Bab 4
Perintah Perulangan

Proses perulangan (loop) adalah proses yang mengerjakan satu atau lebih statement lainnya secara berulang-ulang. Pada bahasa BASIC terdapat beberapa perintah untuk proses perulangan yaitu WHILE..WEND, DO..LOOP UNTIL dan FOR...NEXT.

4.1 Penggunaan GOTO untuk proses perulangan

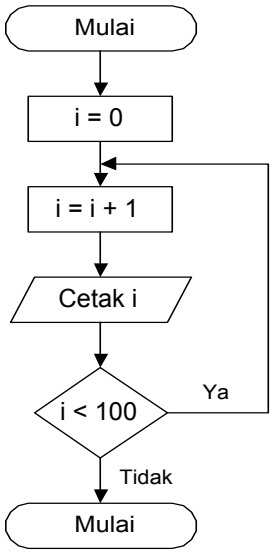
Pada compiler BASIC versi lama seperi BASICA atau GWBASIC perintah GOTO sangat populer digunakan untuk proses perulangan. Pada saat itu teknik pemrograman terstruktur masih belum populer digunakan. Saat ini hampir semua software compiler seperti Turbo BASIC, Turbo Pascal, Turbo C, dll. masih menyediakan fasilitas GOTO tetapi pemakaiannya sangat tidak dianjurkan atau sebaiknya dihindari. Hal ini disebabkan karena pemakaian perintah GOTO dapat menyebabkan suatu program menjadi tidak terstruktur karena alur logikanya loncat-loncat. Tetapi buku ini masih membahas penggunaan perintah GOTO karena untuk programmer pemula perintah ini paling mudah dipahami.

Perintah GOTO selalu berpasangan dengan label. Label digunakan untuk menandai atau menamai suatu baris program yang akan dituju dengan perintah GOTO. Pada compiler BASIC versi lama setiap baris program selalu diberi label yang berupa nomer baris seperti terlihat pada contoh dibawah ini.

Program 4.1. :

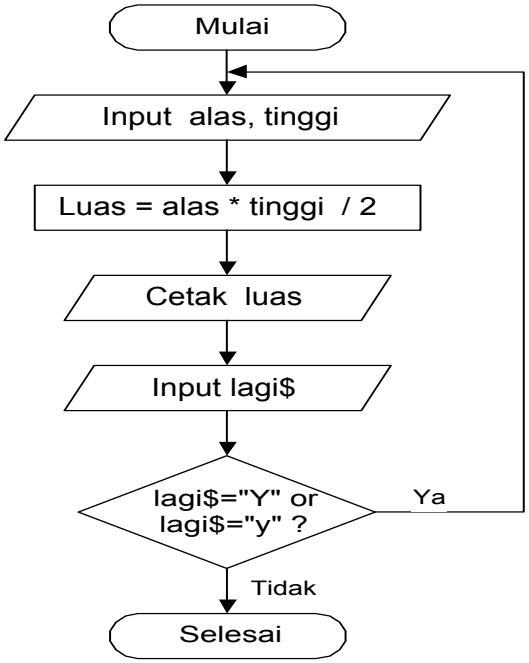
```
10 REM Contoh Program yang menggunakan perintah GOTO
20 REM untuk proses perulangan
30 REM Program ini untuk menghitung luas segitiga
40 CLS
50 INPUT "Masukkan alas :",alas
60 INPUT "Masukkan tinggi :",tinggi
70 luas = alas * tinggi /2
80 PRINT "Luas segitiga= ",luas
90 INPUT "Mau menghitung lagi (Y/N) ?",lagi$
100 IF lagi$="Y" OR lagi$="y" THEN GOTO 40
110 END
```

Jika program di atas digambarkan dalam bentuk flowchart, bentuknya adalah sebagai berikut :



Gambar 4.1. Flowchart untuk program 4.1

Algorithma untuk mencetak deret urut 1 s/d 100

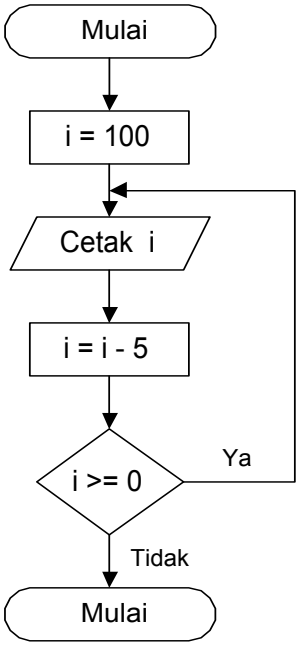


Gambar 4.2. Flowchart untuk mencetak deret 1..100

Program 4.2 :

```
REM Contoh Program yang menggunakan perintah GOTO
REM Program untuk mencetak deret 1 s/d 100
cls
i=0
awal :
  i=i+1
  print i;
  if i<100 then goto awal
end
```

Algorithma untuk mencetak deret turun 100 95 90 …5 0



Gambar 4.3. Flowchart untuk mencetak deret turun 100 95...5 0

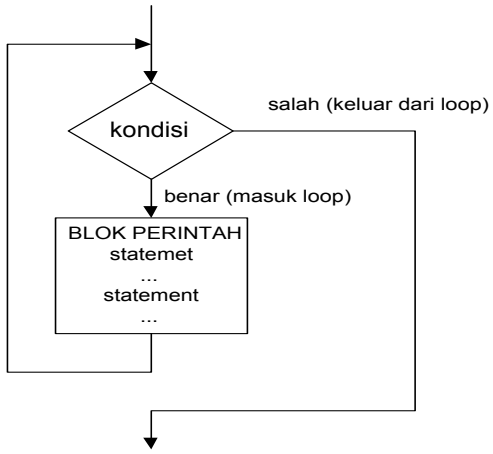
Program 4.3. :

```
REM Contoh Program yang menggunakan perintah GOTO
REM Program untuk mencetak deret 100  95 90 …. 10 5 0
cls
i=100
awal :
   print i;
   i=i-5
if i>=0 then goto awal
end
```

4.2. WHILE…WEND

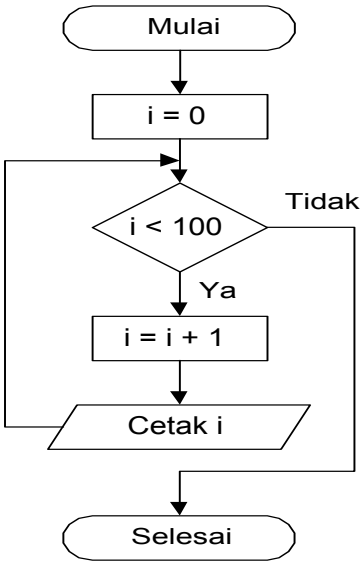
Perintah WHILE…WEND adalah perintah yang berpasangan. WHILE sebagai awal dari loop dan WEND sebagai penutupnya. Loop akan terus dikerjakan selama kondisi benar (kondisi ditulis di awal loop tepatnya setelah perintah WHILE).

Bentuk loop while..wend dapat digambarkan seperti berikut ini :



Gambar 4.4. Diagram loop WHILE..WEND

Algorithma untuk mencetak deret 1 s/d 100 menggunakan WHILE..WEND



Gambar 4.5. Mencetak deret 1..100 dengan WHILE..WEND

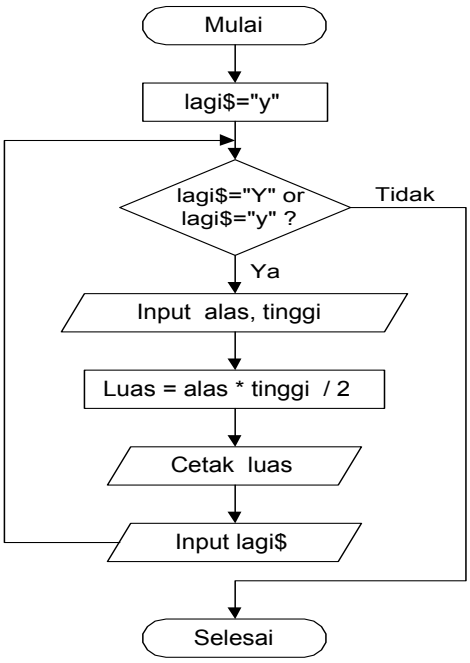
Program 4.4. :

```
REM Program untuk mencetak deret 1 s/d 100
REM menggunakan loop WHILE..WEND
i=0
WHILE i<100
  i=i+1
  print i;
WEND
```

Program 4.5. :

```
REM Program untuk mencetak deret 100 95 …10 5 0
REM menggunakan loop WHILE..WEND
i=100
WHILE i=>0
  Print I;
  i=i-5
WEND
```

Algorithma untuk membuat menjalankan program secara berulang-ulang dengan perintah WHILE..WEND (menggantikan penggunaan perintah GOTO).



Gambar 4.6. Membuat program perulangan dengan WHILE..WEND

Program 4.6. :

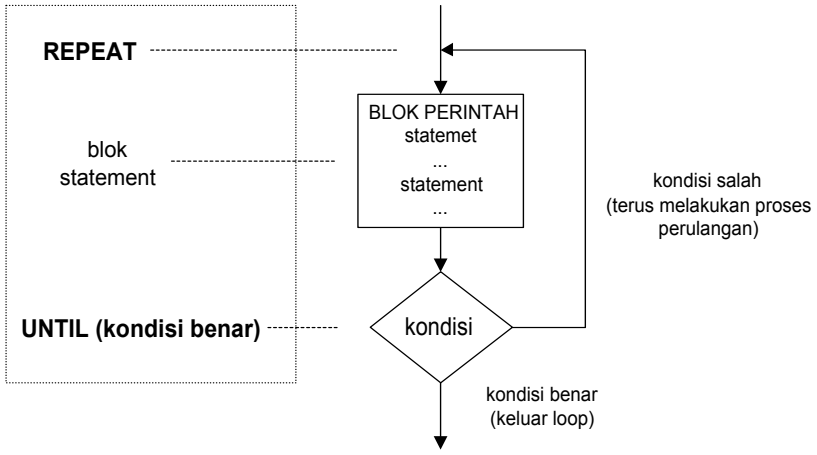
```
REM Program untuk menjalankan program secara berulang
REM menggunakan loop WHILE..WEND
lagi$="Y"
WHILE lagi$="Y" or lagi$="y"
    CLS
    INPUT "Alas : ",alas
    INPUT "Tinggi : ",tinggi
    Luas = alas * tinggi /2
    PRINT "Luas segitiga = ",luas
    INPUT "Mau menghitung lagi ? (Y/N)",lagi$
 WEND
```

Pada program di atas, loop akan terus dikerjakan selama user menekan tombol Y (baik huruf besar maupun kecil). Perintah WHILE..WEND di atas dapat

menggantikan penggunaan perintah GOTO yang telah dijelaskan pada contoh program sebelumnya.

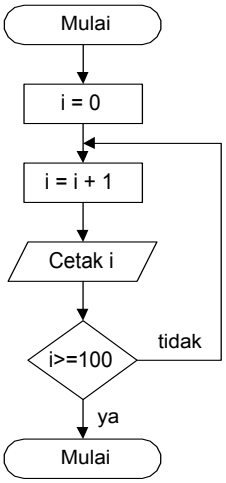
4.3. DO…LOOP UNTIL

Instruksi DO..LOOP UNTIL fungsinya sama dengan perintah WHILE..WEND, hanya saja kalau WHILE ..WEND kondisinya ada di atas (awal loop) sedangkan DO..LOOP UNTIL kondisinya ada di bawah (akhir loop).



Gambar 4.7. Diagram proses perulangan DO..LOOP UNTIL

Algorithma untuk mencetak deret 1 s/d 100 dengan DO..LOOP UNTIL



Gambar 4.8. Mencetak deret 1 s/d 100 dengan DO..LOOP UNTIL

Program 4.7.

```
REM Program untuk mencetak derat 1..100
REM menggunakan DO..LOOP UNTIL
cls
i=0
DO
  i=i+1
  PRINT i;
LOOP UNTIL i>=100
```

Program 4.8.

```
REM Program yg menggunakan perintah DO..LOOP UNTIL
REM untuk menjalankan program secara berulang
lagi$="Y"
DO
    CLS
    INPUT "Alas : ",alas
    INPUT "Tinggi : ",tinggi
    Luas = alas * tinggi /2
    PRINT "Luas segitiga = ",luas
    INPUT "Mau menghitung lagi ? (Y/N)",lagi$
 LOOP UNTIL ucase$(lagi$)<>"Y"
```

Perbedaan WHILE…WEND dengan DO..LOOP UNTIL

- While ..Wend : loop dikerjakan selama kondisi benar, jika kondisi salah maka program akan keluar dari loop

  DO..LOOP UNTIL : loop akan dikerjakan terus sampai kondisi benar. Jadi loop justru dikerjakan selama kondisi salah (false), jika kondisi benar maka program akan keluar dari loop
- While ..Wend : kondisi terletak di atas, sedangkan DO..LOOP kondisi terletak di bawah

- Perhatikan baik-baik 2 program berikut ini :

Menggunakan WHILE..WEND	Menggunakan DO..LOOP UNTIL

A=200
WHILE A < 100
  A= A+1
  ? A
WEND
	A=200
	DO
	  A= A+1
	  ? A
	LOOP UNTIL A>=100

Output di layar: tidak ada	Output di layar: 201

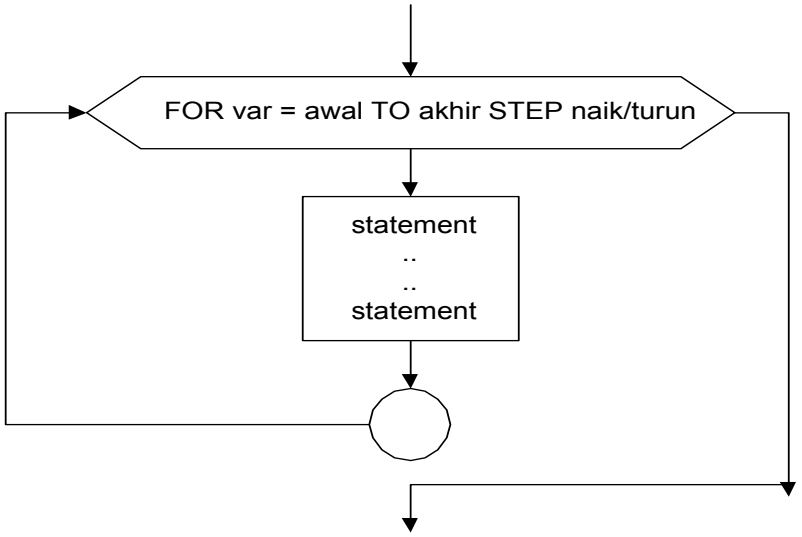
Program sebelah kiri (menggunakan WHILE..WEND) jika dijalankan/di-RUN tidak menghasilkan output di layar. Hal ini disebabkan karena harga awal variabel A adalah 200. Pada saat akan masuk loop, ada pertanyaan apakah A <100 ?, karena A bernilai 200 maka pernyataan bernilai salah (false) sehingga loop WHILE..WEND tidak pernah dijalankan sehingga tidak ada output yang tercetak di layar. (Ingat, loop WHILE..WEND hanya dikerjakan jika kondisi benar)

Pada program sebelah kanan, DO..LOOP UNTIL melakukan pengecekan kondisi di akhir loop, sehingga variabel A sempat masuk ke dalam loop, kemudian nilainya dinaikkan 1 (A=A+1) dan dicetak ke layar. Setelah itu ada pertanyaan : 'Apakah A>= 100', yang mempunyai jawaban 'ya' sehingga keluar dari loop dan program selesai.

4.4. FOR..NEXT

Perintah FOR..NEXT digunakan untuk mengeksekusi suatu baris/blok instruksi secara berulang-ulang selama kondisi terpenuhi. Perintah ini lebih tepat digunakan untuk proses perulangan yang dapat diketahui jumlah perulangannya.
Bentuk umum perintah FOR..NEXT :
FOR variabel=nilai_awal TO nilai_akhir [STEP nilai penambah/pengurang]

Jika nilai penambah/pengurang (STEP) tidak ditulis, maka secara otomatis akan digunakan nilai penambahan 1.



Gambar 4.9. Diagram loop FOR..NEXT

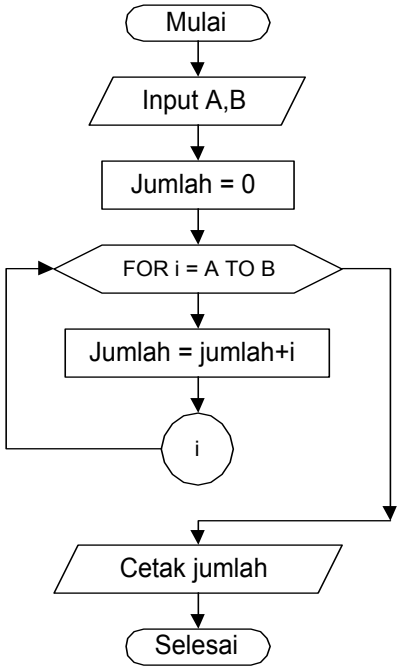
Program 4.9.

```
REM Contoh Program yang menggunakan FOR..NEXT
REM Program untuk mencetak deret 1..100
FOR a= 1 to 100
    Print a;
NEXT
```

Program 4.10.

```
REM Contoh Program yang menggunakan FOR..NEXT
REM Program untuk mencetak deret 100 95 … 10  5  0
FOR a= 100 to 0 step -5
    Print a;
 NEXT
```

Algorithma untuk penjumlahan deret A s/d B



Gambar 4.10. Algorithma penjumlahan deret A s/d B

Program 4.11.

```
REM Contoh Program yang menggunakan FOR..NEXT
REM untuk menghitung jumlah semua bilangan A s/d B
Input "Masukkan nilai A: ",A
Input "Masukkan nilai B: ",B
Jumlah=0
FOR C= A to B
    Jumlah=jumlah+C
NEXT
PRINT "Jumlah deret A s/d B : ";Jumlah
```

Output :

Masukkan nilai A: 10
Masukkan nilai B: 15
Jumlah deret A s/d B : 75

Instruksi FOR..NEXT bersarang

Dalam sebuah loop FOR..NEXT diperbolehkan untuk membuat loop FOR..NEXT lagi yang disebut FOR..NEXT bersarang (nested loop).

Program 4.12.

```
REM Contoh Program yang menggunakan
REM loop FOR..NEXT bersarang
cls
PRINT " A  B"
FOR a= 1 to 3
  FOR b=1 to 3
        PRINT a;b
  NEXT
  PRINT
NEXT
```

Output :

```
1 1
1 2
1 3

2 1
2 2
2 3

3 1
3 2
3 3
```

Program 4.13.

```
REM Contoh Program yang menggunakan
REM loop FOR..NEXT bersarang
cls
input “Masukkan sembarang bilangan :”,bil
FOR a= 1 to bil
  FOR b=1 to a
    ? a;
  NEXT
  PRINT
NEXT
```

Jika dijalankan, maka program di atas akan menghasilkan keluaran sbb.

Masukkan sembarang bilangan : 4
1
2 2
3 3 3
4 4 4 4

Program 4.14.

```
CLS
INPUT "Masukkan sembarang bilangan : ",n
FOR i = 1 TO n
   FOR j = 1 TO i
     print "*";
   NEXT j
   print
NEXT i
FOR i = n-1 TO 1 STEP -1
   FOR j = 1 TO i
     print "#";
   NEXT j
   Print
NEXT i
```

Output :

Masukkan sembarang bilangan : 4
*****
******
****
****
****
******
*****