Pengulangan

Pola yang Berulang

- Misal 1, 2, 3, 4, 5, 6 …
- Jika sebuah elemen disimpan dalam i, maka bias dilihat pola
  - i←i+1
- Bandingkan dengan deret 2, 4, 6, 8, 10, …
- Jika sebuah elemen disimpan dalam i, maka bias dilihat pola
  - i←i+2
- Contoh :
- Pola 2, 4, 9, 16, 25

Struktur Pengulangan

- Secara umum struktur terdiri dari :

1. Kondisi Pengulangan
2. Badan (Body) pengulangan

- Struktur pengulangan biasanya disertai dengan bagian :

1. Inisialisasi : aksi yang harus dilakukan sebelum perulangan
2. Terminasi : aksi yang harus dilakukan setelah perulangan

- Inisialisasi dan terminasi tidak selalu harus ada

Kontruksi Perulangan

- For
- While
- Repeat

Pengulangan FOR

- Struktur for digunakan untuk melakukan pengulangan sejumlah kali yang telah di spesifikasi sebelumnya.
- Jumlah pengulangan diketahui dan dapat ditentukan sebelum eksekusi program untuk pencacah berapa kali pengulangan dilakukan
- Kita memerlukan sebuah peubah penjajah atau counter yang berubah nilainya selalu bertambah satu secara otomatis setiap kali pengulangan dilakukan.
- Jika cacah pengulangan sudah mencapai jumlah pengulangan yang dispesifikasikan maka Proses pengulangan selesai bentuk umum konstruksi for ada dua macam yaitu menaik (ascending) dan menurun (descending)

Pengulangan FOR

- For menaik

```
For pencacah ← <nilai_awal> to <nilai_akhir> do
      aksi
End for
```

a) Pencacah haruslah peubah dari tipe data yang memiliki predecessor dan suksesor dalam hal ini hanya integer atau karakter, tipe riil yang tidak dapat digunakan sebagai pencacah

b) Aksi dapat berupa satu atau lebih instruksi

c) Nilai awal harus lebih kecil atau sama dengan nilai akhir Jika nilai awal lebih besar dari nilai akhir maka badan pengulangan tidak dimasuki

d) Pada awal pengulangan pencacah diinisialisasi dengan nilai awal nilai pecah secara otomatis bertambah satu setiap kali badan pengulangan dimasuki sampai akhirnya nilai pencacah sama dengan nilai akhir

e) Jumlah pengulangan yang terjadi adalah nilai akhir - nilai awal ditambah 1

Pengulangan FOR

- **Contoh**

```
For i←1 to 10 do
    write (‘Hello World’)
End
N=100 // read (N)
For i←1 to N do
    write (‘Hello World’)
end
```

Pengulangan FOR

- Contoh

```
For i←1 to 10 do
   write (‘Hello World’)
End
N=100 // read (N)
For i←1 to N do
   write (i)
End
```

- For (i=1;**i<=10**;i++)
  - Printf(‘Helloword’);

Instruksi Penjumlahan

```
sum=0
For i←1 to N do
     sum← sum+i
     write (sum)
End
Contoh :
Hitung rata-rata
```

Pengulangan FOR

- For menurun

```
For pencacah ← <nilai_akhir> downto <nilai_awal> do
      aksi
End for
```

a) Pencacah haruslah peubah dari tipe data yang memiliki predecessor dan suksesor dalam hal ini hanya integer atau karakter, tipe riil yang tidak dapat digunakan sebagai pencacah

b) Aksi dapat berupa satu atau lebih instruksi

c) Nilai awal harus lebih kecil atau sama dengan nilai akhir Jika nilai awal lebih besar dari nilai akhir maka badan pengulangan tidak dimasuki

d) Pada awal pengulangan pencacah diinisialisasi dengan nilai awal nilai pecah secara otomatis bertambah satu setiap kali badan pengulangan dimasuki sampai akhirnya nilai pencacah sama dengan nilai akhir

e) Jumlah pengulangan yang terjadi adalah nilai akhir - nilai awal ditambah 1

Pengulangan FOR

- Contoh

```
For i←10 downto 1 do
    write (‘Hello World’)
End
N=100 // read (N)
For i←N downto 1 do
    write (‘Hello World’)
end
```

Konstruksi While

- Bentuk umum konstruksi While

```
While kondisi do
      aksi
End while
```

- Yang secara harfiah berarti selama kondisi memenuhi, kerjakan aksi

Konstruksi While

- sebelum memasuki badan pengulangan kondisi diperiksa dulu apakah masih memenuhi atau benar atau sudah tidak memenuhi atau salah
- aksi dikerjakan berulang selama kondisi Masih benar jika kondisi salah maka badan pengulangan tidak akan dimasuki yang berarti pengulangan selesai
- kondisi di akhir pengulangan setelah n disebut looking varian yaitu merubah kondisi yang nilainya sudah tidak berubah lagi
- yang harus diperhatikan adalah pengulangan suatu saat harus berhenti pengulangan yang tidak pernah berhenti disebut looping menandakan bahwa ada algoritma yang salah
- Pengulangan akan berhenti apabila kondisi bernilai false, agar kondisi berubah menjadi false maka di dalam badan pengulangan harus ada instruksi yang mengubah nilai kondisi

Contoh

Program CetakHelloWorld

```
Deklarasi :
     i, N : integer
Algoritma :
     read N;
     i←1
     while i <= N do
          write (‘Hello World’)
          i← i+1
    end while
    //i>N → loop invariant
```

- Kesalahan umum lupa inisialisasi nilai awal I dan lupa menulis instruksi untuk mengubah nilai pencacah

Kapan sebaiknya menggunakan WHILE

- sekilas antara for dan while sama saja kegunaannya namun WHILE memiliki keunggulan yang tidak dimiliki oleh FOR.
- Pada kasus-kasus dimana jumlah pengulangan diketahui di awal program while dapat digunakan sebaik penggunaan for seperti pada contoh di atas namun untuk proses yang jumlah pengulangan tidak dapat ditentukan di awal hanya struktur WHILE yang dapat kita gunakan sebab kondisi pengulangan diperiksa di awal pengulangan jadi meskipun kita tidak mengetahui kapan persisnya WHILE berhenti tetapi kita menjamin bahwa jika kondisi bernilai salah maka pengulangan pasti berhenti.

Contoh mencetak angka 1, 2, 3, …, N

Contoh Menjumlah deret 1+2+3+…+N

Contoh

- Menghitung rata-rata nilai ujian yg diinput di read dari keyboard, read akan terus berjalan hingga ada yg memasukkan nilai negative.

Konstruksi REPEAT

```
Repeat
    aksi
Until kondisi
```

- Yang secara harfiah berarti ulangi aksi sampai kondisi terpenuhi

Konstruksi REPEAT

- Konstruksi REPEAT berdasarkan pengulangan pada kondisi yang bernilai boolean
- pemeriksaan kondisi dilakukan pada akhir setiap pengulangan
- aksi dikerjakan berulang-ulang sampai kondisi terpenuhi atau bernilai true dengan kata lain jika kondisi masih false pengulangan masih terus dilakukan
- karena Proses pengulangan suatu saat harus berhenti maka di dalam badan pengulangan harus ada instruksi yang mengubah nilai peubah kondisi
- konstruksi REPEAT memiliki makna yang serupa dengan WHILE dan dalam beberapa masalah Kedua konstruksi tersebut komplemen satu sama lain

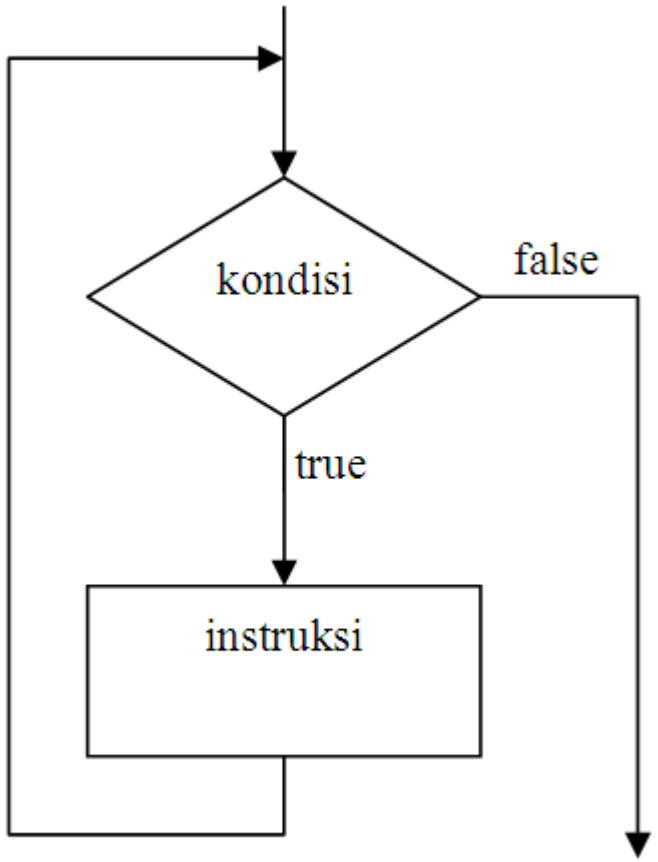
Contoh

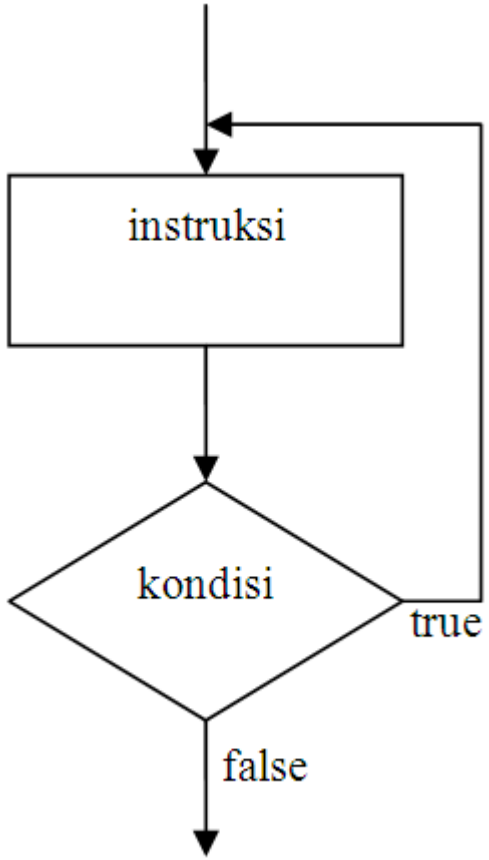
Program CetakHelloWorld

```
Deklarasi :
     i, N : integer
Algoritma :
     read N;
     i←1
     repeat
        write (‘Hello World’)
        i← i+1
     until i > N
    //i>N → loop invariant
```

- Kesalahan umum lupa inisialisasi nilai awal I dan lupa menulis instruksi untuk mengubah nilai pencacah

Perbandingan while
dengan do-while (repeat until)





WHILE atau REPEAT

- while dan repeat dapat ekuivalen penggunaannya
- Namun ada persoalan tertentu yang kita harus memilih while atau repeat bergantung pada natural persoalan itu sendiri ini artinya ada masalah yang hanya benar bila menggunakan struktur While tetapi bisa fatal bila menggunakan Repeat
- untuk mengetahui struktur mana yang lebih tepat kita harus mengetahui perbedaan mendasar di antara keduanya perbedaannya adalah sebagai berikut :
  - pada konstruksi repeat kondisi pengulangan diperiksa pada akhir pengulangan jadi instruksi di badan pengulangan dilaksanakan dulu barulah pengetesan kondisi dilakukan konsekuensinya badan pengulangan dilaksanakan paling sedikit 1 kali
  - sebaliknya para instruksi while kondisi pengurangan diperiksa di awal pengulangan jadi instruksi di dalam badan keuangan hanya dapat dilaksanakan bila pengetesan kondisi menghasilkan nilai true, konsekuensinya badan pengulangan mungkin tidak akan pernah dilaksanakan bila kondisi pengulangan pertama kali bernilai false

WHILE atau REPEAT

- berdasarkan perbedaan diatas maka kita dapat menarik kesimpulan Kapan menggunakan while dan kapan menggunakan repeat
  - Gunakan konstruksi while pada kasus yang mengharuskan terlebih dahulu pemeriksaan kondisi objek sebelum objek tersebut dimanipulasi
  - Gunakan konstruksi repaet pada kasus yang terlebih dahulu memanipulasi objek baru kemudian memeriksa kondisi objek tersebut

Konstruksi CASE

- Untuk masalah dua atau lebih kasus, konstruksi CASE dapat menyederhanakan penulisan IF-THEN-ELSE yang bertingkat-tingkat. Konstruksi CASE sbb:

case (ekspresi): //nilai1, nilai2, nilai3,… nilai99

    nilai1 : aksi1

    nilai2 : aksi2

    nilai3 : aksi3

    …

    otherwise aksiX

End case

Contoh

- Program nama hari

```
Deklarasi :
no_hari : integer
Algoritma :
    read (no_hari)
    case (no_hari):
          1 : write (‘senin’)
          2 : write (‘selasa’)
           …
          7 : write (‘minggu’)
          otherwise write (‘No hari tidak valid’)
    endcase
```

Contoh

- Program GanjilGenap

```
Deklarasi :
bilangan : integer
Algoritma :
    read (bilangan)
    case (bilangan mod 2):
          0 : write (‘Genap’)
          1 : write (‘Ganjil’)
          otherwise write (‘input tidak valid’)
    endcase
```