PART 43

PERULANGAN WHILE DO DALAM PASCAL

KONSEP DASAR PERULANGAN WHILE DO DALAM PASCAL

Perulangan FOR DO dan FOR DOWNTO DO yang saya bahas sebelum ini cocok untuk kondisi dimana kita sudah tahu berapa banyak perulangan yang ingin dijalankan. Dalam FOR DO, nilai awal perulangan dan nilai akhir sudah harus ditulis di awal kode program.

Untuk situasi dimana jumlah perulangan belum bisa dipastikan, kita bisa menggunakan perulangan **WHILE DO** atau **REPEAT UNTIL**. Dalam tutorial kali ini saya akan fokus kepada WHILE DO.

Berikut format dasar penulisan perulangan WHILE DO dalam bahasa pemrograman PASCAL:

```
1  WHILE (condition) DO
2  begin
3    (kode program yang ingin diulang disini...)
4    (kode program untuk mengubah condition..)
5  end;
```

Kunci dari perulangan WHILE DO ada di **condition** dan **kode program untuk mengubah condition**. **Condition** bisa dikatakan sebagai syarat agar perulangan bisa dijalankan. Selama syarat ini terpenuhi (bernilai **TRUE**), perulangan akan terus dijalankan. Jika syarat ini tidak terpenuhi (bernilai **FALSE**), perulangan tidak akan berjalan.

Mari kita lihat contoh kode programnya.

CONTOH KODE PROGRAM PERULANGAN WHILE DO DALAM PASCAL

Agar seragam dengan perulangan FOR DO, saya ingin membuat 10 baris teks "**Hello World**" menggunakan WHILE DO. Berikut kode programnya:

```
1  program while_do;
2  uses crt;
3  var
4    i: integer;
5  begin
6    clrscr;
7    i:= 0;
8
9    while i < 10 do
10     begin
11       writeln('Hello World');
12       i:= i + 1;
13     end;
14
15   readln;
16 end.
```

Pada awal program, saya membuat variabel **i** yang berfungsi sebagai **variabel counter**. Sebelum perulangan, saya memberikan nilai 0 untuk **i**. Setelah itu kita masuk ke perulangan **WHILE DO**.

```
1  program while_do;
2  uses crt;
3  var
4    i: integer;
5  begin
6    clrscr;
7    i:= 0;
8
9    while i < 10 do
10     begin
11       writeln('Hello World');
12       i:= i + 1;
13     end;
14
15   readln;
16 end.
```

Pada awal program, saya membuat variabel **i** yang berfungsi sebagai **variabel counter**. Sebelum perulangan, saya memberikan nilai 0 untuk **i**. Setelah itu kita masuk ke perulangan **WHILE DO**.

Baris program **while i < 10 do** adalah awal dari perulangan. Inilah **kondisi** atau **syarat** yang harus dipenuhi supaya perulangan bisa diproses. Ketika kode program jalan pertama kali, nilai variabel i adalah 0, artinya kondisi **i < 10** menghasilkan nilai **TRUE**. Karena tentu saja 0 kurang dari 10.

Karena syarat di penuhi, blok **begin** hingga **end;** segera di eksekusi. Baris pertama adalah **writeln('Hello World')**. Ini digunakan untuk menampilkan teks 'Hello World'. Tidak ada masalah.

Baris berikutnya saya membuat **i:= i + 1**. Bagian ini dikenal juga sebagai **increment**, artinya saya ingin menambah nilai variabel counter **i** sebanyak 1 angka. Ini dilakukan supaya bisa mengubah kondisi **i < 10** yang terdapat di awal perulangan. Jika ini tidak ditulis, perulangan tidak akan pernah berhenti (**infinity loop**).

Sampai disini, kode program akan kembali ke awal dan mengecek apakah **i < 10**? Ingat, variabel i sekarang sudah bernilai 1. Oke, **1 < 10 = benar** (**TRUE**), kembali jalankan **writeln('Hello World')**, yang diikuti dengan **i:= i + 1**. Karena **1 + 1 = 2**, varibel i sekarang bernilai 2.

Kode program kembali ke awal dan mengecek apakah **i < 10**? Sekarang nilai i adalah 2, dan 2 masih kurang dari 10, **2 < 10 = benar** (**TRUE**), sekali lagi kode blok perulangan akan dijalankan.

Proses seperti ini terus berlangsung sampai kondisi **i < 10** menghasilkan **FALSE**. Kapan kondisi ini terjadi? Yakni ketika variabel **i = 10**. **10 < 10** adalah **FALSE**. Artinya, perulangan WHILE DO akan dijalankan sebanyak 10 kali, dimana dalam setiap perulangan, nilai i akan menaik mulai dari 0, 1, 2, 3, 4, 5, 6, 7, 8, hingga 9.

Agar konsep perulangan ini bisa lebih paham, silahkan anda bayangkan (kalau perlu ditulis), bagaimana nilai variabel counter i bisa naik dari 1 sampai 9, dan pada setiap kenaikan perulangan akan menampilkan teks **'Hello World'**.

Jika kode program diatas masih kurang paham, mari masuk ke contoh kedua:

```
1  program while_do;
2  uses crt;
3  var
4    i: integer;
5  begin
6    clrscr;
7    i:= 0;
8
9    while i < 10 do
10     begin
11       writeln('Variabel i sekarang bernilai: ',i);
12       i:= i + 1;
13     end;
14   readln;
15 end.
```

Kondisi perulangan **WHILE DO** yang saya pakai sama persis seperti sebelumnya. Hanya saja kali ini perintah yang dijalankan pada setiap perulangan adalah **writeln('Variabel i sekarang bernilai: ',i)**. Ini akan membantu kita melihat nilai variabel counter i yang terus bertambah 1 selama perulangan dijalankan.

PEMAHAMAN LOGIKA UNTUK PERULANGAN WHILE DO

Untuk bisa membuat perulangan dengan WHILE DO, kita perlu pemahaman logika. Logika diperlukan untuk menentukan **nilai awal** dan **kondisi akhir**.

Sebagai latihan soal, saya ingin anda membuat kode program yang menghasilkan teks: *"Variabel i sekarang bernilai: 5"*, *"Variabel i sekarang bernilai: 6"*, ... Hingga *"Variabel i sekarang bernilai: 10"*. Yup, hanya 5 baris. Silahkan anda modifikasi kode program diatas.

```
1  program while_do;
2  uses crt;
3  var
4    i: integer;
5  begin
6    clrscr;
7    i:= 5;
8
9    while i <= 10 do
10     begin
11       writeln('Variabel i sekarang bernilai: ',i);
12       i:= i + 1;
13     end;
14   readln;
15 end.
```

Kuncinya adalah bagaimana menentukan **kondisi awal** variabel i, dan **kondisi akhir** dari perulangan. Dengan membuat **i:= 5**, dan **while i <= 10 do**, artinya nilai i akan mulai dari 5, 6, 7, 8, 9, dan 10.

Tapi saya juga bisa menggunakan kode program berikut:

```
1  program while_do;
2  uses crt;
3  var
4    i: integer;
5  begin
6    clrscr;
7    i:= 5;
8
9    while i < 11 do
10     begin
11       writeln('Variabel i sekarang bernilai: ',i);
12       i:= i + 1;
13     end;
14   readln;
15 end.
```

Dapatkah anda melihat bedanya? Perhatikan bahwa kali ini saya menggunakan kondisi **while i < 11 do.** Sebenarnya ini sama seperti **while i <= 10 do**. Angka 10 akan TRUE jika dibandingkan dengan "**< 11**", maupun "**<= 10**".

Konsep seperti ini agar selalu diperhatikan ketika membuat perulangan WHILE DO.

Hati-hati dengan Infinity Loop

Infinity Loop adalah sebuah perulangan yang tidak pernah berhenti. Ini terjadi karena variabel kondisi akan selalu bernilai TRUE.

Perhatikan kode program berikut:

```
1  program while_do;
2  uses crt;
3  var
4    i: integer;
5  begin
6    clrscr;
7    i:= 5;
8
9    while i < 11 do
10     begin
11       writeln('Variabel i sekarang bernilai: ',i);
12     end;
13   readln;
14 end.
```

Jika anda menjalankan kode program diatas, akan terjadi **Infinity Loop**. Ini karena kondisi while **i <= 10 do** akan selalu **TRUE**. Di dalam perulangan saya tidak membuat 'sesuatu' yang bisa mengubah nilai variabel i (untuk membuat syarat **i <= 10** menjadi **FALSE**).

Infinity Loop biasanya terjadi karena kesalahan logika dari programmer, terutama untuk perulangan WHILE DO dan REPEAT UNTIL. Untuk menghentikan infinity loop, anda bisa menekan kombinasi tombol CRTL + C, atau menutup paksa Free Pascal.

MEMBUAT HITUNG MUNDUR DENGAN WHILE DO

Untuk membuat perulangan yang mundur, kita bisa mengubah nilai awal **variabel counter** dan **kondisi syarat** dari perulangan WHILE DO. Berikut contohnya:

```
1  program while_do;
2  uses crt;
3  var
4    i: integer;
5  begin
6    clrscr;
7    i:= 100;
8
9    while i >= 0 do
10     begin
11       writeln('Hitung mundur: ',i);
12       i:= i - 1;
13     end;
14   readln;
15 end.
```

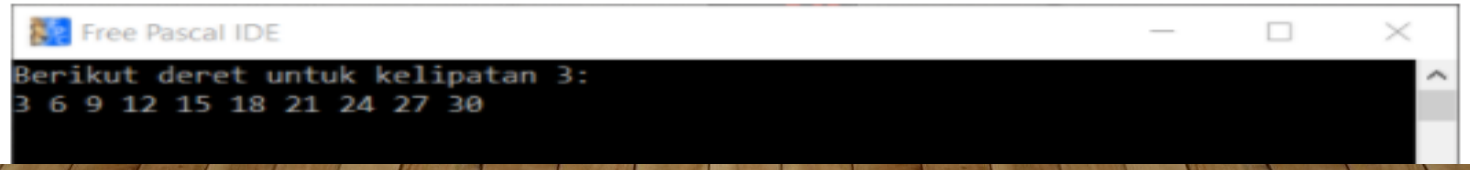
MEMBUAT PERULANGAN LOMPAT DENGAN WHILE DO

Dengan memodifikasi bagian **counter**, kita bisa membuat perulangan yang "lompat", yakni tidak berurutan dari 1, 2, 3, dst. Kita bisa membuat variabel counter yang naik misalnya dari 3, 6, 9, 12, dst.

Perhatikan kode program berikut, dan dapatkah anda menebak urutan angka yang ditampilkan?

```
1  program while_do;
2  uses crt;
3  var
4    i: integer;
5  begin
6    clrscr;
7    i:= 3;
8
9    writeln('Berikut deret untuk kelipatan 3: ');
10
11   while i <= 30 do
12     begin
13       write(i,' ');
14       i:= i + 3;
15     end;
16   readln;
17 end.
```

Perhatikan bahwa variabel **i** mulai dari **3**, dan perubahannya menggunakan **i:= i + 3**. Artinya variabel i akan 'lompat' 3 angka setiap perulangan. Berikut hasilnya:



Sampai disini saya sarankan anda mencoba2 berbagai kombinasi perulangan lain, misalnya membuat perulangan yang lompat setiap 7 angka, atau perulangan yang mundur 4 angka, misalnya dari 100, ke 96, ke 92, ke 88, dst.