Percabangan dan Perulangan

Percabangan

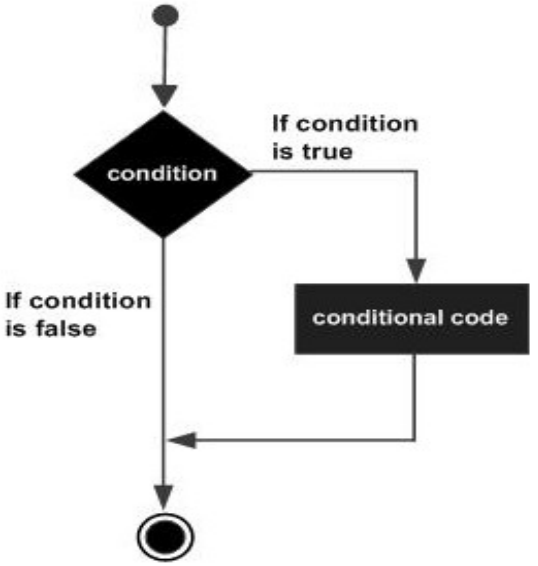
Percabangan atau *decision* digunakan untuk mengatur aliran logika pada suatu program. Pada **Octave** kita dapat menerapkan percabangan dengan menggunakan beberapa *statement* berikut:

- *If-else*
- *If-elseif-else*
- *Nested if*
- *Switch*
- *Nested switch*

If

Statement if digunakan untuk mengatur aliran logika pada program. Pada **Octave** kita dapat menerapkan if-else dengan format penulisan sebagai berikut.

```
if <expression>
    % akan dieksekusi jika
    % expression bernilai true (benar)
    <statement>
end
```



If

Contoh:

```
a = 45;
if a < 100
   printf(“a lebih kecil dari 100\n”);
end
```

Output:

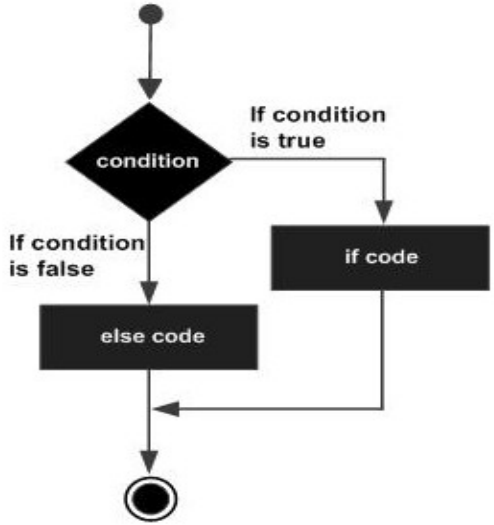
```
>> a lebih kecil dari 100
```

If-else

Dengan menggunakan *statement* if, kita dapat mengatur aliran logika program, namun hanya ketika ***expression* pada *if* bernilai *true*** (benar). Kita kadang juga **perlu menangani alur jika suatu *expression* bernilai *false***.
Maka digunakan *statement* else!

Format penulisan:

```
if <expression>
    % eksekusi jika expression true
else
    % eksekusi jika expression false
end
```



If-else

Contoh:

```
a = 125;
if a < 100
    printf(“a lebih kecil dari 100\n”);
else
    printf(“a lebih besar dari 100\n”);
end
```

Output:

```
>> a lebih besar dari 100
```

Else If

Dengan menggunakan *statement* if-else, kita dapat menangani jika suatu *expression* bernilai *true* (benar) atau *false* (salah) saja. Lalu bagaimana jika kita ingin melihat kemungkinan lain pada alur logika program?

Perhatikan kode berikut:

```
a = 10;
if a == 20
    printf(“a bernilai 20\n”);
else
    printf(“a tidak bernilai 20\n”);
end
```

Else If

Pada kode sebelumnya, dengan *statement* if-else kita hanya dapat "**bertanya**" apakah **nilai a bernilai 20** atau **tidak saja**. Dengan menggunakan ***statement* else-if**, kita dapat memberikan "**pertanyaan**" yang lain (dalam hal ini pertanyaan dimaksudkan *expression*). Misalnya apakah nilai a bernilai 10, apakah nilai a bernilai 20 dan sebagainya.

Else If

Permasalahan tadi dapat **diselesaikan** dengan menggunakan ***statement* else-if**. Berikut format penulisan *statement* else-if:

```
if <expression 1>
    % eksekusi jika expression 1 bernilai true
else if <expression 2>
    % eksekusi jika expression 2 bernilai true
else
    % eksekusi jika semua expression bernilai false
end
```

If-else

Contoh:

```
a = 100;
if a == 200
    printf(“a bernilai 200\n”);
else if a == 100
    printf(“a bernilai 100\n”);
else
    printf(“a bukan bernilai 100 dan 200\n”);
end
```

Output:

```
>> a bernilai 100
```

Nested if

Kita juga dapat **menangani kembali** alur logika program **pada blok kode** yang **berada dalam *statement* if-else**. Dengan kata lain terdapat *statement* if-else di dalam *statement* if-else, yakni *nested if statement*!

Format penulisan:

```
if <expression 1>
    % eksekusi jika expression 1 bernilai true
    if <expression 2>
        % eksekusi jika expression 2 bernilai true
    end
end
```

Nested if

Contoh:

```
a = 150;
if a > 100
    if a < 200
        printf(“Nilai a berada diantara 100 dan 150\n”);
    end
end
```

Output:

```
>> Nilai a berada diantara 100 dan 150
```

Switch

Statement **Switch** digunakan untuk mengatur aliran program, bedanya dengan *statement* if-else, *switch* **beroperasi pada satu nilai** yang mengatur alur program dan satu nilai tersebut akan **dibandingkan** menggunakan ***expression boolean***. Format penulisan:

```
switch (n)
    case <expression>
        <blok kode>
    case <expression>
        <blok kode>
    otherwise
        <blok kode>
end
```

Switch

Contoh:

```
a = 150;
switch(a)
    case 100
        printf(“Nilai a adalah 100\n”);
    case 150
        printf(“Nilai a adalah 150\n”);
    otherwise
        printf(“Nilai a bukan 100 dan 150\n”);
end
```

Output:

```
>> Nilai a adalah 150
```

Perulangan

Perulangan pada program digunakan untuk mengeksekusi suatu blok kode secara berulang-ulang dengan suatu ketentuan tertentu. Pada Octave kita dapat menerapkan perulangan dengan beberapa *statement* berikut:

- *While loop*
- *For loop*
- *Nested loop*

While Loop

While loop adalah salah satu *statement* yang digunakan dalam perulangan. Format penulisan *while loop* adalah sebagai berikut:

```
while <expression>
    <blok kode>
end
```

<blok kode> akan terus dieksekusi apabila ***expression*** bernilai *true* (benar). Untuk keluar dari *while loop*, kita perlu membuat suatu *statement* yang nantinya menyebabkan *expression* bernilai *false* atau salah. Jika tidak, kita akan terjebak pada perulangan tak hingga (***Infinity loop***).

Infinite Loop

While Loop

Contoh:

```
n = 7;
while num < 10
    printf(“Nilai n masih kurang dari 10\n”);
    num = num + 1;
end
```

Output:

```
>> Nilai n masih kurang dari 10
    Nilai n masih kurang dari 10
    Nilai n masih kurang dari 10
```

For Loop

For loop adalah salah satu *statement* yang digunakan dalam penerapan pengulangan. **Perbedaan**nya dengan ***while loop*** adalah untuk ***for loop*** kita sudah **tahu secara pasti berapa kali** kita ingin **mengulang eksekusi** dari suatu blok kode.

Format penulisan *for loop*:

```
for index = vector
   <blok kode>
end
```

While Loop

Contoh:

```
for i = 1:5
    printf(“%d “, i);
end
```

Output:

```
>> 1
    2
    3
    4
    5
```

Nested Loop

Kita juga dapat memasukan suatu *looping* pada suatu *looping* lain. Atau dengan kata lain, terdapat *loop* di dalam *loop* lain (*nested*).

Format penulisan:

```
for index = start:end
    for index = start:end
        <blok kode>
    end
end
```

Nested Loop

Contoh:

```
for i = 1:2
    for j = 1:2
        printf("i: %d dan j: %d\n", i, j);
    end
end
```

Output:

```
>> i: 1 dan j: 1
   i: 1 dan j: 2
   i: 2 dan j: 1
   i: 2 dan j: 2
```

Programming in Octave

~TERIMAKASIH~