Telkom University

Minggu 13 (Pengayaan)

Algoritma Rekursif

Algoritma Pemrograman – CII1F4

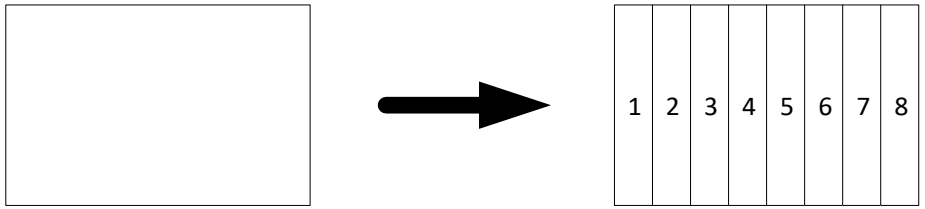
Fakultas Informatika

2021

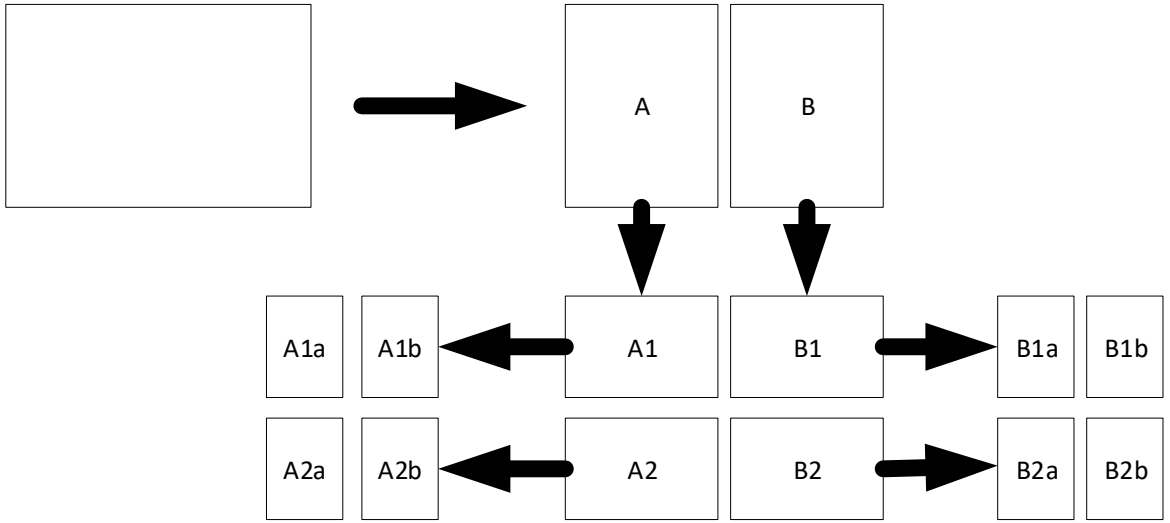
Definisi Rekursif

Rekursif secara sederhana dapat diartikan sebagai cara menyelesaikan suatu masalah dengan cara menyelesaikan sub-masalah yang identik dari masalah utama. Misalnya contoh kasusnya adalah membagi kertas ke dalam delapan bagian sama besar. Pada permasalahan ini terdapat 2 cara penyelesaian:

Tanpa Rekursif



Dengan Rekursif



Rekursif dalam Pemrograman

- Rekursif merupakan suatu teknik melakukan iterasi dengan cara membuat sebuah subprogram (fungsi atau prosedur) untuk memanggil dirinya sendiri dengan suatu cara tertentu.
- Pemanggilan terhadap dirinya sendiri tersebut **harus** berada didalam suatu kondisi, mungkin di dalam suatu struktur percabangan atau struktur iterasi.
- Berikut illustrasinya:

```
procedure A()
algoritma
    if <kondisi> then
        A()
    endif
endprocedure
```

```
function B() -> type
algoritma
    if <kondisi> then
        <hal-hal lain>
    else
        B()
    endif
endfunction
```

```
procedure C()
algoritma
    while <kondisi> do
        C()
    endwhile
endprocedure
```

Rekursif dalam Pemrograman

- Umumnya subprogram berbentuk rekursif memiliki parameter nilai yang selalu diubah pada setiap pemanggilan rekursif dan nilai tersebut diuji terhadap kondisi pada struktur kontrol yang digunakan.
- Dengan cara Ini dapat dipastikan bahwa pemanggilan rekursif tersebut dapat berakhir.
- Walaupun tidak selalu mudah dilakukan, pada dasarnya setiap bentuk iterasi menggunakan struktur kontrol iteratif dapat diubah menjadi bentuk rekursif, dan sebaliknya.
- Lihat contoh sederhana pencetakan deret 1..n
- Pada contoh ini, prosedur tersebut akan dipanggil dengan suatu nilai n. Misalnya **`count_std(10)`** untuk membuat deret dari 1 s.d. 10

```
procedure count_std(in n: integer)
kamus
    i : integer
algoritma
    i ← 1
    while i ≤ n do
        print(i)
        i ← i + 1
    endwhile
endprocedure
```

Contoh deret 1..N versi A

Pada contoh ini, prosedur rekursif menggunakan parameter i sebagai pengganti variabel iteratif i. Sehingga saat pertama kali pemanggilan, parameter tersebut diisi dengan 1. Misalnya untuk membuat deret dari 1 s.d. 10 dapat dipanggil sebagai **count_rekA(1,10).**

```
procedure count_rekA(in i, n: integer)
algoritma
    if i < n then
        print(i)
        count_rekA(i+1, n)
    endif
endprocedure
```

Contoh deret 1..N versi B

Seperti juga algoritma dengan cara iteratif, ada banyak variasi solusi untuk menjawab hal yang sama dengan cara rekursif. Bentuk berikut ini **menghindari** penggunaan parameter tambahan i.

Perhatikan letak dari pemanggilan rekursifnya! Juga perhatikan bahwa setiap pemanggilan rekursif, selalu ada nilai parameter yang diubah.

```
procedure count_rekB(n: integer)
algoritma
    if n > 0 then
        count_rekB(n-1)
        print(n)
    endif
endprocedure
```

Contoh deret 1..N versi B

Untuk lebih memahami bagaimana komputer meng-eksekusi program dengan rekursif, Berikut ini adalah tracing eksekusi rekursif deret versi B. Misalnya, prosedur rekursif dipanggil sebagai **count_rekB(5).**

```
procedure count_rekB(n: integer)
algoritma
    if n > 0 then
        count_rekB(n-1)
        print(n)
    endif
endprocedure
```

Output:

```
1  2  3  4  5
```

count_rekB(5) → count_rekB(4) → count_rekB(3) → count_rekB(2) → count_rekB(1) → count_rekB(0)

Rekursif di Ujung (Tail-End Recursion)

Bentuk rekursif versi A diatas disebut juga tail-end recursion, yaitu pemanggilan rekursif selalu merupakan instruksi terakhir dalam blok percabangan. Bentuk rekursif ini dapat dengan mudah dikonversi menjadi bentuk iteratif biasa dan sangat efisien.

Dibawah ini adalah format umum konversi antara bentuk iterasi biasa dengan bentuk rekursif, dimana:

- **<init(vars)>** : inisialisasi variabel[2] **vars** yang digunakan oleh iterasi
- **<kondisi(vars,parm)>**: kondisi loop dimana ekspresi menggunakan nilai dari **vars** dan parameter **parm.**
- **<proses(vars,parm)>**: proses utama yang dilakukan iterasi tersebut. Proses tersebut mungkin menggunakan nilai dari **vars** dan parameter **parm.**
- **<update(vars)>**: mengubah nilai **vars** agar loop tersebut berevolusi dan suatu saat dapat berakhir.
- Bagian sebelum loop dan setelah loop dapat diletakkan dalam prosedur lain. Dalam contoh bagian sebelum memasuki loop, yaitu inisialisasi **<init(vars)>** diletakkan diluar prosedur rekursifnya, dalam prosedur **recurse_master(parm).**

Rekursif di Ujung (Tail-End Recursion)

```
procedure iterate(parm)
algoritma
    <init(vars)>
    while <kondisi(vars,parm)> do
        <proses(vars,parm)>
        <update(vars)>
    endwhile
endprocedure
```

⇔

```
procedure recuse_master(parm)
algoritma
    <init(vars)>
    recurse(parm, vars)
endprocedure

procedure recurse(parm, vars)
algoritma
    if <kondisi(vars,parm)> then
        <proses(vars,parm)>
        <update(vars)>
        recurse(parm,vars)
    endif
endprocedure
```

Contoh Algoritma Rekursif

Berikut ini contoh algoritma rekursif dari beberapa algoritma iteratif yang telah kita kenal.

Contoh 1:
Faktorial n!=n×(n-1)×(n-2)⋯×2×1

```
function factorial_std(n:integer) → integer
{function dengan iterasi}
Kamus
    f : integer
algoritma
    f ← 1
    for i ← 2 to n do
        f ← f * i
    endfor
    return f
endfunction
```

```
{function dengan rekursif}
function factorial(n:integer) → integer
algoritma
    if n > 1 then
        return n*factorial(n-1)
    else
        return 1
    endif
endfunction

function factorial_master(n:integer) → integer
algoritma
    return factorial(n)
endfunction
```

Contoh 2: Fibonacci $fib_n = fib_{n-1} + fib_{n-2}$

```
{function dengan iterasi}
function fibonacci_std(n:integer) → integer
Kamus
    f0,f1,ft : integer
algoritma
    f0 ← 1
    f1 ← 1
    for i ← 1 to n do
        ft ← ft + f1
        f0 ← f1
        f1 ← ft
    endfor
    return f1
endfunction
```

```
{function dengan rekursif}
function fibonacci(n:integer) → integer
algoritma
    if n > 1 then
        return fibonacci(n-1)+fibonacci(n-2)
    else
        return 1
    endif
endfunction

func fibonacci_master(n:integer) → integer
algoritma
    return fibonacci(n)
endfunction
```

Contoh 3: Max

```
{program dengan iterasi}
program max_std
Kamus
    max_val, x : integer
algoritma
    input(max_val)
    if max_val != END_INPUT then
        input(x)
        while x != END_INPUT do
            if x > max_val then
                max_val ← x
            endif
            input(x)
        endwhile
    endif
    print(max_val)
endprogram
```

```
{program dengan rekursif}
function max_slave(maxv : integer) → integer
Kamus
    x : integer
algoritma
    input(x)
    if x == END_INPUT then
        return maxv
    else if x > maxv then
        return max_slave(x)
    else
        return max_slave(maxv)
    endif
endfunction

program max_master
Kamus
    max_val : integer
algoritma
    input(max_val)
    if max_val != END_INPUT then
        max_val ← max_slave(max_val)
    print(max_val)
endprogram
```

Contoh 4: Search

```
Type tabStr : array [0...NMAX] of string

{function dengan iterasi}
function search_std(T:tabStr, N: integer, X:string) → boolean
Kamus
    i : integer
algoritma
    i ← 0
    found ← false
    while i < N and not found do
        found ← T[i] == X
        i ← i + 1
    endwhile
    return found
endfunction
```

```
{function dengan rekursif}
function search_slave(T:tabStr,i,N:integer,X:string) → boolean
algoritma
    if i != N-1 and T[i] != X then
        return search_slave(T,i+1,N,X)
    else
        return T[i] == X
    endif
endfunction

function search_master(T:tabStr, N: integer ,X:string) → boolean
algoritma
    return search_slave(T,0,N,X)
Endfunction
```

Contoh 5: Insertion Sort

```
Type tabStr : array [0...NMAX] of string
{procedure dengan iterasi}
procedure insertionsort_std(in/out T:tabStr, in N:integer)
kamus
    pass, i : integer
    temp : string
algoritma
    for pass <-1 to N-1 do
        temp <- T[pass]
        i <- pass
        while i > 0 and temp < T[i-1] do
            T[i] <- T[i-1]
            i <- i - 1
        endwhile
        T[i] <- temp
    endfor
endprocedure
```

```
{procedure dengan rekursif}
procedure insert(in/out T:tabStr, in i:integer, temp:string)
algoritma
    if i > 0 and temp < T[i-1] then
        T[i] <- T[i-1]
        insert(T,i-1,temp)
    else
        T[i] <- temp
    endif
endprocedure
procedure insertionsort(in/out T:tabStr, in pass, N:integer)
algoritma
    if pass <= N-1 then
        insert(T,pass,T[pass])
        insertionsort(T,pass+1,N)
    endif
endprocedure
procedure insertionsort_master(in/out T:tabStr, in N:integer)
algoritma
    insertionsort(T, 1, N)
endprocedure
```

Soal Latihan

- Buat algoritma rekursif untuk menghitung rata-rata dari n bilangan riil

- Buat algoritma rekursif untuk mencetak seluruh bilangan ganjil positif dari input yang diakhiri dengan marker -999.

- Buat algoritma rekursif selection sort untuk n bilangan integer dalam array.
- Buat algoritma rekursif binary search untuk n bilangan integer terurut membesar dalam array.

TERIMA KASIH