

# **PENGULANGAN**

# **PENGULANGAN**

## **Struktur Pengulangan**

Struktur pengulangan terdiri atas 2 bagian yaitu :

1. Kondisi pengulangan, yaitu ekspresi boolean yang harus dipenuhi untuk melaksanakan pengulangan. Kondisi ada yang dinyatakan secara explisit oleh pemrogram.
2. Badan (body) pengulangan, yaitu satu atau lebih aksi yang akan diulang

Dalam algoritma terdapat beberapa macam struktur pengulangan yang berbeda, beberapa struktur dapat dipakai untuk masalah yang sama, namun ada notasi pengulangan yang hanya cocok dipakai untuk masalah tertentu, struktur pengulangan tersebut adalah :

## **Perntaraan FOR**

Struktur FOR digunakan untuk menghasilkan pengulangan sejumlah kali tanpa penggunaan kondisi apapu, struktur ini menyebabkan aksi diulangi sejumlah kali (tertentu). Bentuk umum struktur FOR ada 2 macam : menaik *(ascending)* dan menurun *(descending)*.

**FOR menaik** :

|  |
| --- |
| For peubah ← nilai\_awal to nilai\_akhir doAksiEndfor |

Keterangan :

* 1. Peubah : haruslah bertipe sederhana.
	2. Nilai\_awal : haruslebih kecil atau sama dengan nilai\_akhir.
	3. Pada awalnya, peubah diinisialisasi dengan nilai\_awal. Nilai peubah secara otomatis bertambah satru setiap kali aksi pengulangan dimasuki, sampai akhirnya nilai peubah sama dengan nilai\_akhir.

Ini adalah contoh pada contoh WHILE-DO

Contoh 1 :

**Algoritma cetak\_banyak\_halo**

*{ DEKLARASI }*

 K : integer *{ pencacah pengulangan }*

*{ DESKRIPSI* *}*

 for K ← 1 to 10 do

 write (‘HALO’)

 endfor

*{ kondisi berhenti : k > 10 }*

## **Pernyataan WHILE**

|  |
| --- |
| while *<kondisi>* do*Aksi*endwhile |

Bentuk umum struktur WHILE-DO adalah aksi (atau runtunan aksi) akan dilaksanakan berulangkali sepanjang *<kondisi>* boolean masih tetap bernilai *true*, jika *<kondisi>* bernilai *false*, badan pengulangan tidak akan dilaksanakan. Pengulangan selesai.

**Contoh 1:**

Tuliskan algoritma untuk mencetak banyak HALO sebanyak 10 kali .

**Algoritma cetak\_banyak\_halo**

*{ DEKLARASI }*

K : integer *{ pencacah pengulangan }*

*{ DESKRIPSI }*

K ← 1 *{ inisialisasi }*

 while k ≤ 10 do

 write (‘HALO’)

K ← K+1

 endwhile

*{ kondisi berhenti : k > 10 }*

**Contoh 2 :**

Tuliskan Algoritma untuk mencetak urutan angka 1 s/d 10

**Algoritma cetak\_angka**

*{ Deklarasi }*

 Angka : integer

*{ Deskripsi }*

Angka ← 1

 while angka ≤ 10 do

 write (angka)

angka ← angka +1

 endwhile

## **Pernyataan REPEAT**

|  |
| --- |
| *Repeat**Aksi* *Until <kondisi>* |

Struktur REPEAT-UNTIL memiliki makna yang sama dengan WHILE-DO namun ada perbedaan mendasar diantara keduanya. Pada struktur REPEAT-UNTIL aksi (atau sekumpulan aksi) dilaksanakan minimal satu kali, karena kondisi pengulangan diperiksa pada **akhir struktur**, sedangkan pada struktur WHILE-DO kondisi pengulangan diperiksa pada **awal struktur** sehingga memungkinkan pengulangan tidak pernah dilaksanakan bila kondisi pengulangan bernilai false.

Ini adalah contoh pada contoh WHILE-DO

**Contoh 1 :**

**Algoritma cetak\_banyak\_halo**

*{ DEKLARASI }*

K : integer *{ pencacah pengulangan }*

*{ Deskripsi}*

 K ← 1 *{inisialisasi}*

 repeat

 write (‘HALO’)

 K ← K+1

 until k > 10

*{ kondisi berhenti : k > 10 }*

## **WHILE atau REPEAT ?**

Dalam beberapa masalah, pemilihan WHILE atau REPEAT bergantung pada kemurnian masalah itu sendiri. Ada masalah yang hanya benar bila menggunakan struktur WHILE, tetapi bisa ada yang fatal bila menggunakan REPEAT. Perbedaannya terdapat pada pemeriksaan proses pengulangan pada saat diekseskusi oleh program. REPEAT memeriksa diakhir, dan WHILE sebaliknya.

Berdasarkan perbedaannya pemakiana yang tepat untuk kedua pengulangan tersebut adalah :

* 1. Penggunaan konstruksi WHILE pada kasus tertentu mengharuskan pemeriksaan kondisi objek sebelum objek tersebut dimanipulasi.
	2. Penggunaan konstruksi REPEAT pada kasus tertentu mendahulukan manipulasi objek kemudian memeriksa kondisi objek tersebut.

**WHILE (benar).**

PROGRAM PenjumlahanDeretPecahan

*{ Menghitung jumlah seluruh nilai pecahan dari sejumlah data bilangan*

 *bulat (x) yang dibaca dari papan ketik. Akhir pembacaan adalah bila*

 *nilai x yang dibaca = 0 }*

*DEKLARASI*

x : integer *{ data bilangan bulat yang dibaca }*

S : real *{ jumlah deret }*

ALGORITMA :

S ← S + 1/x { inisiasi jumlah deret }

read(x)

while x ≠ 0 do

 S ← S + 1/x

 read(x)

endwhile

{ *x = 0* } { kondisi setelah pengulangan berhenti }

write(S)

Keterangan : Penggunaan WHILE pada Algoritma di atas benar, karena

sebelum dijumlahkan, x diperiksa di awal pengulangan apakah sudah 0

atau belum 0.

**REPEAT (benar).**

PROGRAM SimulasiMenuProgram

*{ Menampilkan menu, membaca pilihan menu, dan menampilkan nomor menu*

 *yang dipilih oleh pengguna }*

*DEKLARASI*

NomorMenu : integer

ALGORITMA :

repeat

 { Cetak menu }

 write(‘ MENU ‘)

 write(‘ 1. Baca data ‘)

 write(‘ 2. Cetak data ‘)

 write(‘ 3. Ubah data ‘)

 write(‘ 4. Hapus data ‘)

 write(‘ 5. Keluar Program ‘)

 write(‘ Masukan pilihan anda (a/2/3/4/5) ? ‘)

 read(NomorMenu) { baca nomor menu yang akan dipilih }

 case NomorMenu

 1 : write(‘ Anda memilih menu nomor 1 ’)

 2 : write(‘ Anda memilih menu nomor 2 ’)

 3 : write(‘ Anda memilih menu nomor 3 ’)

 4 : write(‘ Anda memilih menu nomor 4 ’)

 5 : write(‘ Keluar program ’)

 endcase

until NomorMenu = 5

Keterangan : Konstruksi REPEAT benar untuk masalah ini sebab, nomor

pilihan menu dibaca terlebih dahulu, barulah kemudian diperiksa

diakhir pengulangan apakah nomor menu yang dipilih = 5.