

REPETITION CONTROL STRUCTURE

Pertemuan 5

Objectives

- To develop algorithms that use the DOWHILE and REPEAT...UNTIL control structures
- To introduce a pseudocode structure for counted repetition loops
- To develop algorithms using variations of the repetition construct

Outline

- 5.1 Repetition using the DOWHILE structure
- 5.2 Repetition using the REPEAT...UNTIL structure
- 5.3 Counted repetition

Algoritma solusi yang dikembangkan sejauh ini memiliki satu karakteristik yang sama:

mereka menunjukkan logika program yang diperlukan untuk memproses hanya satu set nilai input. Namun, sebagian besar program memerlukan logika yang sama untuk diulang untuk beberapa set data. Cara paling efisien untuk mengatasi situasi ini adalah dengan membuat struktur perulangan dalam algoritma yang akan menyebabkan logika pemrosesan diulang beberapa kali.

Ada tiga cara berbeda di mana satu set instruksi dapat diulang, dan setiap cara ditentukan oleh di mana keputusan untuk mengulang ditempatkan:

- di awal loop (**leading decision loop**)
- di akhir loop (**trailing decision loop**)
- beberapa kali dihitung (**counted loop**).

Leading decision loop

konstruksi **DOWHILE** diperkenalkan sebagai representasi pseudocode dari loop berulang. Formatnya adalah:

```
DOWHILE condition p is true  
    statement block  
ENDDO
```

Konstruksi **DOWHILE** adalah loop keputusan terdepan – kondisi diuji sebelum pernyataan apa pun dieksekusi. Pada loop **DOWHILE** di atas, pemrosesan berikut terjadi:

- 1 Kondisi logis p diuji.
- 2 Jika kondisi p ditemukan benar, pernyataan dalam blok pernyataan dieksekusi satu kali. Pembatas **ENDDO** kemudian memicu kembalinya kontrol ke pengujian ulang kondisi p.
- 3 Jika kondisi p masih benar, pernyataan dieksekusi lagi, dan proses pengulangan berlanjut sampai kondisi ditemukan salah.
- 4 Jika kondisi p ditemukan salah, kontrol diteruskan ke pernyataan berikutnya setelah pembatas **ENDDO** dan tidak ada pemrosesan lebih lanjut yang terjadi di dalam loop

Ada dua pertimbangan penting yang harus Anda perhatikan sebelum merancang loop **DOWHILE**:

- Pengujian kondisi berada di awal loop. Ini berarti bahwa mungkin perlu untuk melakukan beberapa pemrosesan awal untuk mengatur kondisi secara memadai sebelum dapat diuji.
- Satu-satunya cara untuk mengakhiri perulangan adalah dengan membuat kondisi **DOWHILE** salah. Ini berarti beberapa proses harus diatur dalam blok pernyataan yang pada akhirnya akan mengubah kondisi sehingga kondisi menjadi salah. Kegagalan untuk melakukan ini menghasilkan loop tanpa akhir

Using DOWHILE to repeat a set of instructions a known number of times

Ketika satu set instruksi harus diulang beberapa kali, counter dapat digunakan dalam pseudocode, yang diinisialisasi sebelum pernyataan **DOWHILE** dan bertambah tepat sebelum pernyataan **ENDDO**. Mari kita lihat sebuah contoh.

Fahrenheit–Celsius conversion

Setiap hari, stasiun cuaca menerima 15 suhu yang dinyatakan dalam derajat Fahrenheit. Sebuah program akan ditulis yang akan menerima setiap suhu Fahrenheit, mengubahnya menjadi Celcius dan menampilkan suhu yang dikonversi ke layar. Setelah 15 suhu diproses, kata-kata 'Semua suhu diproses' akan ditampilkan di layar.

A Defining diagram

Input	Processing	Output
f_temp (15 temperatures)	Get Fahrenheit temperatures Convert temperatures Display Celsius temperatures Display screen messag	c_temp (15 temperatures)

Setelah menentukan input, output, dan pemrosesan, Anda siap untuk menguraikan solusi untuk masalah tersebut. Ini dapat dilakukan dengan menuliskan struktur kontrol yang diperlukan dan variabel tambahan apa pun yang akan digunakan dalam algoritma solusi. Dalam contoh ini, Anda membutuhkan:

- Struktur **DOWHILE** untuk mengulang proses yang diperlukan
- Penghitung, yang disebut hitungan suhu, diinisialisasi ke nol, yang akan mengontrol 15 pengulangan

B Solution algorithm

```
Fahrenheit_Celsius_conversion
1  Set temperature_count to zero
2  DOWHILE temperature_count < 15
3    Prompt operator for f_temp
4    Get f_temp
5    compute c_temp = (f_temp – 32) * 5/9
6    Display c_temp
7    add 1 to temperature_count
8  ENDDO
END
```

Algoritma solusi ini menggambarkan sejumlah poin:

1 Variabel temperature_count diinisialisasi sebelum kondisi **DOWHILE** dijalankan.

2 Selama temperature_count kurang dari 15 (yaitu, **DOWHILE** kondisi benar), pernyataan antara **DOWHILE** dan **ENDDO** akan dieksekusi.

3 Variabel temperature_count bertambah satu kali dalam loop, tepat sebelum pembatas **ENDDO** (yaitu, tepat sebelum diuji lagi dalam kondisi **DOWHILE**).

4 Setelah 15 iterasi, temperature_count akan sama dengan 15, yang menyebabkan kondisi **DOWHILE** menjadi salah dan kontrol diteruskan ke pernyataan setelah **ENDDO**.

C Desk checking

Meskipun program akan membutuhkan 15 record untuk diproses dengan benar, pada tahap ini hanya perlu memeriksa algoritma dengan dua set data yang valid.

1 Input data

2 Expected results

	First data set	Second data set
f_temp	32	50

	First data set	Second data set
c_temp	0	10

3 Desk check table

Statement number	temperature_count	DOWHILE condition	f_temp	c_temp
1	0			
2		true		
3, 4			32	
5				0
6				display
7	1			
2		true		
3, 4			50	
5				10
6				display
7	2			

Repetition using the REPEAT...UNTIL structure

Struktur **REPEAT...UNTIL** mirip dengan struktur **DOWHILE**, di mana sekelompok pernyataan diulang sesuai dengan kondisi yang ditentukan. Namun, di mana struktur **DOWHILE** menguji kondisi di awal loop, struktur **REPEAT...UNTIL** menguji kondisi di akhir loop. Ini berarti bahwa pernyataan dalam loop akan dieksekusi sekali sebelum kondisi diuji. Jika kondisi salah, pernyataan akan diulang SAMPAI kondisi menjadi benar.

Format dari struktur **REPEAT...UNTIL** adalah:

REPEAT

statement

statement

.

.

UNTIL condition is true

REPEAT...UNTIL adalah keputusan yang mengikuti loop; pernyataan dieksekusi sekali sebelum kondisi diuji. Ada dua pertimbangan yang perlu Anda perhatikan sebelum menggunakan **REPEAT...UNTIL**.

Pertama, pengulangan REPEAT...UNTIL dieksekusi ketika kondisinya salah; hanya ketika kondisinya menjadi benar, pengulangan berhenti. Jadi, logika klausa kondisi dari struktur REPEAT...UNTIL adalah kebalikan dari DO..WHILE. Misalnya, 'DOWHILE more record' sama dengan 'REPEAT... SAMPAI tidak ada record lagi', dan 'DOWHILE number NOT = 99' sama dengan 'REPEAT...UNTIL number = 99'.

Kedua, pernyataan dalam struktur REPEAT...UNTIL akan selalu dieksekusi setidaknya sekali. Akibatnya, tidak perlu membaca priming saat menggunakan REPEAT...UNTIL. Satu pernyataan Baca di awal loop sudah cukup; namun, pernyataan IF tambahan segera setelah pernyataan Baca harus disertakan, untuk mencegah pemrosesan rekaman cuplikan. Sekarang mari kita bandingkan algoritma yang menggunakan struktur DO..WHILE dengan masalah yang sama menggunakan struktur REPEAT...UNTIL.

Perhatikan loop DO..WHILE berikut:

```
Process_student_records
  Set student_count to zero
  Read student record
  DOWHILE student_number NOT = 999
    Write student record
    increment student_count
    Read student record
  ENDDO
  Print student_count
END
```

Ini dapat ditulis ulang sebagai loop keputusan tambahan, menggunakan REPEAT...UNTIL struktur sebagai berikut:

```
Process_student_records
  Set student_count to zero
  REPEAT
    Read student record
    IF student number NOT = 999 THEN
      Write student record
      increment student_count
    ENDIF
  UNTIL student number = 999
  Print student_count
END
```

Sebuah program diperlukan untuk membaca serangkaian catatan inventaris yang berisi nomor item, deskripsi item, dan gambar stok. Catatan terakhir dalam file memiliki nomor item nol. Program ini menghasilkan laporan stok barang rendah, dengan hanya mencetak catatan yang memiliki angka stok kurang dari 20 barang. Judul akan dicetak di bagian atas laporan dan jumlah total stok barang yang sedikit dicetak di bagian akhir.

A Defining diagram

input	Processing	Output
inventory record <ul style="list-style-type: none">• item_number• item_description• stock_figure	Read inventory records Select low stock items Print low stock records Print total low stock items	heading selected records <ul style="list-style-type: none">• item_number• item_description• stock_figure total_low_stock_items

Anda perlu mempertimbangkan persyaratan berikut saat membuat algoritma solusi:

- sebuah REPEAT...UNTIL untuk melakukan pengulangan
- pernyataan IF untuk memilih angka stok kurang dari 20
- akumulator untuk total_low_stock_items
- IF ekstra, dalam loop REPEAT, untuk memastikan inventory record tidak diproses.

B Solution algorithm using REPEAT...UNTIL

```
Process_inventory_records
1   Set total_low_stock_items to zero
2   Print 'Low Stock Items' heading
   REPEAT
3     Read inventory record
4     IF item_number > zero THEN
       IF stock_figure < 20 THEN
         Print item_number, item_description, stock_figure
         increment total_low_stock_items
       ENDIF
     ENDIF
5   UNTIL item_number = zero
6   Print total_low_stock_items
END
```

Algoritma solusi memiliki struktur sederhana, dengan satu pernyataan Baca di awal loop REPEAT...UNTIL dan pernyataan IF tambahan di dalam loop untuk memastikan rekaman trailer tidak salah ditambahkan ke akumulator total_low_stock_items.

C Desk checking

total_low_stock_items (nomor item sama dengan nol) akan digunakan untuk menguji algoritma:

1 Input data

	First record	Second record	Third record
item_number	123	124	0
stock_figure	8	25	

2 Expected results

Low Stock Items

123 8 (first record)

Total Low Stock Items = 1

3 Desk check table

Statement number	item_number	stock_figure	REPEAT UNTIL	stock_total_low item_s	heading
1				0	
2					print
3	123	8			
4	print	print		1	
5			false		
3	124	25			
4					
5			false		
3	0				
4					
5			true		
6				print	

Counted repetition

Counted loop

Pengulangan terhitung terjadi ketika jumlah yang tepat dari iterasi loop diketahui sebelumnya. Eksekusi loop dikendalikan oleh indeks loop, dan alih-alih menggunakan DOWHILE, atau REPEAT...UNTIL, kata kunci sederhana DO digunakan sebagai berikut:

```
DO loop_index = initial_value to final_value  
  statement block  
ENDDO
```

Loop DO melakukan lebih dari sekedar mengulang blok pernyataan. Itu akan:

- 1 inialisasi loop_index ke initial_value yang diperlukan
- 2 tingkatkan loop_index dengan 1 untuk setiap melewati loop
- 3 uji nilai loop_index di awal setiap loop untuk memastikan bahwa itu berada dalam kisaran nilai yang dinyatakan
- 4 mengakhiri loop ketika loop_index telah melebihi nilai akhir yang ditentukan.

Dengan kata lain, konstruk pengulangan terhitung akan melakukan inialisasi, penambahan, dan pengujian penghitung loop secara otomatis. Ini juga akan menghentikan loop setelah jumlah pengulangan yang diperlukan telah dieksekusi.

Mari kita lihat lagi Contoh sebelumnya, yang memproses 15 suhu di stasiun cuaca setiap hari. Algoritma solusi dapat ditulis ulang untuk menggunakan loop DO.

Fahrenheit–Celsius conversion

Setiap hari, stasiun cuaca menerima 15 suhu yang dinyatakan dalam derajat Fahrenheit. Sebuah program akan ditulis yang akan menerima setiap suhu Fahrenheit, mengubahnya menjadi Celsius dan menampilkan suhu yang dikonversi ke layar. Setelah 15 suhu diproses, kata-kata 'Semua suhu diproses' akan ditampilkan di layar.

A Defining diagram

Input	Processing	Output
f_temp (15 temperatures)	Get Fahrenheit temperatures Convert temperatures Display Celsius temperatures Display screen message	c_temp (15 temperatures)

Setelah menentukan input, output, dan pemrosesan, Anda siap untuk menguraikan solusi untuk masalah tersebut. Ini dapat dilakukan dengan menuliskan struktur kontrol yang diperlukan dan variabel tambahan apa pun yang akan digunakan dalam algoritma solusi. Dalam contoh ini, Anda membutuhkan:

- struktur DOWHILE untuk mengulang proses yang diperlukan
- penghitung, disebut temperature_count, diinisialisasi ke nol, yang akan mengontrol 15 pengulangan.

B Solution algorithm

Solusinya akan membutuhkan loop DO dan penghitung loop (temperature_count) untuk memproses pengulangan.

```
Fahrenheit_Celsius_conversion
1  DO temperature_count = 1 to 15
2    Prompt operator for f_temp
3    Get f_temp
4    compute c_temp = (f_temp - 32) * 5/9
5    Display c_temp
   ENDDO
6  Display 'All temperatures processed' to the screen
END
```

Perhatikan bahwa loop DO mengontrol semua pengulangan:

- Menginisialisasi temperature_count ke 1.
- Menambah temperature_count sebesar 1 untuk setiap melewati loop.
- Menguji temperature_count di awal setiap pass untuk memastikan bahwa itu berada dalam kisaran 1 s/d 15.
- Secara otomatis mengakhiri loop setelah temperature_count melebihi 15.

C Desk checking

Dua catatan yang valid harus cukup untuk menguji kebenaran algoritma. Tidak perlu memeriksa konstruksi loop DO untuk semua 15 record.

1 Input data

	First data set	Second data set
f_temp	32	50

2 Expected results

	First data set	Second data set
c_temp	0	10

3 Desk check table

Statement number	temperature_count	f_temp	c_temp
1	1		
2, 3		32	
4			0
5			display
1	2		
2, 3		50	
4			10
5			display

Desk checking algoritma dengan dua kasus uji input menunjukkan bahwa hasil yang diharapkan telah tercapai.