1.5 Operasi matematika

1.5.1 Penjumlahan matrik

Operasi penjumlahan pada dua buah matrik hanya bisa dilakukan bila kedua matrik tersebut berukuran sama. Misalnya matrik $\mathbf{C}_{2\times3}$

$$\mathbf{C} = \begin{bmatrix} 9 & 5 & 3 \\ 7 & 2 & 1 \end{bmatrix}$$

dijumlahkan dengan matrik $\mathbf{A}_{2\times3}$, lalu hasilnya (misalnya) dinamakan matrik $\mathbf{D}_{2\times3}$

D = A + C

$$\begin{aligned} \mathbf{D} &= \begin{bmatrix} 3 & 8 & 5 \\ 6 & 4 & 7 \end{bmatrix} + \begin{bmatrix} 9 & 5 & 3 \\ 7 & 2 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 3+9 & 8+5 & 5+3 \\ 6+7 & 4+2 & 7+1 \end{bmatrix} \\ &= \begin{bmatrix} 12 & 13 & 8 \\ 13 & 6 & 8 \end{bmatrix} \end{aligned}$$

Tanpa mempedulikan nilai elemen-elemen masing-masing matrik, operasi penjumlahan antara matrik $\mathbf{A}_{2\times3}$ dan $\mathbf{C}_{2\times3}$, bisa juga dinyatakan dalam indeks masing-masing dari kedua matriktersebut, yaitu

$$\begin{bmatrix} d_{11} & d_{12} & d_{13} \\ d_{21} & d_{22} & d_{23} \end{bmatrix} = \begin{bmatrix} a_{11}+c_{11} & a_{12}+c_{12} & a_{13}+c_{13} \\ a_{21}+c_{21} & a_{22}+c_{22} & a_{23}+c_{23} \end{bmatrix}$$

Dijabarkan satu persatu sebagai berikut

$$\begin{aligned} d_{11} &= a_{11} + c_{11} \\ d_{12} &= a_{12} + c_{12} \\ d_{13} &= a_{13} + c_{13} \\ d_{21} &= a_{21} + c_{21} \\ d_{22} &= a_{22} + c_{22} \\ d_{23} &= a_{23} + c_{23} \end{aligned} \tag{1.6}$$

Dari sini dapat diturunkan sebuah rumus umum penjumlahan dua buah matrik

$$d_{ij} = a_{ij} + c_{ij} \tag{1.7}$$

dimana `i`=1,2 dan `j`=1,2,3. **Perhatikan baik-baik! Batas `i` hanya sampai angka 2 sementarabatas `j` sampai angka 3. Kemampuan anda dalam menentukan batas indeks sangat pentingdalam dunia *programming*.**

1.5.2 Komputasi penjumlahan matrik

Berdasarkan contoh operasi penjumlahan di atas, indeks `j` pada persamaan (1.7) **lebih cepat** berubah dibanding indeks `i` sebagaimana ditulis pada 3 baris pertama dari Persamaan (1.6),

$$d_{11} = a_{11} + c_{11}$$

$$d_{12} = a_{12} + c_{12}$$

$$d_{13} = a_{13} + c_{13}$$

Jelas terlihat, ketika indeks `i` masih bernilai 1, indeks `j` sudah berubah dari nilai 1 sampai 3.Hal inimembawa konsekuensi pada *script* pemrograman, dimana *looping* untuk indeks `j` harusdiletakkan di dalam *looping* indeks `i`. **Aturan mainnya adalah yang *looping*-nya paling cepat harus diletakkan paling dalam; sebaliknya, *looping* terluar adalah *looping* yang indeksnya paling jarang berubah.**
Bila anda masih belum paham terhadap kalimat yang dicetak tebal, saya akan berikan contoh*source code* dasar yang nantinya akan kita optimasi selangkah demi selangkah. OK, kitamulai dari *source code* paling mentah berikut ini.

```
clear all
clc

A=[3 8 5; 6 4 7]; % inisialisasi matrik A

C=[9 5 3; 7 2 1]; % inisialisasi matrik C

% ---proses penjumlahan matrik----
D(1,1)=A(1,1)+C(1,1);
D(1,2)=A(1,2)+C(1,2);
D(1,3)=A(1,3)+C(1,3);
D(2,1)=A(2,1)+C(2,1);
D(2,2)=A(2,2)+C(2,2);
D(2,3)=A(2,3)+C(2,3);

% ---menampilkan matrik A, C dan D----
A
C
D
```

Tanda % berfungsi untuk memberikan komentar atau keterangan. Komentar atau keterangan tidak akan diproses oleh Matlab. Saya yakin anda paham dengan logika yang ada padabagian % —*proses penjumlahan matrik*—- dalam source code di atas. Misalnya pada baris ke-9,elemen d_{11} adalah hasil penjumlahan antara elemen a_{11} dan c_{11}, sesuai dengan baris pertamaPersamaan 1.6.
Tahap pertama penyederhanaan *source code* dilakukan dengan menerapkan perintah *for - end* untuk proses *looping*. *Source code* tersebut berubah menjadi

```
clear all
clc

A=[3 8 5; 6 4 7]; % inisialisasi matrik A

C=[9 5 3; 7 2 1]; % inisialisasi matrik B

% ---proses penjumlahan matrik----
for j=1:3
      D(1,j)=A(1,j)+C(1,j);
end

for j=1:3
      D(2,j)=A(2,j)+C(2,j);
end

% ---menampilkan matrik A, C dan D----
```

```
A
C
D
```

Pada baris ke-9 dan ke-13, saya mengambil huruf `j` sebagai nama indeks dimana `j` bergerakdari 1 sampai 3. Coba anda pikirkan, mengapa `j` hanya bergerak dari 1 sampai 3?Modifikasi tahap kedua adalah sebagai berikut

```
clear all
clc

A=[3 8 5; 6 4 7]; % inisialisasi matrik A

C=[9 5 3; 7 2 1]; % inisialisasi matrik B

% ---proses penjumlahan matrik----
i=1
for j=1:3
D(i,j)=A(i,j)+C(i,j);
end

i=2
for j=1:3
D(i,j)=A(i,j)+C(i,j);
end

% ---menampilkan matrik A, C dan D----
A
C
D
```

```
clear all
clc

A=[3 8 5; 6 4 7]; % inisialisasi matrik A

C=[9 5 3; 7 2 1]; % inisialisasi matrik B

% ---proses penjumlahan matrik----
for i=1:2
     for j=1:3
          D(i,j)=A(i,j)+C(i,j);
     end
end
% ---menampilkan matrik A, C dan D----
A
C
D
```

Saya gunakan indeks `i` pada baris ke-9 dan ke-14 yang masing-masing berisi angka 1 dan 2.Dengan begitu indeks `i` bisa menggantikan angka 1 dan 2 yang semula ada di baris ke-11 danke-16. Nah sekarang coba anda perhatikan, statemen pada baris ke-10, ke-11 dan ke-12 samapersis dengan statemen pada baris ke-15, ke-16 dan ke-17, sehingga mereka bisa disatukankedalam sebuah *looping* yang baru dimana `i` menjadi nama indeksnya.

```
clear all
clc

A=[3 8 5; 6 4 7]; % inisialisasi matrik A
```

```
C=[9 5 3; 7 2 1]; % inisialisasi matrik B

% ---proses penjumlahan matrik----
dim=size(A);
n=dim(1);
m=dim(2);

for i=1:n
      for j=1:m
            D(i,j)=A(i,j)+C(i,j);
      end
end

% ---menampilkan matrik A, C dan D----
A
C
D
```

Coba anda pahami dari baris ke-9, mengapa indeks `i` hanya bergerak dari 1 sampai 2?Source code di atas memang sudah tidak perlu dimodifikasi lagi, namun ada sedikit saranuntuk penulisan *looping* bertingkat dimana sebaiknya *looping* terdalam ditulis agak menjorok
kedalam seperti berikut ini

```
clear all
clc

A=[3 8 5; 6 4 7]; % inisialisasi matrik A
C=[9 5 3; 7 2 1]; % inisialisasi matrik B

% ---proses penjumlahan matrik----
dim=size(A);
n=dim(1);
m=dim(2);

for i=1:n
      for j=1:m
            D(i,j)=A(i,j)+C(i,j);
      end
end

% ---menampilkan matrik A, C dan D----
A
C
D
```

Sekarang anda lihat bahwa *looping* indeks `j` ditulis lebih masuk kedalam dibandingkan loopingindeks `i`. Semoga contoh ini bisa memperjelas **aturan umum pemrograman dimana yang*looping*-nya paling cepat harus diletakkan paling dalam; sebaliknya, *looping* terluar adalah*looping* yang indeksnya paling jarang berubah.** Dalam contoh ini *looping* indeks `j` bergeraklebih cepat dibanding *looping* indeks `i`.