Bab 5. Rekursi

Algoritma dan Struktur Data

Politeknik Elektronika Negeri Surabaya

2007

Umi Sa'adah
umi@eepis-its.edu

Overview

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

Umi Sa'adah
umi@eepis-its.edu

Pendahuluan

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

Umi Sa'adah
umi@eepis-its.edu

Konsep Dasar

- Rekursi adalah suatu proses yang bisa memanggil dirinya sendiri
- Dalam sebuah rekursi sebenarnya tekandung pengertian sebuah prosedur atau fungsi.
- Perbedaannya adalah bahwa rekursi bisa memanggil dirinya sendiri, sedangkan fungsi harus dipanggil melalui pemanggil fungsi (harus ada *function call*).

Umi Sa'adah
umi@eepis-its.edu

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)* . . . * (1)

Umi Sa'adah
umi@eepis-its.edu

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:

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

Umi Sa'adah
umi@eepis-its.edu

Fase Pada Rekursi

- **Fase awal**
  - Masing-masing proses memanggil dirinya sendiri.
  - Fase berhenti ketika pemanggilan telah mencapai kondisi terminal.
  - Kondisi teminal adalah kondisi dimana sebuah fungsi rekursi kembali dari pemanggilan, artinya fungsi tersebut sudah tidak memanggil dirinya sendiri dan kembali pada sebuah nilai.
  - Contoh : dalam penghitungan faktorial dari n, kondisi terminal adalah *n = 1, n = 0.*
  - Untuk setiap fungsi rekursi, minimal harus ada satu kondisi terminal.
- **Fase balik**
  - fungsi sebelumnya akan dikunjungi lagi dalam fase balik ini.
  - Fase ini berlanjut sampai pemanggilan awal, hingga secara lengkap proses telah berjalan

Umi Sa'adah
umi@eepis-its.edu

Fase pada fungsi rekursi

$F(4)=4x\ F(3)$	fase awal
$F(3)=3x\ F(2)$	.
$F(2)=2x\ F(1)$	.
$F(1)=1$	kondisi terminal
$F(2)=(2)x(1)$	fase balik
$F(3)=(3)x(2)$	.
$F(4)=(4)x\ (6)$	.
24	rekursi lengkap

Umi Sa'adah
umi@eepis-its.edu

Syarat Proses Rekursi

1. Permasalahan dapat dipecah menjadi lebih sederhana. Seperti contoh untuk permasalahan faktorial di atas, n! = (n-1)! * n. Dari permasalahan ini dapat dijelaskan bahwa n! dapat dipecah dengan melibatkan faktorial yang lebih sederhana, yaitu (n-1)!
2. Karena dengan rekursi kita memanggil fungsi secara berulang-ulang, maka harus ada suatu kondisi yang mengakhiri prosesnya yaitu kondisi terminal. Jika tidak, maka proses tidak akan pernah berhenti sampai memori yang digunakan tidak dapat menampung lagi. Sedangkan untuk rekursi pada fungsi faktorial, kondisi jika n=1 atau n=0 merupakan batas akhir proses rekursi dari fungsi tersebut.
3. Ada bagian program yang melakukan pemanggilan terhadap dirinya sendiri.

Umi Sa'adah
umi@eepis-its.edu

Rekursi Tail

- Sebuah proses rekursi dikatakan rekursi tail jika pernyataan terakhir yang akan dieksekusi berada dalam tubuh fungsi dan hasil yang kembali pada fungsi tersebut bukanlah bagian dari fungsi tersebut.
- Ciri fungsi rekursi tail adalah fungsi tersebut tidak memiliki aktivitas selama fase balik.
- Pemanggilan rekursi adalah pernyataan terakhir yang dieksekusi dalam aktivitas yang sedang berlangsung, sehingga tidak ada aktivitas yang harus dikerjakan pada saat pemanggilan kembali.

Umi Sa'adah
umi@eepis-its.edu

Rekursi Tail

- Untuk menghitung `n!`, rekursi tail didefinisikan sbb:

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

- Pada definisi ini digunakan parameter kedua, yaitu `a`, yang merupakan nilai utama dari hasil penghitungan faktorial secara rekursif.
- Pada `F(n,a)` → proses berlangsung untuk `n` mulai dari `(n-1)` s/d `1` dan nilai `a` mulai dari `1` diteruskan dengan `(n*a)` seterusnya sampai kondisi terminal `n = 1`

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 rekursi lengkap

Umi Sa'adah
umi@eepis-its.edu

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();
}
```


Umi Sa'adah
umi@eepis-its.edu

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);
}
```


Umi Sa'adah
umi@eepis-its.edu

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)`

Umi Sa'adah
umi@eepis-its.edu

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

Umi Sa'adah
umi@eepis-its.edu