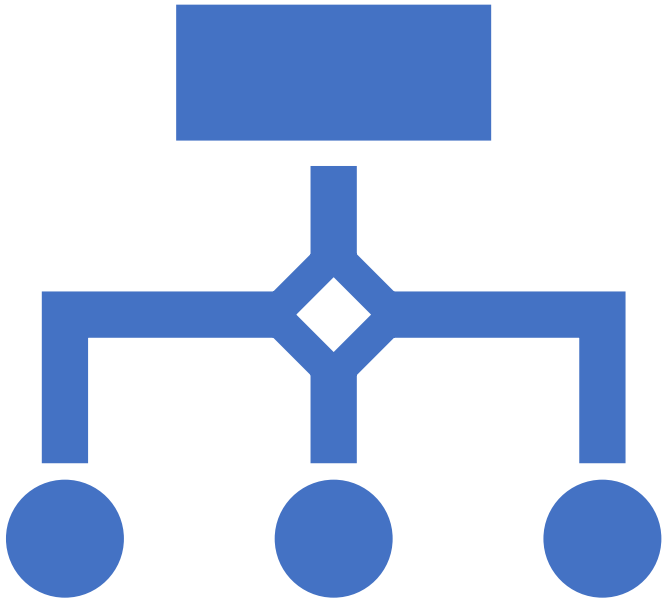


Aplikasi Data Scientist

Ledy Elsera Astrianty, S.Kom., M.Kom

Kehadiran

- Presensi (kehadiran) minimal 75%
- Presensi dibawah 75% tidak boleh mengikuti UTS/UAS

Deskripsi

- Tata cara membuat model penyelesaian kasus-kasus Sains Data dengan menggunakan bahasa pemrograman python dan perangkat-perangkat yang ada di dalamnya.

Penilaian (Capaian)

CP-MK	
M1	Mampu menggunakan bahasa pemrograman python untuk penyelesaian masalah sains data
M2	Mampu menggunakan pandas untuk data analisis dan NumPy untuk data numberik
M3	Mampu menggunakan dan matplotlib untuk visualisasi data dan statistical plot.
M4	Mampu menggunakan SciKit-Learn untuk penyekesaian kasus *machine learning*
M5	Mampu mengimplementasikan model sains data dengan bahasa pemrograman Python terhadap suatu kasus

Evaluasi

- Keaktifan
- Tugas dan Kuis
- UTS/UAS
- Proyek Tugas Akhir

Silabi

1. Dasar Python
2. Python untuk data analisis dengan NumPy, dan Pandas
3. Python untuk data visualisasi dengan matplotlib, dan seaborn
4. Python untuk data modeling dengan SciKit-Learn (Linier Regresi, logistik regresi, Decision Tree, Random Forest, dan K- Mean)
5. Implementasi siklus hidup sains data
6. Proyek akhir dengan kasus-kasus di Dunia Nyata

Nama (variable), tipe, ekspresi, dan nilai

Ledy Elsera Astrianty, S.Kom., M.Kom

Nama (penamaan)

- Pada dasarnya program adalah proses memanipulasi objek-objek di dalam memori, maka objek tersebut harus diberi nama.
- Objek diberi nama supaya mudah diidentifikasi, diacu, dan dibedakan dari objek lainnya.

Aturan Penamaan dalam Pseudocode:

1. Nama harus dimulai dengan huruf alphabet, tidak boleh dimulai dengan angka, spasi atau karakter khusus lainnya.
2. Huruf besar atau huruf kecil tidak dibedakan. Jadi suatu nama yang ditulis dalam huruf besar atau huruf kecil dianggap sama.
3. Karakter penyusun nama hanya boleh huruf alphabet, angka, dan "_". Karakter garis bawah/underscore dihitung sebagai sebuah huruf.
4. Nama tidak boleh mengandung operator aritmetika, operator relasional, tanda baca, dan karakter khusus lainnya.
5. Karakter-karakter di dalam nama tidak boleh dipisah dengan spasi.
6. Panjang nama tidak dibatasi.

Contoh Nama (penamaan)

Contoh nama yang salah:

6titik	{karena dimulai dengan angka}
nilai ujian	{karena dipisahkan dengan spasi}
PT-1	{karena mengandung operator aritmetik kurang}
hari!	{karena mengandung karakter khusus}
A 1	{karena mengandung spasi}

Contoh nama yang benar:

Titik6 atau titik_6

nilai_ujian atau nilaiujian

PT_1 atau PT1

hari

A1 atau A_1

Tipe Data

- Pada umumnya, program komputer bekerja dengan memanipulasi objek (data) di dalam memori. Objek yang akan diprogram bermacam-macam tipenya, misalnya tipe numerik, karakter, string dsb. Pada algoritma Euclidean, sebagai contoh, m dan n adalah objek yang dimanipulasi yang bertipe integer (bilangan bulat).

- Tipe data dapat dikelompokkan menjadi atas **dua macam**: ***tipe data dasar dan tipe data bentukan***. Tipe data dasar adalah tipe data yang dapat langsung dipakai, sedangkan tipe data bentukan dibentuk dari tipe data dasar atau dari tipe data bentukan lain yang sudah didefinisikan.

Tipe Data (dalam Python)

Text Type:	`str`
Numeric Types:	`int`, `float`, `complex`
Sequence Types:	`list`, `tuple`, `range`
Mapping Type:	`dict`
Set Types:	`set`, `frozenset`
Boolean Type:	`bool`
Binary Types:	`bytes`, `bytearray`, `memoryview`
None Type:	`NoneType`

```python
a, b, c = 1, 2, "Mantap"

print('a:', a)
print('b:', b)
print('c:', c)
```

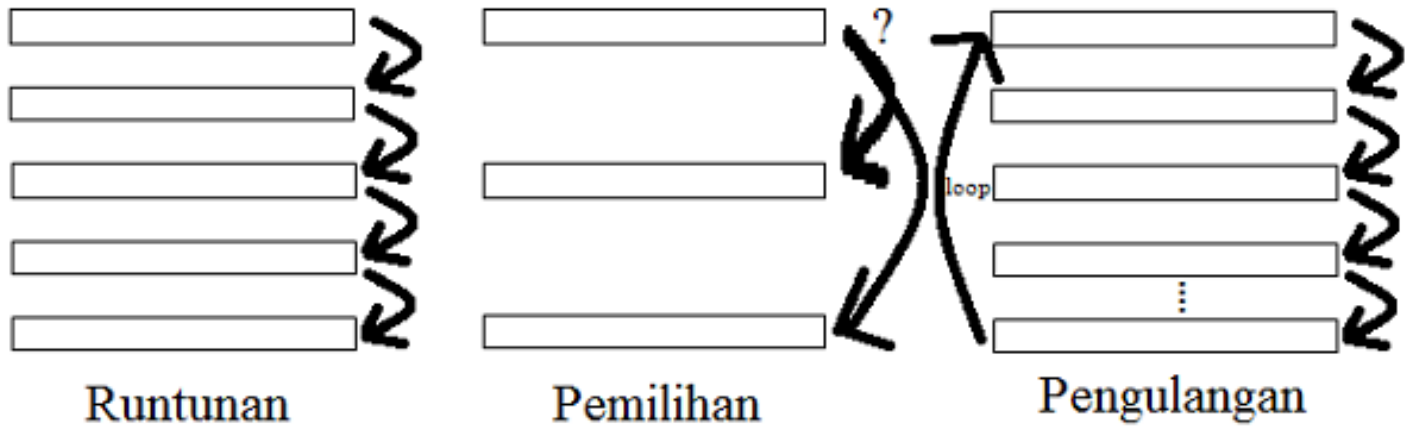
```python
d = e = f = 10

print('d:', d)
print('e:', e)
print('f:', f)
```

```python
nama = 'Nurul Huda'
usia = 24
sudah_menikah = True

print('nama:', nama)
print('ujia:', usia)
print('sudah menikah:', sudah_menikah)
```

```python
a = 'Madura'
b = 50

print(type(a))
print(type(b))
```

Tiga Konstruksi Dasar

- Sebuah **algoritma** dibangun dari **tiga konstruksi dasar**, yaitu **runtunan**, **pemilihan**, dan **pengulangan**



Sequence

- Contoh algoritma sequence (?)
- Urutan instruksi dalam algoritma adalah penting. Urutan instruksi menunjukkan urutan logika penyelesaian masalah.
- Urutan instruksi yang berbeda mungkin tidak ada pengaruh terhadap solusi persoalan, tetapi mungkin juga menghasilkan keluaran yang berbeda, tergantung pada masalahnya

Sequence

- Contoh urutan instruksi yang **berbeda** tetapi **tidak** mempengaruhi hasil.

Deklarasi:
A, B, C, D : integer
Deskripsi:
read (A, B)
C ← A + B
D ← A * B
write (C, D)

Deklarasi:
A, B, C, D : integer
Deskripsi:
read (A, B)
D ← A * B
C ← A + B
write (C, D)

- Contoh urutan instruksi yang **berbeda** tetapi mempengaruhi hasil.

Deklarasi:
A, B, C, D : integer
Deskripsi:
C ← A + B
D ← A * B
read (A, B)
write (C, D)

Selection

- Memungkinkan suatu aksi dieksekusi jika suatu kondisi terpenuhi atau tidak terpenuhi.
- Struktur pemilihan mampu memungkinkan pemroses mengikuti jalur aksi yang berbeda berdasarkan kondisi yang ada.
- Tidak setiap baris program akan dikerjakan.
- Baris program akan dikerjakan jika memenuhi syarat.
- Jadi, struktur pemilihan adalah: struktur program yang melakukan proses pengujian untuk mengambil suatu keputusan apakah suatu baris program atau blok instruksi akan diproses atau tidak.
- Pengambilan keputusan menggunakan pernyataan Boolean (true/false) dengan menggunakan operator pembanding(>,<,>=,<=,==,!=) yang bisa dikombinasikan dengan operator Boolean (AND, OR dan NOT).

Selection

- Contoh
- 5=5 ? //true
- 3=4 ? //false
- 3>1 ? //true
- 5<2 ? //false
- A=5 ?
- (A>5) and (B=2) ? //

Selection

1. Program penentuan_lulus;
2. N: integer;
3. Mulai
4. Print("Masukkan Nilai Siswa");
5. Baca(N);
6. If N>=70 then
7. Print("Siswa dinyatakan Lulus");
8. Else
9. Print("Siswa dinyatakan tidak Lulus");
10. Selesai

Nb: tidak semua baris diproses

Repeatition

- **Pengulangan** adalah instruksi yang dapat mengulang sederetan instruksi secara berulang sesuai persyaratan yang ditetapkan.
- Salah satu kelebihan computer adalah mampu mengerjakan pekerjaan yang sama berulang kali tanpa kenal lelah
- Struktur pengulangan memungkinkan kita untuk membuat suatu algoritma dari instruksi yang berulang lebih efektif
- Contoh: mencetak suatu kalimat sebanyak 100 kali

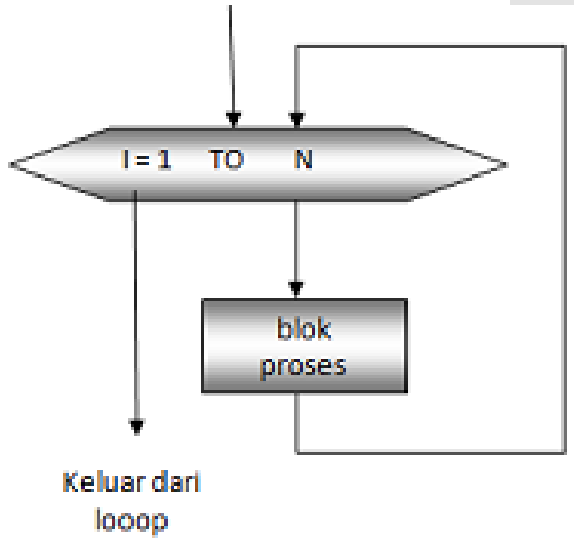
Repeatition

- Struktur instruksi perulangan pada dasarnya terdiri atas:
  - Kondisi perulangan; suatu kondisi yang harus dipenuhi agar perulangan dapat terjadi.
  - Badan(body) perulangan; deretan instruksi yang akan diulang-ulang pelaksanaannya.
  - Pencacah (counter) perulangan; suatu variabel yang nilainya harus berubah agar perulangan dapat terjadi dan pada akhirnya membatasi jumlah perulangan yang dapat dilaksanakan

Repeatition

- Jenis
  1. For loop; proses pengulangan akan terus dilakukan selama kondisi loop bernilai bernar

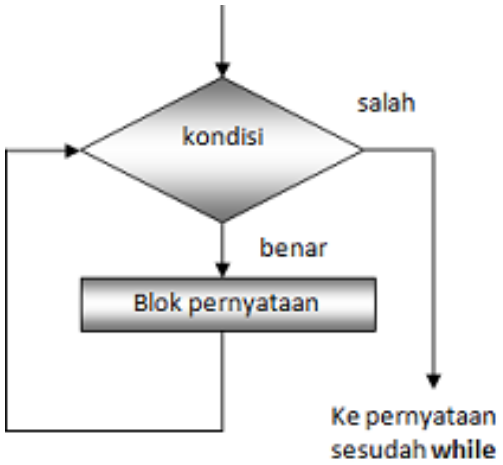
```
for (nil_awal; kondisi_loop; nil_step;)
{
  //blok loop,
  //statement yang akan diulang
}
```



Repeatition

- Jenis
  2. While loop; dilakukan pengecekan terlebih dahulu baru lanjut ke perulangan

```
while (kondisi-loop) {

      //blok loop
      //statement-statement;

}
```



Algoritma Cetak_Angka
{mencetak 1, 2, .., 8 ke piranti keluaran}
Deklarasi :
- K: integer

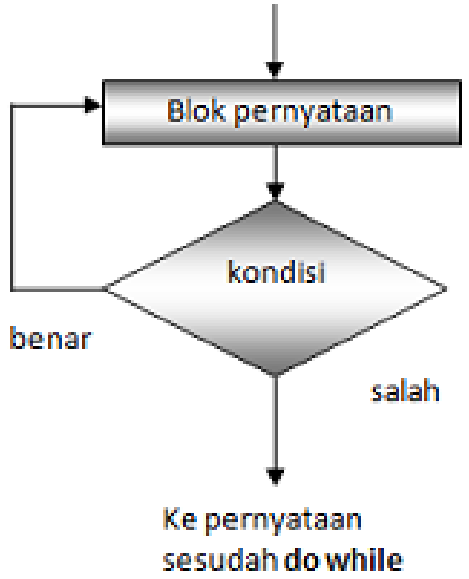
Deskripsi :
- K = 1 {inisialisasi}
- while k <= 8 do
  - write (k)
  - k = k + 1

endwhile

Repeatition

- Jenis
  3. Do-while loop; mengecek kondisi dibelakan (minimal menghasilkan satu keluaran)

```
do
 {
        //blok loop
        //statement-statement;


   }
} while (kondisi-loop);
```

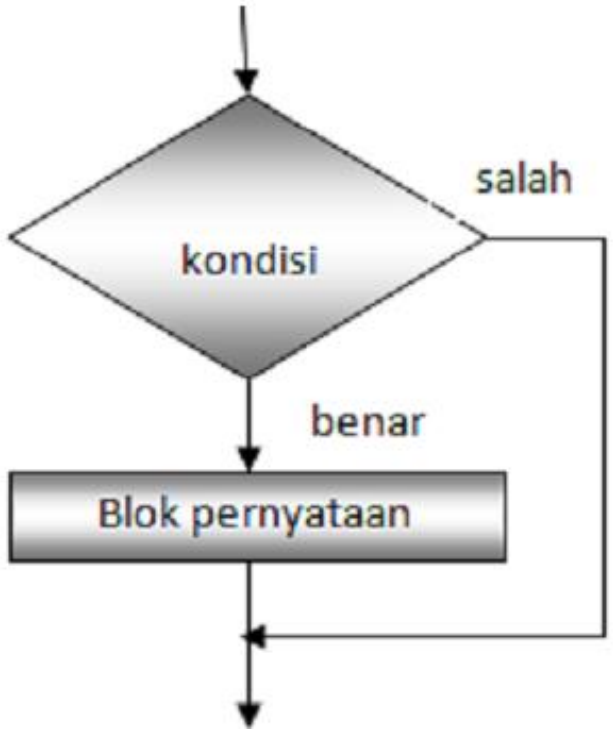


Kapan menggunakan while?

- 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.
- Namun, untuk proses yang jumlah pengulangannya 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 ini berhenti, tetapi kita menjamin bahwa jika kondisi bernilai salah, pasti pengulangan akan berhenti.

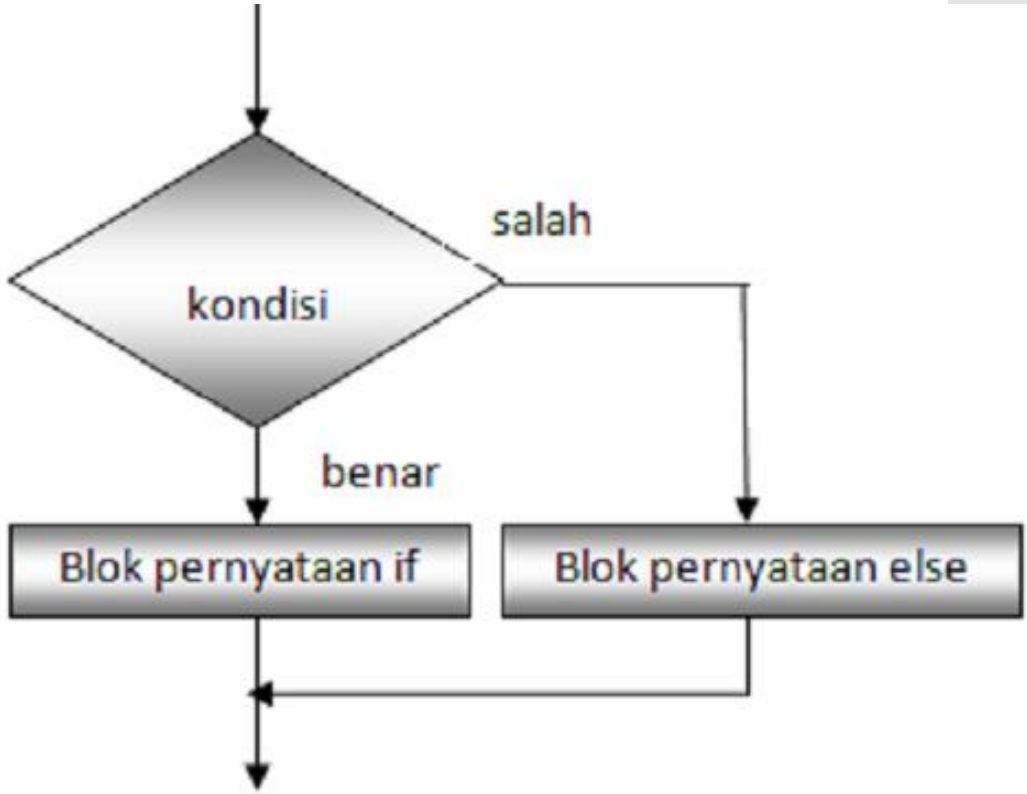
Bentuk umum if

```
if (kondisi) {
    //blok pernyataan yang
    //dijalankan
    //kalau kondisi benar
}
```



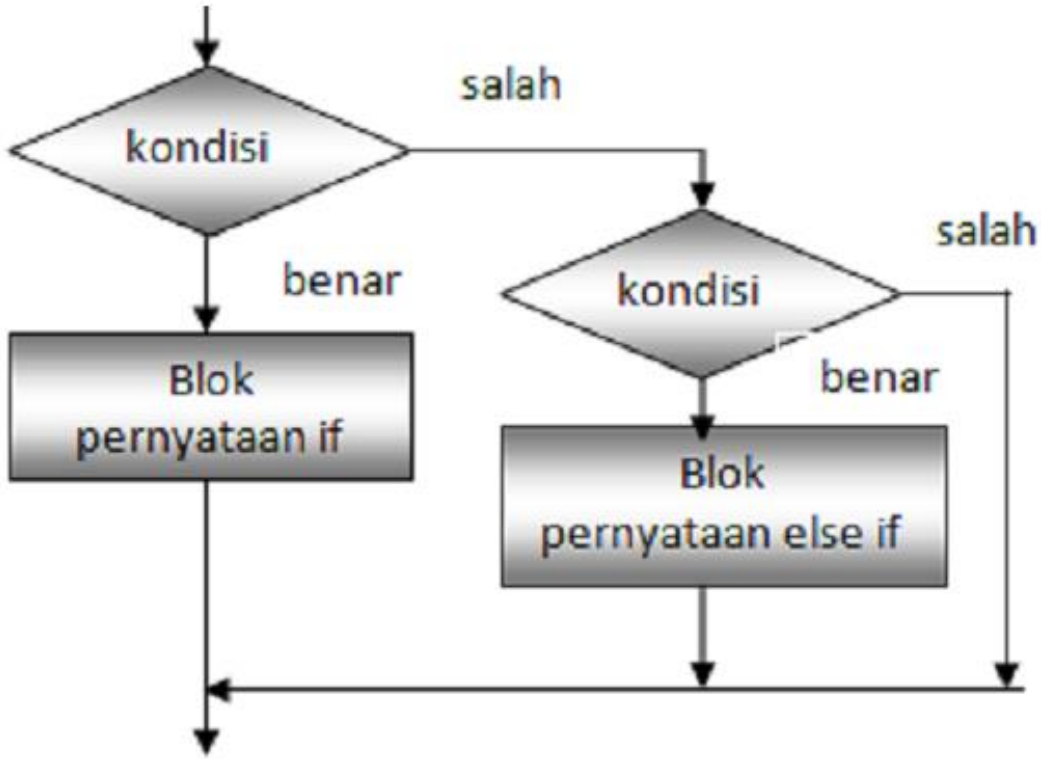
Statement if-else

```
if (kondisi) {
   //blok pernyataan yang
   //dijalankan
   //kalau kondisi benar
} else{
   //blok pernyataan yang
   //dijalankan
   //kalau kondisi salah
}
```



Statement if-else bersarang

```
if (kondisi) {
   //blok pernyataan yang
   //dijalankan
   //kalau kondisi benar
} else if (kondisi){
       //blok pernyataan yang
       //dijalankan
       //kalau kondisi benar
      }
      else {
       //blok pernyataan yang
       //dijalankan
       //kalau kondisi salah
       }
```



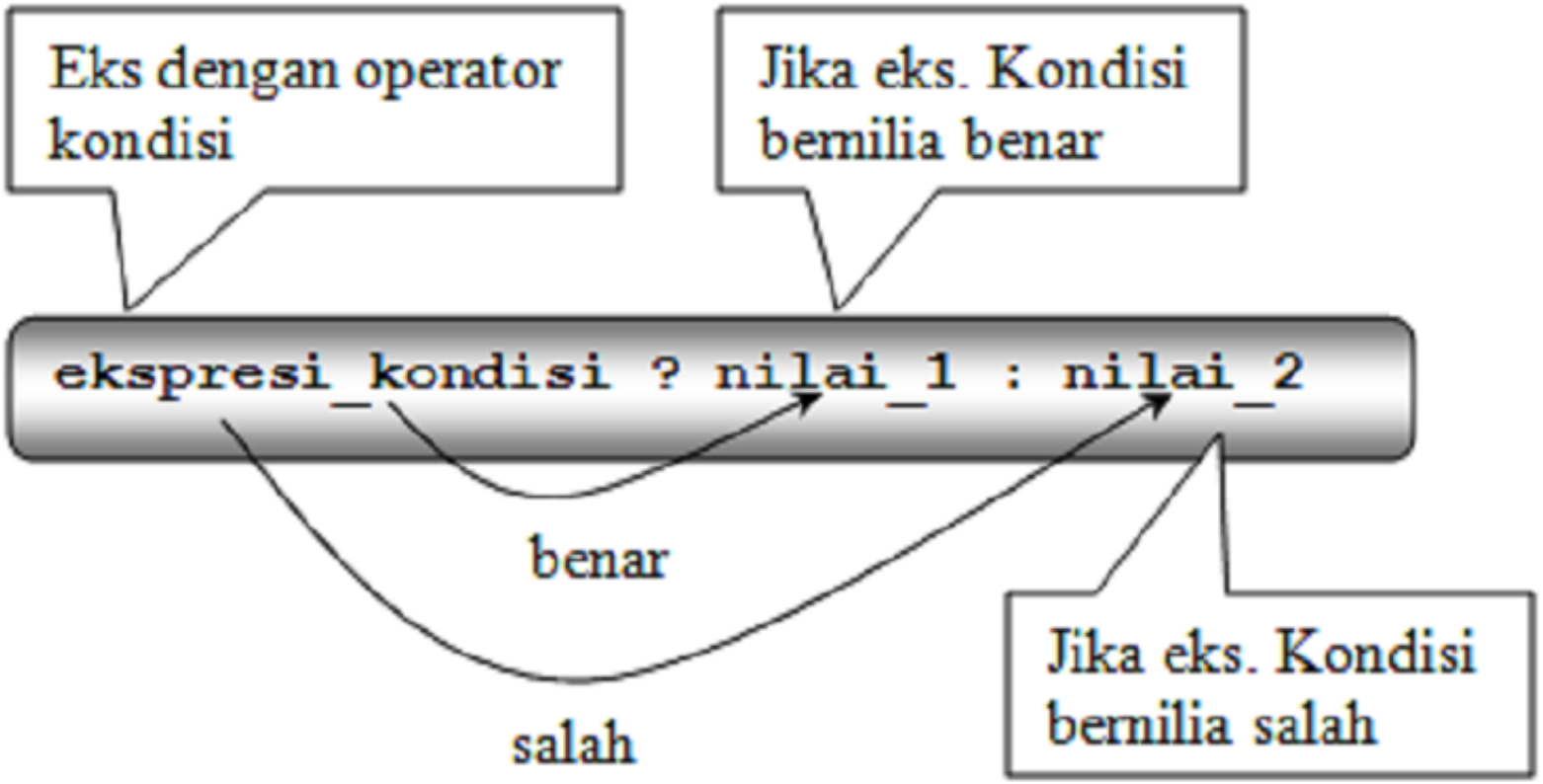
Opertor kondisi (ternary)

Operator terkondisi dikenal dengan sebutan operator ternary, karena operator ini melibatkan tiga buah argumen. Kaidah pemakaian operator ini:

ekspresi_kondisi ? nilai_1 : nilai_2

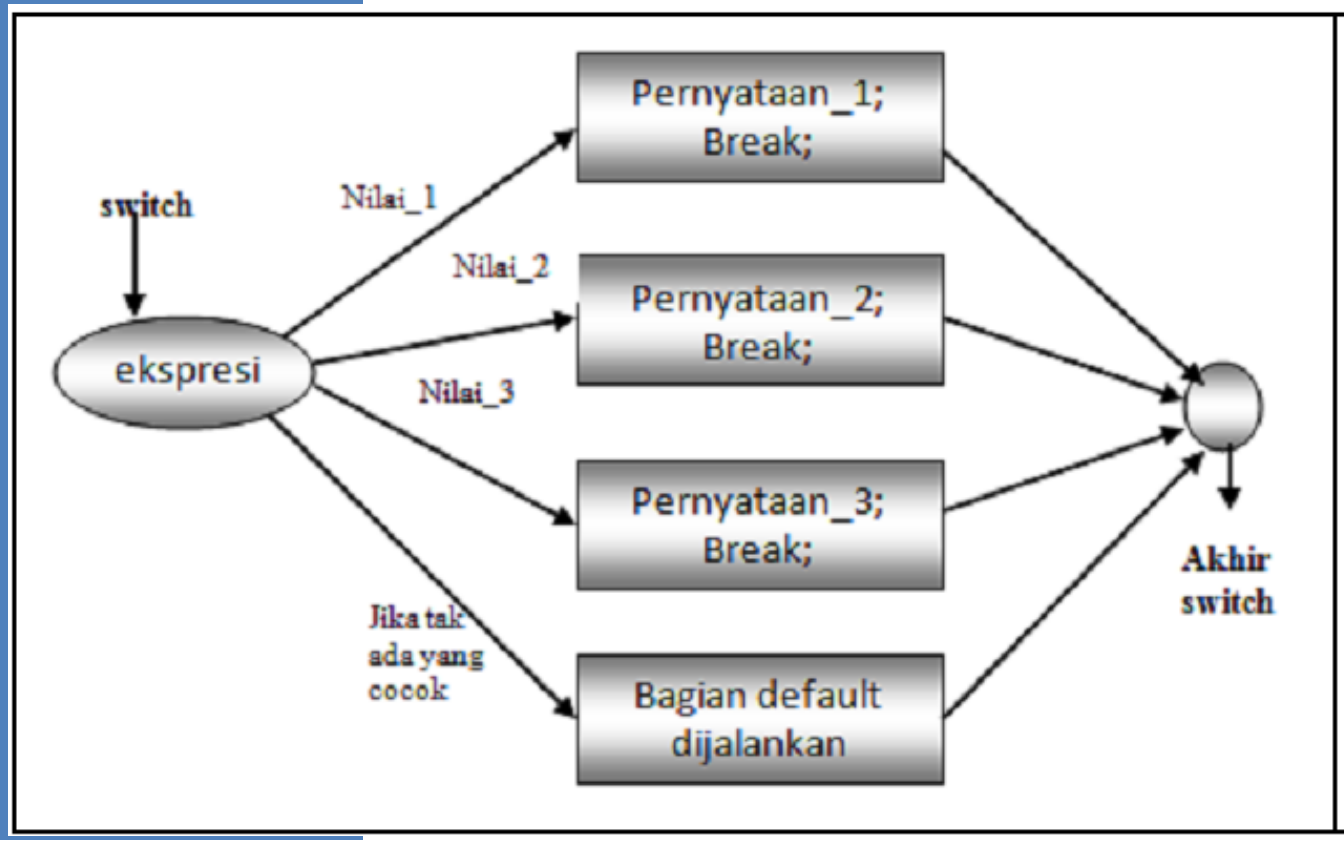
Dalam hal ini jika ekspresi_kondisi bernilai benar makan ekspresi dengan operator ?: ini menghasilkan nilai_1. Untuk keadaan sebaliknya, hasil ekspresi berupa nilai_2.

Opertor kondisi (ternary)



Switch Case

Perintah switch memungkinkan untuk melakukan sejumlah tindakan berbeda terhadap sejumlah kemungkinan nilai. Bentuk perintah ini:




```
switch (ekspres) {
    case nilaiSatu:
            Pernyataan_1;
            break;
    case nilaiDua:
            Pernyataan_1;
            break;
    case nilaiTiga:
            Pernyataan_1;
            break;
      ..............
    default:pernyataan_n;
}
```

Contoh

Karyawan honorer di PT "ABC" digaji berdasarkan jumlah jam kerjanya selama satu minggu. Upah per jam misalkan Rp 2000,00. Bila jumlah jam kerja lebih besar dari 48 jam, maka sisanya dianggap sebagai jam lembur. Upah lembur misalkan Rp3000,00/jam. Tulislah algoritma yang membaca jumlah jam kerja seorang karyawan selama satu minggu lalu menentukan upah mingguannya!

Penyelesaian

Misalkan jumlah jam kerja karyawan adalah JJK.

Kasus1: Jika JJK ≤ 48, maka upah = JJK * 2000

Kasus2: Jika JJK > 48, maka

lembur = JJK – 48

upah = 48 * 2000 + lembur * 3000

Contoh

```
PROGRAM UpahKaryawan
{menentukan upah mingguan seorang karyawan}
DEKLARASI
    nama: string
    JJK: integer
    lembur: integer
    upah: real

ALGORITMA:
    read(nama, JJK)
    if JJK ≤ 48 then
        upah ← JJK * 2000
    else
        lembur ← JJK - 48
        upah ← 48 * 2000 + lembur * 3000
    end if
    write(upah)
```

Kesimpulan IF vs CASE

Unggul yang mana antara struktur *if* dengan struktur *case*.?

Sebenarnya kalau dikatakan unggul, keduanya sama-sama unggul. Karena struktur *case* memiliki kelebihan dari pada struktur *if,* dan struktur *if* pun memiliki kelebihan dari pada struktur *case.*

Kelebihan statemen *case* terletak pada struktur script atau codenya yang lebih ringkas dibandingkan dengan struktur pada *if.*

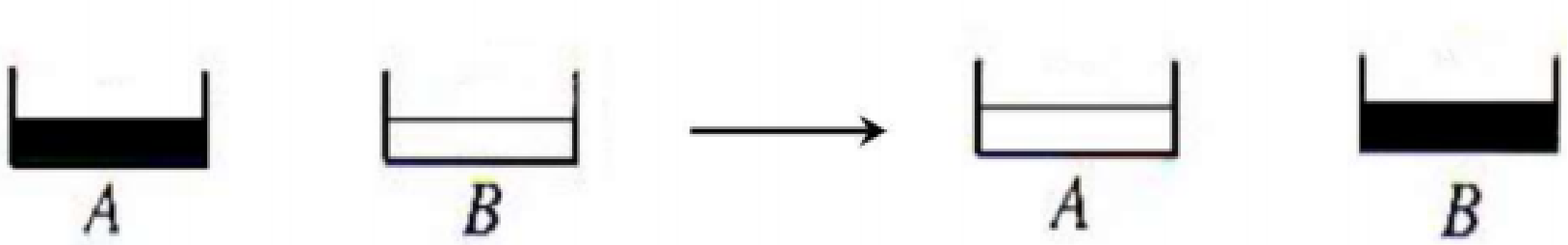
Sedangkan kelebihan statemen *if* yaitu struktur *if* dapat menyelesaikan hampir seluruh permasalahan, sedang *case* tidak semua permasalahan dapat diselesaikan dengan ini.

Mengapa statemen *if* dapat menyelesaikan hampir semua masalah, sedangkan statemen *case* tidak bisa?

Karena pada statemen if dapat menjalankan suatu *percabangan di dalam percabangan* yang disebut juga dengan istilah ***if bersarang***, dan pada statemen case tidak ada hal yang seperti itu.

Pertukaran isi gelas

- Tinjau sebuah masalah sederhana yakni mempertukarkan isi dari dua buah gelas. Gelas A berisi air kopi dan gelas B berisi air susu
- Kita ingin mempertukarkan isi kedua gelas itu sedemikian sehingga gelas A akan berisi air susu dan gelas B berisi air kopi



Mengambil Air 4 Liter

Kita diperintahkan untuk mendapatkan air sebanyak 4 liter (tidak kurang tidak lebih) dari sebuah danau. Sedangkan kita hanya punya alat dua buah ember masing-masing berkapasitas 5 liter dan 3 liter. Bagaimana caranya mendapatkan air 4 liter dengan tepat tanpa menggunakan alat lain maupun hanya mengira-ira

Thank You

- Ledy Elsera Astrianty
- 083847956052
- ledyelsera@gmail.com