

# Algoritma & Pemrograman

*PENGULANGAN*

## Pendahuluan

- Salah satu kelebihan komputer dibandingkan dengan manusia adalah kemampuannya untuk melaksanakan suatu instruksi berulang kali tanpa mengenal lelah dan bosan.
- Di dalam algoritma pengulangan atau kalang (repetition atau loop) dapat dilakukan sejumlah kali pengulangan atau sampai kondisi berhenti pengulangan tercapai.

## Struktur Pengulangan

Secara umum terdiri atas dua bagian :

- *kondisi pengulangan*, yaitu ekspresi boolean yang harus dipenuhi untuk melaksanakan pengulangan yang dinyatakan secara eksplisit oleh pemrogram atau implisit oleh komputer.
- *badan pengulangan*, yaitu bagian algoritma yang diulang.

## Struktur Pengulangan

Struktur pengulangan biasanya (opsional) disertai bagian :

- *inisialisasi*, yaitu aksi yang dilakukan sebelum pengulangan dilakukan pertama kali.
- *Terminasi*, yaitu aksi yang dilakukan setelah pengulangan selesai dilaksanakan.

## Struktur umum :

<inisialisasi>  
awal pengulangan  
  badan pengulangan  
akhir pengulangan  
<terminasi>

## notasi struktur pengulangan :

- Struktur FOR
- Struktur WHILE
- Struktur REPEAT

## Struktur FOR

- Digunakan untuk menghasilkan pengulangan sejumlah kali yang dispesifikasikan. Jumlah pengulangan diketahui atau dapat ditentukan sebelum eksekusi.

## FOR menaik

```
For pencacah ← nilai_awal to nilai_akhir do  
    Aksi  
Endfor
```

Keterangan :

- pencacah harus bertipe data yang memiliki predesesor dan successor (integer atau karakter).
- aksi adalah satu atau lebih instruksi yang diulang.
- nilai-awal harus = nilai-akhir, jika tidak maka badan pengulangan tidak dimasuki.
- nilai pencacah diinisialisasi dengan nilai\_awal kemudian secara otomatis bertambah satu sampai nilai\_akhir tercapai.
- banyak pengulangan yaitu (nilai\_akhir - nilai\_awal +1).

## Contoh 3

### Algoritma Cetak\_Angka

{mencetak 1, 2, ..., 8 ke piranti keluaran}

Deklarasi :

K: integer

Deskripsi :

for k ← 1 to 8 do

write (k)

endfor

## Contoh 2

### Algoritma Cetak\_Kalimat

{mencetak Kalimat "C++ dan Pascal"  
    sebanyak 5 kali ke piranti keluaran}

Deklarasi :

K: integer

Deskripsi :

for k ← 1 to 5 do

write ('C++ dan Pascal')

endfor

## Latihan 1 :

### Algoritma Penjumlahan

{menghitung jumlah nilai 1 .. 10 yang dibaca dari piranti masukan dan hasilnya dicetak dipiranti keluaran}

## Contoh 3

### Algoritma Hitung\_rata

{mengitung rata-rata N buah bilangan bulat yang dibaca dari piranti masukan}

#### Deklarasi :

N: integer {jumlah data, > 0}

x: integer {bilangan bulat yg dibaca dari papan kunci}

k: integer {pencacah banyaknya pengulangan}

jumlah : integer {pencacat jumlah nilai}

rerata: integer {rata-rata nilai}

## Contoh

Deskripsi :

read (N)

Jumlah  $\leftarrow$  0

for k  $\leftarrow$  1 to N do

read (x)

    jumlah  $\leftarrow$  jumlah + x

endfor

rerata  $\leftarrow$  jumlah/N

write (rerata)

## FOR menurun

For pencacah  $\leftarrow$  nilai\_akhir downto nilai\_awal do

    Aksi

Endfor

Keterangan :

- pencacah harus bertipe data yang memiliki predesesor dan successor (integer atau karakter).
- aksi adalah satu atau lebih instruksi yang diulang.
- nilai-akhir harus = nilai-awal, jika tidak maka badan pengulangan tidak dimasuki.
- nilai pencacah diinisialisasi dengan nilai\_akhir kemudian secara otomatis berkurang satu sampai nilai\_awal tercapai.
- banyak pengulangan yaitu (nilai\_akhir - nilai\_awal + 1).

## Algoritma Peluncuran\_Roket

{hitung mundur peluncuran roket}

Deklarasi :

K: integer

Deskripsi :

for k ← 100 downto 0 do

write (k)

endfor

write ('GO') {roket meluncur}

## Struktur WHILE

Notasi algoritmik :

while kondisi do

    Aksi

Endwhile

Keterangan :

**Aksi** ( atau runtunan aksi) akan dilaksanakan berulang kali selama **kondisi** bernilai **true**, jika false maka badan pengulangan tidak akan dilaksanakan yang berarti pengulangan selesai.

### Algoritma Cetak\_Angka

{mencetak 1, 2, ..., 8 ke piranti keluaran}

Deklarasi :

K: integer

Deskripsi :

$K \leftarrow 1$  {inisialisasi}

while  $k \leq 8$  do

write (k)

$k \leftarrow k + 1$

endwhile

### Contoh 3

#### Algoritma Hitung\_rata

{mengitung rata-rata N buah bilangan bulat yang dibaca dari piranti masukan}

Deklarasi :

N: integer {jumlah data,  $> 0$ }

x: integer {bilangan bulat yg dibaca dari papan kunci}

k: integer {pencacah banyaknya pengulangan}

jumlah : integer {pencacat jumlah nilai}

rerata: integer {rata-rata nilai}

Deskripsi :

read (N)

Jumlah  $\leftarrow$  0

while k  $\leq$  N do

read (x)

jumlah  $\leftarrow$  jumlah + x

k  $\leftarrow$  k + 1

endwhile

rerata  $\leftarrow$  jumlah/N

write (rerata)

## Struktur REPEAT

Notasi algoritmik :

repeat

Aksi

Until kondisi

Keterangan :

- **Aksi** ( atau runtunan aksi) akan dilaksanakan berulang kali sampai **kondisi** bernilai **true**, jika **kondisi** bernilai **false** maka pengulangan masih terus dilakukan.

## Contoh

### Algoritma Cetak\_Angka

{mencetak 1, 2, ..., 8 ke piranti keluaran}

Deklarasi :

K: integer

Deskripsi :

$K \leftarrow 1$       {inisialisasi}

repeat

write (k)

$k \leftarrow k + 1$

until  $k > 8$

## Contoh 3

### Algoritma Hitung\_rata

{mengitung rata-rata N buah bilangan bulat yang dibaca dari piranti masukan}

Deklarasi :

N: integer {jumlah data,  $> 0$ }

x: integer {bilangan bulat yg dibaca dari papan kunci}

k: integer {pencacah banyaknya pengulangan}

jumlah : integer {pencacat jumlah nilai}

rerata: integer {rata-rata nilai}

Deskripsi :

read (N)

Jumlah  $\leftarrow$  0

repeat

read (x)

    jumlah  $\leftarrow$  jumlah + x

    k  $\leftarrow$  k + 1

until k > N

rerata  $\leftarrow$  jumlah/N

write (rerata)

## Struktur WHILE atau REPEAT

- Untuk menyelesaikan persoalan-persoalan tertentu dapat digunakan struktur while ataupun repeat, tetapi pada kebanyakan kasus, pemilihan struktur while atau repeat bergantung pada natural dari persoalan. Ini artinya ada persoalan yang hanya benar bila menggunakan struktur while tetapi bisa salah bila menggunakan repeat atau sebaliknya.

## Perbedaan Mendasar

- Pada **REPEAT**, kondisi pengulangan diperiksa pada akhir pengulangan. Jadi, instruksi di dalam badan pengulangan dilaksanakan dulu, barulah kondisi diperiksa, yang berarti pengulangan dilaksanakan minimal satu kali.
- Pada **WHILE**, kondisi pengulangan diperiksa di awal pengulangan. Jadi instruksi di dalam badan pengulangan hanya dapat dilaksanakan bila kondisi bernilai true, yang berarti badan pengulangan mungkin tidak akan pernah dilaksanakan.