanf("%d", &n);
  berhenti_n_kali(n);
}

void berhenti_n_kali(int n) {
  static int i=0;

  if (n<=0)
        exit(0);
  printf("%d kali\n",++i);
  berhenti_n_kali(n-1);
}
```

# Rekursi Tail

Sebuah proses dinyatakan sebagai rekursi tail adalah apabila permasalahan sudah diselesaikan pada saat mencapai kondisi terminal, sehingga proses tidak melakukan apapun saat fase balik

# Rekursi Tail

- Untuk menghitung n ! , rekursi tail didefinisikan sbb:

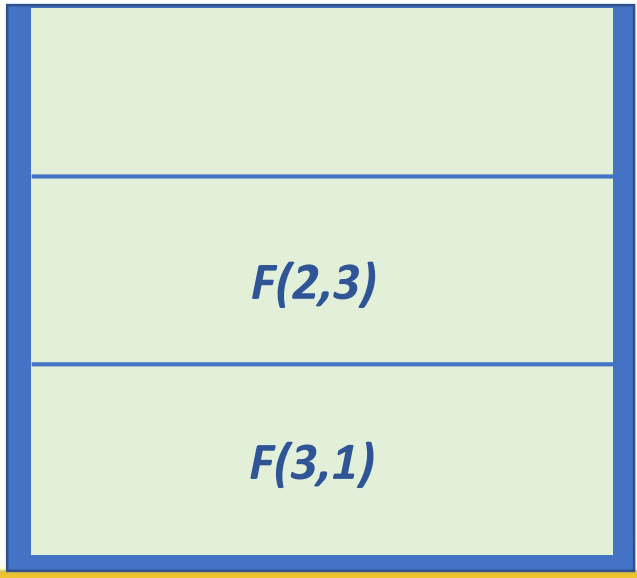
$$F(n,a) = \begin{cases} 0 & \text{jika } n<0 \\ a & \text{jika } n=0,\ n=1 \\ F(n-1,na) & \text{jika } n>1 \end{cases}$$

| | |
|---|---|
| $F(4,1)=F(3,4)$ | fase awal |
| $F(3,4)=F(2,12)$ | . |
| $F(2,12)=F(1,24)$ | . |
| $F(1,24)=24$ | kondisi terminal |
| 24 | fase balik<br>rekursi lengkap |

# Fase Pada Rekursi Tail

| | |
|---|---|
| *F(3,1) = F(2,3)* | *Fase awal* |
| *F(2,3) = F(1,6)* | |
| *F(1,6) = 6* | *Kondisi terminal* |
| *F(3,1) = 6* | *Fase balik*<br>*Rekursi lengkap* |

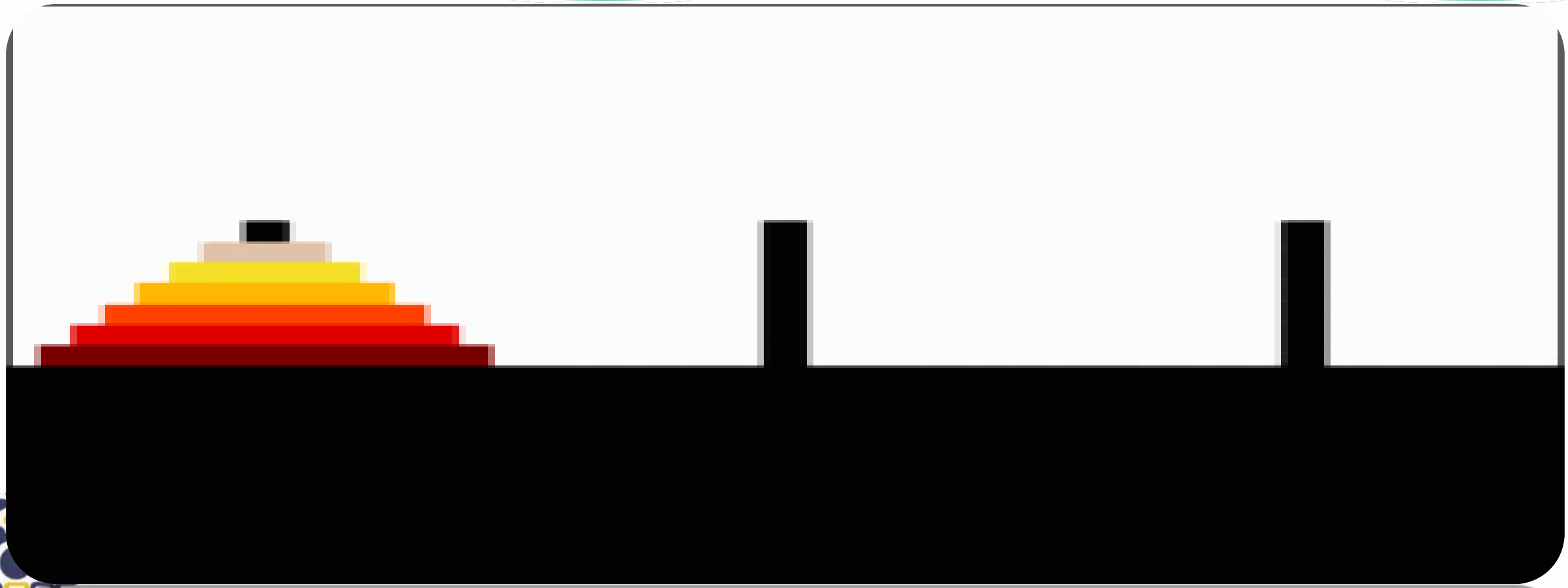
**Ilustrasi Fungsi Dalam Memori**



# PENGAPLIKASIAN REKURSI

- Menghitung faktorial
- Menghitung perpangkatan
- Mengecek bilangan prima
- Mencari FPB
- Menyelesaikan Menara Hanoi
- Membuat tree
- dll

# MENARA HANOI



# Fungsi Prima

$$F(n,a) = \begin{cases} 1 & \text{jika } a<2 \\ 0 & \text{jika } n\%a=0 \\ F(n,a-1) & \text{jika lainnya} \end{cases}$$

- `n` adalah bilangan yang akan dicek prima atau bukan
- `a` adalah `sqrt(n)`
- Keterangan : gunakan `math.h`

# Fungsi FPB

$$F(a,b) = \begin{cases} a & \text{jika b=0} \\ F(b, a\%b) & \text{jika lainnya} \end{cases}$$

- a dan b adalah 2 bilangan yang akan dicari FPB-nya

# Disusun oleh:

**Fikkri Prasetya 2110151010**

**Rafidah Atika 2110151016**

**Faza Zulfika Permana Putra 2110151023**

# Referensi


1. Sa'adah, Umi. 2007. Bab 5 Rekursi. Surabaya: Politeknik Elektronika Negeri Surabaya
2. https://www.youtube.com/watch?v=DHkaG-pZR_8 ( diakses 13 Juni 2016 )
3. http://www.petanikode.com/2016/06/memahami-cara-kerja-fungsi-rekursif.html (diakses 13 Juni 2016 )
4. https://m1perpustakaanmateri.files.wordpress.com/2011/10/a2-asd-fungsirekusif-gnt.ppt ( diakses 13 Juni 2016 )

bridge to the future

http://www.eepis-its.edu