

SELECTION CONTROL STRUCTURES

Pertemuan 4

Objectives

- To elaborate on the uses of simple selection, multiple selection and nested selection in algorithms
- To introduce the case construct in pseudocode
- To develop algorithms using variations of the selection control structure

Outline

- 4.1 The selection control structure
- 4.2 Algorithms using selection
- 4.3 The case structure

The selection control structure

Struktur kontrol seleksi diperkenalkan pada Bab 2 sebagai konstruksi kedua dalam Teorema Struktur. Struktur ini mewakili kemampuan pengambilan keputusan dari komputer. Artinya, Anda dapat menggunakan struktur kontrol pemilihan dalam pseudocode untuk mengilustrasikan pilihan antara dua atau lebih tindakan, tergantung pada apakah suatu kondisi benar atau salah. Kondisi dalam pernyataan IF didasarkan pada perbandingan dua item, dan biasanya dinyatakan dengan salah satu operator relasional berikut:

| | |
|----|--------------------------|
| < | less than |
| > | greater than |
| = | equal to |
| <= | less than or equal to |
| >= | greater than or equal to |
| <> | not equal to |

Ada beberapa variasi struktur seleksi, sebagai berikut:

1 Simple selection (simple IF statement)

Seleksi sederhana terjadi ketika pilihan dibuat antara dua jalur alternatif, tergantung pada hasil dari suatu kondisi yang benar atau salah. Struktur direpresentasikan dalam pseudocode menggunakan kata kunci **IF**, **THEN**, **ELSE** dan **ENDIF**. Sebagai contoh:

```
IF account_balance < $300 THEN
    service_charge = $5.00
ELSE
    service_charge = $2.00
ENDIF
```

Hanya satu dari jalur **THEN** atau **ELSE** yang akan diikuti, tergantung pada hasil kondisi dalam klausa **IF**.

2 Simple selection with null false branch (null ELSE statement)

Struktur **ELSE** null adalah variasi dari struktur **IF** sederhana. Ini digunakan ketika tugas dilakukan hanya ketika kondisi tertentu benar. Jika kondisinya salah, maka tidak ada pemrosesan yang terjadi dan pernyataan **IF** akan dilewati. Sebagai contoh:

```
IF student_attendance = part_time THEN  
    add 1 to part_time_count  
ENDIF
```

Dalam hal ini, kolom `part_time_count` akan diubah hanya jika `student_attendance` adalah `part_time`.

3 Combined selection (combined IF statement)

Pernyataan **IF** gabungan adalah pernyataan yang berisi beberapa kondisi, masing-masing terhubung dengan operator logika **AND** atau **OR**. Jika konektor **AND** digunakan untuk menggabungkan kondisi maka kedua kondisi harus benar agar kondisi gabungan menjadi benar. Sebagai contoh:

```
IF student_attendance = part_time  
AND student_gender = female THEN  
    add 1 to female_part_time_count  
ENDIF
```

Dalam hal ini, setiap catatan siswa akan menjalani dua tes. Hanya siswa perempuan dan yang kehadirannya terdaftar sebagai paruh waktu yang akan dipilih, dan variabel `female_part_time_count` akan bertambah. Jika salah satu kondisi ditemukan salah, penghitung akan tetap tidak berubah.

Jika konektor **OR** digunakan untuk menggabungkan dua kondisi apa pun, maka hanya satu kondisi yang harus benar agar kondisi gabungan dianggap benar. Jika tidak ada kondisi yang benar, kondisi gabungan dianggap salah. Mengubah **AND** dalam contoh di atas menjadi **OR** secara dramatis mengubah hasil dari pemrosesan pernyataan **IF**.

```
IF student_attendance = part_time  
OR student_gender = female THEN  
    add 1 to female_part_time_count  
ENDIF
```

Dalam contoh ini, jika salah satu atau kedua kondisi ditemukan benar, kondisi gabungan akan dianggap benar.

Artinya, penghitung akan bertambah:

1. jika siswa paruh waktu, tanpa memandang jenis kelamin, atau
- 2 jika siswa perempuan, terlepas dari pola kehadiran.

Hanya siswa yang bukan perempuan dan bukan paruh waktu yang akan diabaikan. Jadi, `female_part_time_count` akan berisi jumlah total siswa paruh waktu perempuan, siswa paruh waktu laki-laki dan siswa penuh waktu perempuan. Akibatnya, `female_part_time_count` bukan lagi nama yang berarti untuk variabel ini. Anda harus sepenuhnya memahami pemrosesan yang terjadi saat menggabungkan kondisi dengan operator logika **AND** atau **OR**.

Lebih dari dua kondisi dapat dihubungkan bersama dengan operator AND atau OR. Namun, jika kedua operator digunakan dalam satu pernyataan **IF**, tanda kurung harus digunakan untuk menghindari ambiguitas. Lihatlah contoh berikut:

```
IF record_code = '23'  
OR update_code = delete  
AND account_balance = zero THEN  
    delete customer record  
ENDIF
```

Pernyataan **IF** sekarang tidak lagi ambigu, dan jelas kondisi apa yang diperlukan untuk menghapus customer record . Catatan hanya akan dihapus jika saldo akun sama dengan nol dan `customer record = 23` atau `update_code = hapus`.

4 Nested selection (nested IF statement)

Pemilihan bersarang terjadi ketika kata **IF** muncul lebih dari sekali dalam pernyataan **IF**. Pernyataan **IF** bersarang dapat diklasifikasikan sebagai linier atau non-linier.

Linear nested IF statements

Pernyataan **IF** bersarang linier digunakan ketika bidang sedang diuji untuk berbagai nilai dan tindakan yang berbeda akan diambil untuk setiap nilai.

Bentuk **IF** bersarang ini disebut linier, karena setiap **ELSE** segera mengikuti kondisi **IF** yang sesuai dengannya. Perbandingan dilakukan sampai kondisi benar ditemukan, dan tindakan yang ditentukan dijalankan sampai pernyataan **ELSE** berikutnya tercapai. Pernyataan **IF** bersarang linier harus diindentasi agar mudah dibaca, dengan setiap **IF**, **ELSE**, dan **ENDIF** yang sesuai disejajarkan.

Sebagai contoh:

```
IF record_code = 'A' THEN
    Increment counter_A
ELSE
    IF record_code = 'B' THEN
        increment counter_B
    ELSE
        IF record_code = 'C' THEN
            increment counter_C
        ELSE
            increment error_counter
        ENDIF
    ENDIF
ENDIF
```

Perhatikan bahwa ada jumlah pernyataan **IF**, **ELSE** dan **ENDIF** yang sama, bahwa setiap pernyataan **ELSE** dan **ENDIF** diposisikan sedemikian rupa sehingga sesuai dengan pernyataan **IF** yang cocok, dan lekukan yang benar membuatnya mudah dibaca dan dipahami. Blok pernyataan **IF** bersarang seperti ini kadang-kadang disebut sebagai 'pernyataan **IF** berjenjang', karena mereka mengalir seperti air terjun.

Non-linear nested IF statements

Pernyataan **IF** bersarang non-linear terjadi ketika sejumlah kondisi berbeda harus dipenuhi sebelum tindakan tertentu dapat terjadi.

Disebut nonlinier karena pernyataan **ELSE** dapat dipisahkan dari pernyataan **IF** yang dipasangkan. Indentasi sekali lagi penting ketika mengekspresikan bentuk seleksi ini dalam pseudocode. Setiap pernyataan **ELSE** dan **ENDIF** harus disejajarkan dengan kondisi **IF** yang sesuai dengannya.

Contoh:

```
IF student_attendance = part_time THEN
  IF student_gender = female THEN
    IF student_age > 21 THEN
      add 1 to mature_female_pt_students
    ELSE
      add 1 to young_female_pt_students
    ENDIF
  ELSE
    add 1 to male_pt_students
  ENDIF
ELSE
  add 1 to full_time_students
ENDIF
```

Perhatikan bahwa jumlah kondisi **IF** sama dengan jumlah pernyataan **ELSE** dan **ENDIF**. Menggunakan lekukan yang benar membantu untuk melihat kumpulan pernyataan **IF**, **ELSE**, dan **ENDIF** mana yang cocok. Namun, pernyataan **IF** bersarang non-linier mungkin mengandung kesalahan logika yang sulit untuk diperbaiki, sehingga harus digunakan dengan hemat dalam pseudocode.

Jika memungkinkan, ganti serangkaian pernyataan **IF** bersarang non-linier dengan pernyataan **IF** gabungan. Penggantian ini dimungkinkan dalam pseudocode karena dua pernyataan **IF** berurutan bertindak seperti pernyataan **IF** gabungan yang menggunakan operator **AND**. Ambil contoh pernyataan **IF** bersarang nonlinier berikut:

```
IF student_attendance = part_time THEN
  IF student_age > 21 THEN
    increment mature_pt_student
  ENDIF
ENDIF
```

Ini dapat ditulis sebagai pernyataan **IF** gabungan:

```
IF student_attendance = part_time
AND student_age > 21 THEN
  increment mature_pt_student
ENDIF
```

Hasilnya akan sama untuk kedua ekspresi pseudocode, tetapi format yang terakhir lebih disukai, jika logika memungkinkan, hanya karena lebih mudah dipahami.

Algorithms using selection

Mari kita lihat beberapa contoh pemrograman yang menggunakan struktur kontrol seleksi. Dalam setiap contoh, masalah akan didefinisikan, algoritma solusi akan dikembangkan dan algoritma akan diuji secara manual. Untuk membantu mendefinisikan masalah, kata kerja pemrosesan dalam setiap contoh telah digarisbawahi.

Contoh, rancang algoritma yang akan meminta operator tiga karakter, menerima karakter tersebut sebagai input, mengurutkannya ke dalam urutan menaik dan menampilkannya ke layar

A Defining diagram

| Input | Processing | Output |
|----------------------------|--|----------------------------|
| char_1 char_2 char_3 | Prompt for characters Accept three characters Sort three characters Output three characters | char_1 char_2 char_3 |

B Solution algorithm

Algoritma solusi membutuhkan serangkaian pernyataan IF untuk mengurutkan tiga karakter ke dalam urutan menaik.

```
Read_three_characters
1  Prompt the operator for char_1, char_2, char_3
2  Get char_1, char_2, char_3
3  IF char_1 > char_2 THEN
    temp = char_1
    char_1 = char_2
    char_2 = temp
  ENDIF
4  IF char_2 > char_3 THEN
    temp = char_2
    char_2 = char_3
    char_3 = temp
  ENDIF
5  IF char_1 > char_2 THEN
    temp = char_1
    char_1 = char_2
    char_2 = temp
  ENDIF
6  Output to the screen char_1, char_2, char_3
END
```

Dalam solusi ini, sebagian besar logika algoritma berkaitan dengan pengurutan tiga karakter ke dalam urutan menaik.

Penyortiran ini dilakukan tetapi dengan menggunakan pseudocode yang 'menukar' dua item, sebagai berikut:

```
temp = char_1  
char_1 = char_2  
char_2 = temp
```

Di sini, nilai dalam variabel char_1 dan char_2 'ditukar', dengan menggunakan variabel sementara, temp. Pseudocode seperti ini harus ditulis dengan hati-hati untuk memastikan item tidak hilang ditukar.

C Desk checking

Dua set karakter yang valid akan digunakan untuk memeriksa algoritma; karakter k, b dan g sebagai himpunan pertama dan z, s dan a sebagai himpunan kedua.

1 Input data

| | First data set | Second data set |
|--------|----------------|-----------------|
| har_1 | k | z |
| char_2 | b | s |
| char_3 | g | a |

2 Expected results

| | First data set | Second data set |
|--------|----------------|-----------------|
| char_1 | b | a |
| char_2 | g | s |
| char_3 | k | z |

3 Desk check table

Nomor baris telah digunakan untuk mengidentifikasi setiap pernyataan dalam program. Perhatikan bahwa saat desk memeriksa logika, setiap pernyataan **IF** diperlakukan sebagai satu pernyataan.

| Statement number | char_1 | char_2 | char_3 | temp |
|------------------|--------|--------|--------|------|
| First pass | | | | |
| 1, 2 | k | b | g | |
| 3 | b | k | | k |
| 4 | | | g | k |
| 5 | | | | |
| 6 | output | output | output | |
| Second pass | | | | |
| 1, 2 | z | s | a | |
| 3 | s | z | | z |
| 4 | | a | z | z |
| 5 | a | s | | s |
| 6 | output | output | output | |

The case structure

Struktur case control dalam pseudocode adalah cara lain untuk mengekspresikan pernyataan **IF** bersarang linier. Ini digunakan dalam pseudocode karena dua alasan: dapat diterjemahkan ke dalam banyak bahasa tingkat tinggi, dan membuat pseudocode lebih mudah untuk ditulis dan dipahami. **IF** bersarang sering terlihat rumit dalam pseudocode dan bergantung pada struktur dan lekukan yang benar agar mudah dibaca. Mari kita lihat contoh yang digunakan sebelumnya dalam bab ini:

```
IF record_code = 'A' THEN
    increment counter_A
ELSE
    IF record_code = 'B' THEN
        increment counter_B
    ELSE
        IF record_code = 'C' THEN
            increment counter_C
        ELSE
            increment error_counter
        ENDIF
    ENDIF
ENDIF
```

Struktur **IF** bersarang linier ini dapat diganti dengan struktur case control. Kasus sebenarnya bukan struktur kontrol tambahan. Ini menyederhanakan struktur kontrol pemilihan dasar dan memperluasnya dari pilihan antara dua nilai ke pilihan dari beberapa nilai. Dalam satu struktur kasus, beberapa jalur logika alternatif dapat direpresentasikan. Dalam pseudocode, kata kunci **CASE OF** dan **ENDCASE** berfungsi untuk mengidentifikasi struktur, dengan beberapa nilai yang diindentasi, sebagai berikut:

CASE OF single variable

value_1 : statement block_1

value_2 : statement block_2

.

.

value_n : statement block_n

value_other : statement block_other

ENDCASE

Jalur yang diikuti dalam struktur kasus tergantung pada nilai variabel yang ditentukan dalam klausa **CASE OF**. Jika variabel berisi nilai_1, pernyataan blok_1 dijalankan; jika berisi nilai_2, blok pernyataan_2 dijalankan, dan seterusnya. Nilai_lainnya disertakan jika variabel tidak berisi nilai yang tercantum.

Kita sekarang dapat menulis ulang pernyataan **IF** bersarang linier di atas dengan pernyataan kasus, sebagai berikut:

```
CASE OF record_code
```

```
    'A' : increment counter_A
```

```
    'B' : increment counter_B
```

```
    'C' : increment counter_C
```

```
    other : increment error_counter
```

```
ENDCASE
```

Dalam kedua bentuk pseudocode, logika pemrosesannya persis sama. Namun, solusi kasus jauh lebih mudah dibaca.