

Skema Standar (Bag. 4): Pemrosesan File Sekuensial

Tim Pengajar IF1210

Sekolah Teknik Elektro dan Informatika

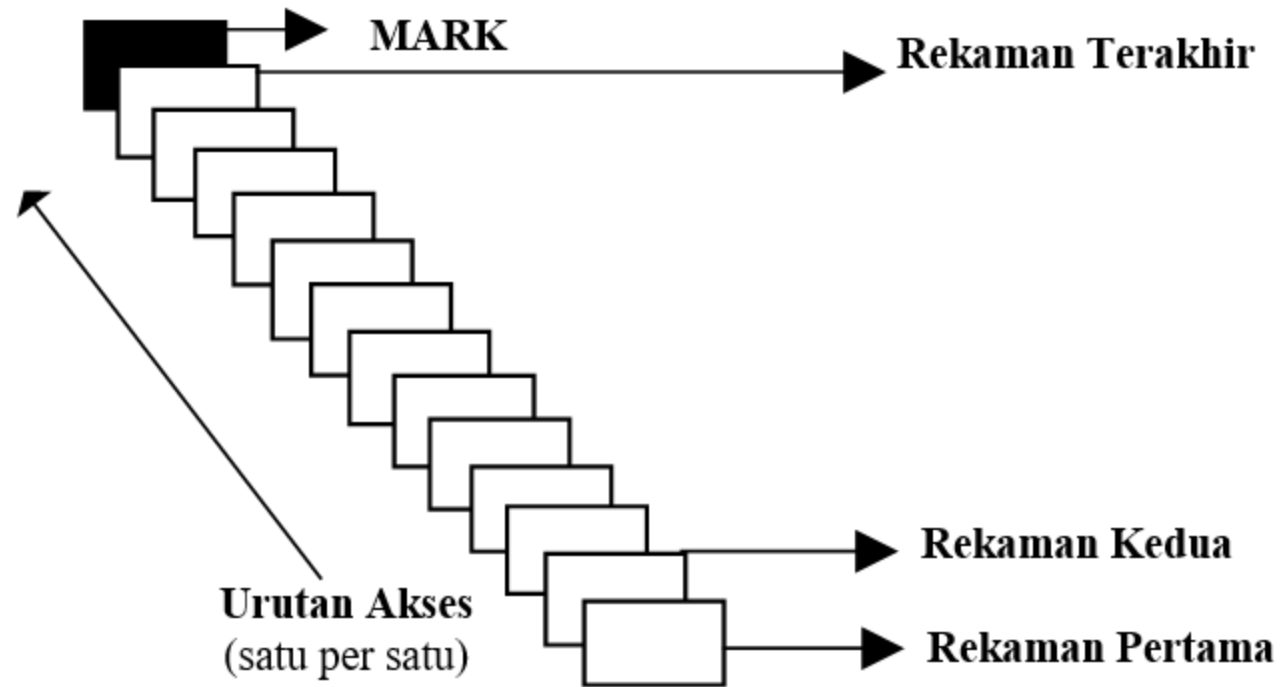
Tujuan

- Mahasiswa memahami primitif-primitif dasar dalam pemrosesan file sekuensial
- Mahasiswa memahami skema-skema dasar untuk pembacaan, penulisan dan pemrosesan file sekuensial
- Mahasiswa mengenal skema konsolidasi file dan *merging* 2 file
- Seluruhnya diimplementasikan dalam Notasi Algoritmik

Akan dipelajari...

- Skema dasar pemrosesan sekuensial:
 - Skema pembacaan dan penulisan file
 - Beberapa contoh pemrosesan sekuensial
- Skema Konsolidasi File
- Skema Merging 2 File
- Catatan implementasi di Python → diberikan dalam slide terpisah, untuk dipelajari oleh mahasiswa secara mandiri

File Sekuensial



Deklarasi

- Notasi Algoritmik

```
type rekaman : ...
```

```
{ sebuah type terdefinisi untuk setiap  
  rekaman }
```

```
NamaArsip :
```

```
  SEQFILE of
```

```
    (*) nama_rek : rekaman
```

```
    (1) mark
```

Primitif Pemrosesan File

- Assign nama fisik ke nama logik
- Membuka file
 - Membuka file untuk membaca isinya (read only)
 - Membuka file untuk menulis isinya (rewrite)
- Membaca isi file
- Menulis isi file
- Menutup file

Assign Nama Fisik ke Nama Logik

- **Assign** nama fisik (nama file di harddisk) ke nama logik (variabel dalam program)

Notasi Algoritmik

assign (*namaArsip, namaFisik*)

{ **Contoh-Contoh** }

assign (ArsipMhs, "dataMhs.dat")

assign (Dokumen, "fileteks.txt")

assign (DaftarNilai, "myNilai.dat")

Membuka File (1/2)

- **open** (namaArsip, nama_rek)
 - Mempersiapkan file untuk dibaca (read-only) sehingga dapat dibaca dengan menggunakan prosedur untuk membaca isi file
 - Rekaman pertama telah dibaca dan disimpan di nama_rek

Notasi Algoritmik

open (*namaArsip*, *nama_rek*)

{ **Contoh-Contoh** }

open (ArsipMhs, RekMhs)

open (Dokumen, CC)

open (DaftarNilai, nilai)

Membuka File (2/2)

- **rewrite** (namaArsip)
 - Mempersiapkan file untuk dibaca dan siap untuk ditulis

Notasi Algoritmik

rewrite (*namaArsip*)

{ **Contoh-Contoh** }

rewrite (ArsipMhs)

rewrite (Dokumen)

rewrite (DaftarNilai)

Membaca File

- **read** (namaArsip, nama_rek)
 - Membaca 1 buah record bertipe rekaman di dalam file of rekaman

Notasi Algoritmik

read (*namaArsip*, *nama_rek*)

{ Contoh-Contoh }

read (ArsipMhs, RekMhs)

read (Dokumen, CC)

read (DaftarNilai, nilai)

Menulis File

- **write** (namaArsip, rekaman_baru)
 - Menuliskan 1 buah record bertipe rekaman di dalam file of rekaman

Notasi Algoritmik

write (*namaArsip*, *rekaman_baru*)

{ Contoh-Contoh }

Mhs.NIM \leftarrow 165

Mhs>Nama \leftarrow "Amir"

Mhs.Nilai \leftarrow 100

write (ArsipMhs, Mhs)

write (Dokumen, 'C')

write (DaftarNilai, 95)

Menutup File

- **close** (NamaArsip)
 - File “ditutup”, tidak dapat diakses dan ditulis lagi
 - Harus berpasangan dengan **open/rewrite**

Notasi Algoritmik

close (*namaArsip*)

{ Contoh-Contoh }

close (ArsipMhs)

close (Dokumen)

close (DaftarNilai)

EOP (End Of Process)

- Rekaman terakhir pada file sekuensial adalah rekaman fiktif (**mark**) yang menandai akhir dari file
- **EOP**
 - Fungsi/variabel yang menghasilkan true jika pemrosesan mencapai **mark** yang menandai akhir dari file

KAMUS

```
f : SEQFILE OF
    (*) I : integer
    (1) 9999
```

function EOP (I : integer) → boolean

ALGORITMA

```
open(f, I)
if not (EOP(I)) then
    repeat
        output(I)
        read(f, I)
    until EOP(I)
else { EOP }
    output("File kosong")
close (f)
```

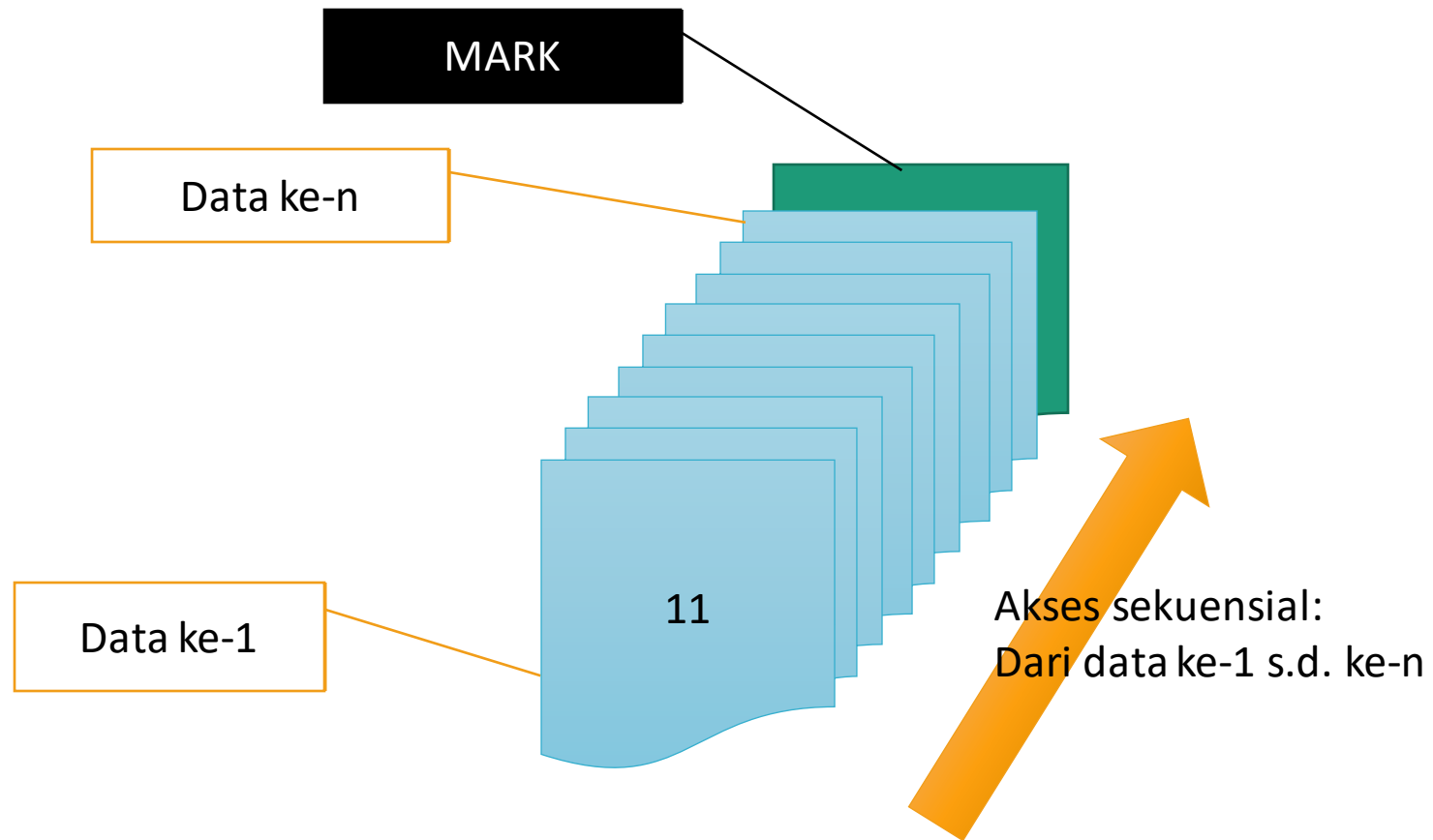
Pemrosesan File Secara Sekuensial

Definisi File Sekuensial

- **Sequential file** (Arsip sekuensial): sekumpulan rekaman yang disimpan dalam media penyimpanan sekunder komputer, yang dapat diakses secara sekuensial mulai dari rekaman pertama sampai dengan rekaman yang terakhir, rekaman per rekaman secara searah saja.

File Sekuensial

- File yang dibaca secara sekuensial dari awal sampai akhir:
 - Tidak ada akses di tengah file
 - Akses hanya bisa maju, tidak bisa mundur, atau lompat
- Untuk itu file harus diproses juga secara sekuensial
- Data yang tersimpan dalam file memiliki type yang sama:
 - *text, file of integer, file of real, dll.*



Menandai Akhir File

- Menggunakan **mark** khusus
 - **MARK**: Suatu rekaman yang dinyatakan sebagai akhir file dan menandai akhir dari pemrosesan sekuensial (*end of process*)
 - File “selalu isi” → minimum berisi MARK (jika hanya berisi MARK berarti file kosong)

Skema Dasar Pemrosesan File Secara Sekuensial

- Pembacaan File
 - Pemeriksaan langsung terhadap MARKModel pemrosesan sekuensial dengan mark
→ Menggunakan while atau repeat-until
- Penulisan Isi File
- Pemrosesan Sekuensial Lain:
 - Mencari nilai rata-rata
 - Menyalin isi file ke array dan sebaliknya

Skema Dasar Pembacaan File

- Memanfaatkan pemeriksaan terhadap **MARK** untuk menghentikan pembacaan file
 - Model pemrosesan sekuensial dengan mark menggunakan if-then-else + repeat-until
 - Pemrosesan terhadap kasus kosong
 - Model dengan mark menggunakan while-do
- Contoh:
 - Membaca file **teks**
 - MARK berupa character '.' (titik)

Skema Dasar Pembacaan File Repeat-Until

Program BacaText2a

{ Membaca sebuah text file diakhiri dengan MARK berupa '.' dan menuliskan isinya ke layar. Asumsi file dataku.txt ada dan sudah terisi, minimum hanya berisi '.' (artinya file kosong). }

KAMUS

f : SEQFILE of
 (*) cc : character
 (1) '.'

ALGORITMA

```

assign (f, "dataku.txt")
open (f,cc)    { First-Elmt }
if (cc = '.') then
    output ("File kosong")
else { not EOP(cc), file tidak kosong }

    repeat
        output (cc)    { Proses current elmt. }
        read (f,cc)    { Next-Elmt }
    until (cc = '.')

close (f)
  
```

Skema Dasar Pembacaan File Repeat-Until

Program BacaText2a

{ Membaca sebuah text file diakhiri dengan MARK berupa '.' dan menuliskan isinya ke layar. Asumsi file dataku.txt ada dan sudah terisi, minimum hanya berisi '.' (artinya file kosong). }

KAMUS

f : SEQFILE of
 (*) cc : character
 (1) '.'

ALGORITMA

```

assign (f, "dataku.txt")
open (f, cc) { First-Elmt }
if (cc = '.') then
    output ("File kosong")
else { not EOP(cc), file tidak kosong }

    repeat
        output (cc) { Proses current elmt. }
        read (f, cc) { Next-Elmt }
    until (cc = '.')

close (f)

```

EOP: CC = ''

Skema Dasar Pembacaan File While-do

Program BacaText2b

{ Membaca sebuah text file diakhiri dengan MARK berupa '.' dan menuliskan isinya ke layar. Asumsi file dataku.txt ada dan sudah terisi, minimum hanya berisi '.' (artinya file kosong). }

KAMUS

f : SEQFILE of
 (*) cc : character
 (1) '.'

ALGORITMA

```
assign (f, "dataku.txt")
open (f, cc) { First-Elmt }
while (cc ≠ '.') do
    output (cc) { Pemrosesan Current Elmt. }
    read (f, cc) { Next-Elmt }
{ EOP : cc = '.' }
close (f)
```

Skema Dasar Pembacaan File While-do

Program BacaText2b

{ Membaca sebuah text file diakhiri dengan MARK berupa '.' dan menuliskan isinya ke layar. Asumsi file dataku.txt ada dan sudah terisi, minimum hanya berisi '.' (artinya file kosong). }

KAMUS

f : SEQFILE of
 (*) cc : character
 (1) '.'

ALGORITMA

```

assign (f, "dataku.txt")
open (f, cc) { First-Elmt }
while (cc ≠ '.') do
    output (cc) { Pemrosesan Current Elmt. }
    read (f, cc) { Next-Elmt }
{ EOP : cc = '.' }
close (f)

```

EOP: CC = '.'

Skema Penulisan File

- Membaca masukan dari pengguna dan menyimpannya ke dalam file
- Model pemrosesan sekuensial dengan mark, penulisan diakhiri dengan memasukkan nilai tertentu
- Nilai mark dijadikan penanda akhir *file*
- Contoh-1:
 - Mengisi file **teks**
 - Diakhiri dengan masukan berupa character ‘.’
- Contoh-2:
 - Mengisi file integer
 - Diakhiri dengan masukan berupa integer 9999

Skema Dasar Penulisan File

Contoh-1: File Teks – While-Do

Program IsiTeks

{ Membaca sejumlah masukan character dari user sampai dimasukkan nilai '.' dan menyimpannya ke dalam file teks dataku.txt }

KAMUS

f : SEQFILE of
 (*) C1 : character
 (1) '.'

ALGORITMA

```

assign (f, "dataku.txt")
rewrite (f)
input (C1)           { First-Elmt }
while (C1 ≠ '.') do
    write (f,C1)       { Proses current elmt. }
    input (C1)         { Next-Elmt }
{ C1 = '.' }
write (f, '.')        { tulis MARK di akhir file }
close (f)
  
```

Skema Dasar Penulisan File

Contoh-2: File Integer – Repeat-Until

Program IsiFileInt

{ Membaca sejumlah masukan integer dari user sampai dimasukkan nilai 9999 dan menyimpannya ke dalam file of integer dataku.dat }

KAMUS

f : SEQFILE of
 (*) x : integer
 (1) 9999

ALGORITMA

```

assign (f, "dataku.dat")
rewrite (f)
input (x) { First-Elmt }
if (x = 9999) then
  output ("File kosong")
else { x ≠ 9999, file tidak kosong }
  repeat
    write (f,x) { Proses current elmt. }
    input (x) { Next-Elmt }
  until (x = 9999) { x = 9999 }
write (f, 9999) { tulis MARK di akhir file }
close (f)
  
```

Pemrosesan Sekuensial

Contoh 1. Menghitung Nilai Rata-Rata

- Diketahui file MHS.dat yang digunakan untuk menyimpan data NIM dan nilai mahasiswa di suatu mata kuliah
- Mark file MHS.dat adalah: **NIM = “999999999”** dan **nilai = 99**
- Buatlah program yang digunakan untuk membaca data dalam MHS.dat dan menghasilkan nilai rata-rata dari semua mahasiswa
- Jika file kosong, tuliskan pesan “Arsip kosong”.

Contoh 1. Menghitung Nilai Rata-Rata (1)

```
Program NilaiRataRata
{ Membaca data dalam MHS.dat dan menghasilkan nilai rata-rata
  dari semua mahasiswa. MHS.dat diasumsikan sudah isi, minimum
  berisi record <'99999999',99> (mark). }
```

KAMUS

```
type rekamanMHS : <NIM : string, nilai : integer>
  ArsipMhs : SEQFILE of
    (*) RekMhs : rekamanMHS
    (1) <"99999999",99>
  SumNil : integer { jumlah nilai }
  JumMHS : integer { jumlah mahasiswa }
```

ALGORITMA

```
assign (ArsipMHS, "MHS.dat")
open (ArsipMHS, RekMHS)

... { next slide }

close (ArsipMHS)
```

Contoh 1. Menghitung Nilai Rata-Rata (2)

ALGORITMA

```
assign (ArsipMHS, "MHS.dat")  
open (ArsipMHS, RekMHS) { First-Elmt }  
if (RekMHS.NIM = "99999999") and (RekMHS.nilai = 99) then  
    output ("Arsip kosong")  
  
else { File tidak kosong }  
    SumNil  $\leftarrow$  0; JumMhs  $\leftarrow$  0 { Inisialisasi }  
    repeat  
        SumNil  $\leftarrow$  SumNil + RekMHS.nilai  
        JumMHS  $\leftarrow$  JumMHS + 1  
        read (ArsipMHS, RekMHS)  
    until ((RekMHS.NIM="99999999") and (RekMHS.nilai=99))  
    output("Rata-rata = ", (SumNil/JumMHS))  
  
close (ArsipMHS)
```

Contoh 1. Menghitung Nilai Rata-Rata (2)

ALGORITMA

```

assign (ArsipMHS, "MHS.dat")
open (ArsipMHS, RekMHS) { First-Elmt }
if (RekMHS.NIM = "99999999") and (RekMHS.nilai = 99) then
    output ("Arsip kosong")
else { File tidak kosong }
    SumNil  $\leftarrow$  0; JumMhs  $\leftarrow$  0 { Inisialisasi }
    repeat
        SumNil  $\leftarrow$  SumNil + RekMHS.nilai
        JumMHS  $\leftarrow$  JumMHS + 1
        read (ArsipMHS, RekMHS)
    until ((RekMHS.NIM="99999999") and (RekMHS.nilai=99))
    output("Rata-rata = ", (SumNil/JumMHS))
close (ArsipMHS)
  
```

Pemeriksaan
mark, dapat
diganti fungsi EOP

Pemrosesan Sekuensial

Contoh 2. Menyalin Isi File ke Array

- Diketahui file MHS.dat yang digunakan untuk menyimpan data NIM dan nilai mahasiswa di suatu mata kuliah
- Buatlah program untuk memindahkan isi file MHS.dat ke sebuah array of data mahasiswa

Contoh 2. Menyalin Isi File ke Array (1)

Program SalinKeArray

{ Membaca data dalam MHS.dat dan menyalin isinya ke dalam sebuah array. MHS.dat diasumsikan sudah ada, isi/kosong. }

KAMUS

type rekamanMHS : <NIM : string, nilai : integer>

ArsipMhs : SEQFILE of
 (*) RekMHS : rekamanMHS
 {1} <“99999999”,99>

TabelMHS : array [1..100] of rekamanMHS

JumMHS : integer

function EOP (rek : rekamanMHS) → boolean
 { menghasilkan true jika pembacaan rek = Mark <“99999999”,99> }

ALGORITMA

assign (ArsipMHS, “MHS.dat”)

open (ArsipMHS, RekMHS)

... { next slide }

close (ArsipMHS)

Contoh 2. Menyalin Isi File ke Array (2)

ALGORITMA

```
assign (ArsipMHS, "MHS.dat")  
open (ArsipMHS, RekMHS)
```

```
if (EOP(RekMHS)) then  
    output("Arsip kosong")  
else { not EOP, File tidak kosong }
```

```
JumMhs  $\leftarrow$  1
```

```
repeat
```

```
    TabelMHSJumMHS.NIM  $\leftarrow$  RekMHS.NIM
```

```
    TabelMHSJumMHS.nilai  $\leftarrow$  RekMHS.nilai
```

```
    JumMHS  $\leftarrow$  JumMHS + 1
```

```
    read (ArsipMHS, RekMHS)
```

```
    until (JumMHS > 100) or (EOP(RekMHS))
```

```
    JumMHS  $\leftarrow$  JumMHS - 1 { penyesuaian jumlah mahasiswa }
```

```
close (ArsipMHS)
```

Pemrosesan Sekuensial

Contoh 3. Menyalin Isi Array ke File

- TabelMHS merupakan sebuah array digunakan untuk menyimpan data NIM dan nilai mahasiswa di suatu mata kuliah.
- Diasumsikan TabelMHS sudah diisi. TabelMHS mungkin kosong (JumMHS = 0).
- Buatlah program untuk memindahkan isi array tersebut ke dalam file MHS.dat
- File diakhiri **mark** berupa : **NIM = "99999999"** dan **nilai = 99**
- Buat 2 versi:
 - Versi-1: gunakan if-then-else + repeat-until
 - Versi-2: gunakan while-do

Contoh 3. Menyalin Isi Array ke File – versi 1 (1)

Program SalinKeFilev1

{ Membaca data dalam TabelMHS dan menyalin isinya ke dalam file eksternal. Mark = <'99999999',99>. }

KAMUS

type rekamanMHS = <NIM : string, nilai : integer>

constant markMHS : rekamanMHS = <"99999999", 99>

ArsipMHS : SEQFILE of
 (*) RekMHS : rekamanMHS
 (1) markMHS

TabelMHS : array [1..100] of rekamanMHS

JumMHS : integer

ALGORITMA

{ diasumsikan sudah ada bagian program yang mengisi TabelMHS dan JumMHS }

... { next slide }

Contoh 3. Menyalin Isi Array ke File – versi 1 (2)

ALGORITMA

```
{ diasumsikan sudah ada bagian program yang mengisi TabelMHS  
  dan JumMHS }
```

```
assign (ArsipMHS, "MHS.dat")  
rewrite (ArsipMHS)
```

```
if (JumMHS = 0) then  
  output ("Tabel kosong")  
else { Tabel tidak kosong }
```

```
  i ← 1  
  repeat  
    RekMHS.NIM ← TabelMHSi.NIM  
    RekMHS.nilai ← TabelMHSi.nilai  
    write(ArsipMHS, RekMHS)  
    i ← i + 1  
  until (i > JumMHS)
```

```
write (ArsipMHS, markMHS) { tulis mark }  
close (ArsipMHS)
```

Contoh 3. Menyalin Isi Array ke File – versi 2 (1)

```
Program SalinKeFilev2
{ Membaca data dalam TabelMHS dan menyalin isinya ke dalam file
eksternal. }
```

KAMUS

```
type rekamanMHS : <NIM : string, nilai : integer>
```

```
constant markMHS : rekamanMHS = <“99999999”,99>
```

```
ArsipMHS : SEQFILE of
            (*) RekMHS : rekamanMHS
            (1) markMHS
```

```
TabelMHS : array [1..100] of rekamanMHS
```

```
JumMHS : integer
```

ALGORITMA

```
{ diasumsikan sudah ada bagian program yang mengisi TabelMHS
  dan JumMHS }
```

```
assign (ArsipMHS, “MHS.dat”)
rewrite (ArsipMHS)
```

```
... { next slide }
```

```
close (ArsipMHS)
```

Contoh 3. Menyalin Isi Array ke File – versi 2 (2)

ALGORITMA

```
{ diasumsikan sudah ada bagian program yang mengisi TabelMHS
  dan JumMHS }
assign (ArsipMHS, "MHS.dat")
rewrite (ArsipMHS)

i ← 1
while (i ≤ JumMHS) do
    RekMHS.NIM ← TabelMHSi.NIM
    RekMHS.nilai ← TabelMHSi.nilai
    write(ArsipMHS, RekMHS)
    i ← i + 1
{ i > JumMHS }
write (ArsipMHS, markMHS) { tulis mark }

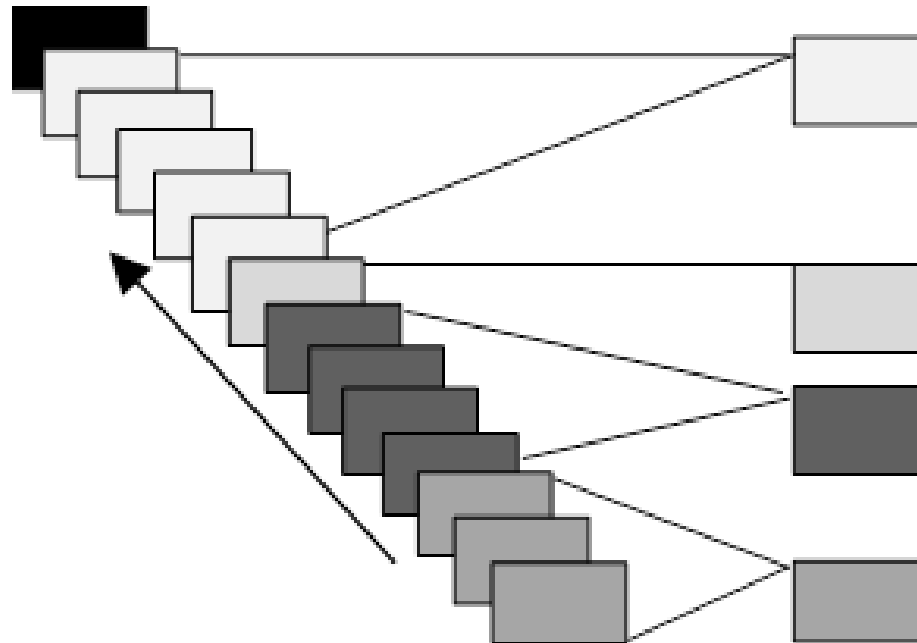
close (ArsipMHS)
```

SKEMA KONSOLIDASI FILE

Konsolidasi

- Didefinisikan sebuah *sequential file* yang terurut, arsip tersebut mengandung **kelompok-kelompok data** dengan **kunci sama** yang harus diproses sebagai satu kesatuan
- Dua model arsip semacam ini:
 - **Tanpa separator**, artinya kita mengenali adanya kelompok yang lain karena kunci berubah
 - **Dengan separator**, artinya ada rekaman tertentu yang memisahkan satu kelompok dan kelompok lainnya. Separator ini boleh satu rekaman atau lebih dari satu rekaman.

Tanpa separator



Skema Konsolidasi v. Tanpa Separator (1)

SKEMA KONSOLIDASI Tanpa Separator

```
{ Input : sebuah arsip sequential, terurut }
{ Proses : Mengelompokkan setiap kategori dan memrosesnya}
{ Output : Sesuai hasil proses }
```

KAMUS

```
type keytype : ... { keytype adalah suatu type dari kunci rekaman }
type valtype : ... { valtype adalah type dari harga rekaman }
type rekaman : < KeyIn : keytype, { kunci }
                  ValIn : valtype > { harga lain yang direkam }
constant mark : rekaman = <...,...> {akhir arsip ditandai oleh mark}
```

```
ArsipIn : SEQFILE of
          (*) RekIn : rekaman
          (1) mark
```

```
Current_Categ : keytype { Kategori yang sedang diproses }
```

```
{ Proses-Proses }
function EOP (rek : rekaman) → boolean { true jika rek = mark }
procedure Inisialisasi_Seluruh_Categ { Inisialisasi global }
procedure Terminasi_Seluruh_Categ { Terminasi global }
procedure Kasus_Kosong { Penanganan kasus kosong }
procedure Init_Categ { Inisialisasi kategori }
procedure Proses_Current_Categ { Proses sebuah elemen dalam
                                1 kategori }
procedure Terminasi_Categ { Terminasi sebuah kategori }
```

Skema Konsolidasi v. Tanpa Separator (2) v1 – tanpa penanganan kasus kosong

ALGORITMA

assign (ArsipIn, ...)

open (ArsipIn, RekIn)

Inisialisasi_Seluruh_Categ

while (not(*EOP*(*RekIn*))) do

{ **Proses satu kategori** }

Init_Categ

Current_Categ ← *RekIn.KeyIn*

repeat

Proses_Current_Categ

read (*ArsipIn*, *RekIn*)

until (*Current_Categ* ≠ *RekIn.KeyIn*)

{ *Current_Categ* ≠ *RekIn.KeyIn*,

RekIn.KeyIn adalah elemen pertama dari *Next_Categ* }

Terminasi_Categ

{ *EOP*(*RekIn*) }

Terminasi_Seluruh_Categ

close(*ArsipIn*)

Skema Konsolidasi v. Tanpa Separator (3) v2 – dengan penanganan kasus kosong

ALGORITMA

```

assign (ArsipIn, ...)
open (ArsipIn, RekIn)

  if (EOP(RekIn)) then
    Kasus_Kosong
  else { ArsipIn tidak kosong }
    Inisialisasi_Seluruh_Categ;
    repeat
      { Proses satu kategori }
      Init_Categ
      Current_Categ ← RekIn.KeyIn
      repeat
        Proses_Current_Categ
        read (ArsipIn, RekIn)
      until (Current_Categ ≠ RekIn.KeyIn)
      { Current_Categ ≠ RekIn.KeyIn,
        RekIn.KeyIn adalah elemen pertama dari Next_Categ }
      Terminasi_Categ
    until (EOP(RekIn))
    Terminasi_Seluruh_Categ

close(ArsipIn)

```

Contoh Skema Konsolidasi Tanpa Separator (1)

- Diketahui sebuah arsip nilai mahasiswa, satu mahasiswa dapat mempunyai beberapa buah nilai (karena dalam satu semester mengambil beberapa matakuliah dan setiap mahasiswa tidak sama matakuliahnya).
- Buat algoritma untuk menghitung nilai rata-rata setiap mahasiswa, dan membuat daftar nilai sederhana, yaitu menuliskan NIM dan nilai rata-rata setiap mahasiswa.

Contoh Skema Konsolidasi Tanpa Separator (2)

Program NilaiMahasiswa

```
{ Input : sebuah arsip sekuensial berisi NIM dan nilai mahasiswa }
{ Proses : proses setiap kategori adalah menghitung nilai rata-rata
setiap mahasiswa }
{ Output : NIM dan Nilai rata-rata setiap mahasiswa }
{ Tanpa penanganan kasus kosong }
```

KAMUS

```
type keytype : string { keytype adalah suatu type dari kunci rekaman }
type valtype : integer { valtype adalah type dari harga rekaman }
type rekaman : < NIM : keytype, { kunci }
                 nilai : valtype > { nilai ujian }
```

```
constant mark : rekaman = <"99999999",0> { mark arsip }
```

```
ArsipMhs : SEQFILE of
    (*) RekMhs : rekaman
    (1) mark
```

```
Current_NIM : keytype { NIM mahasiswa yang sedang diproses }
SumNil      : integer { jumlah nilai seluruh matakuliah seorg mhs }
NKuliah     : integer { jumlah matakuliah seorg mhs }
```


Contoh Skema Konsolidasi Tanpa Separator (3)

ALGORITMA

```

assign (ArsipMhs, "dataMHS.dat")
open (ArsipMhs, RekMhs)

{ Inisialisasi : tidak ada }
while (RekMhs.NIM ≠ mark.NIM) and (RekMhs.nilai ≠ mark.nilai) do

    { Proses satu kategori = 1 NIM }
    SumNil ← 0; NKuliah ← 0 { Inisialisasi kategori }
    Current_NIM ← RekMhs.NIM
    repeat
        SumNil ← SumNil + RekMhs.nilai; NKuliah ← NKuliah + 1
        read (ArsipMhs, RekMhs)
    until (Current_NIM ≠ RekMhs.NIM)
    { Current_NIM ≠ RekMhs.NIM,
      RekMhs.NIM adalah elemen pertama dari Next_NIM }
    output(Current_NIM, " ", (SumNil/NKuliah))

{ EOP(RekMhs) }
{ Terminasi : tidak ada }
close(ArsipIn)

```

Skema Konsolidasi v. Dengan Separator (1)

```
{ SKEMA KONSOLIDASI Dengan Separator }
{ Input : sebuah arsip sequential, terurut }
{ Proses : Mengelompokkan setiap kategori dan memrosesnya }
{ Output : Sesuai hasil proses }
```

KAMUS

```
type keytype : ... { keytype adalah suatu type dari kunci rekaman }
type valtype : ... { valtype adalah type dari harga rekaman }
type rekaman : < KeyIn : keytype,      { kunci }
                  ValIn : valtype >    { harga lain yang direkam }
```

```
constant mark : rekaman = <...,> { akhir arsip ditandai oleh mark }
```

```
ArsipIn : SEQFILE of
  (*) RekIn : rekaman
  (1) mark
```

```
{ Proses-Proses }
function EOP (rek : rekaman) → boolean { true jika rek = mark }
function IsSeparator (K : keytype) : boolean
{ true jika K adalah separator antar kategori }
procedure Inisialisasi_Seluruh_Categ { Inisialisasi global }
procedure Terminasi_Seluruh_Categ { Terminasi global }
procedure Kasus_Kosong { Penanganan kasus kosong }
procedure Init_Categ { Inisialisasi kategori }
procedure Proses_Current_Categ { Proses sebuah elemen dalam
                                1 kategori }
procedure Terminasi_Categ { Terminasi sebuah kategori }
```

ALGORITMA

```

assign (ArsipIn, ...)
open (ArsipIn, RekIn)
{ Skip separator, bisa lebih dari 1 }
while (not(EOP(RekIn))) and (IsSeparator(RekIn.KeyIn)) do
    read (ArsipIn, RekIn)
    { EOP(RekIn) or not(IsSeparator(RekIn.KeyIn)) }
    { RekIn.KeyIn bukan separator,
      RekIn.KeyIn : elemen pertama dari Next_Categ atau EOP }

if (EOP(RekIn)) then
    Kasus_Kosong
else { ArsipIn tidak kosong }
    Inisialisasi_Seluruh_Categ
    repeat
        Init_Categ
        while (not(EOP(RekIn))) and (not(IsSeparator(RekIn.KeyIn))) do
            { Proses 1 kategori }
            Proses_Current_Categ
            read (ArsipIn, RekIn)
            { EOP(RekIn) or IsSeparator(RekIn.KeyIn) }
            Terminasi_Categ

            { Skip separator, bisa lebih dari 1 }
            while (not(EOP(RekIn))) and (IsSeparator(RekIn.KeyIn)) do
                read (ArsipIn, RekIn)
            { EOP(RekIn) or not(IsSeparator(RekIn.KeyIn)) }
            { RekIn.KeyIn bukan separator,
              RekIn.KeyIn : elemen pertama dari Next_Categ atau EOP }

        until (EOP(RekIn))
        Terminasi_Seluruh_Categ
    close(ArsipIn)

```

Skema
Konsolidasi v.
Dengan
Separator (2)

Contoh Skema Konsolidasi Dengan Separator (1)

- Diberikan sebuah arsip teks yang dapat diakses *sequential* huruf per huruf.
- Hendak dihitung panjang kata maksimum dalam teks tersebut.
- Diandaikan bahwa teks hanya mengandung huruf dan "blank".
- Kata adalah sekumpulan huruf yang dipisahkan oleh satu atau beberapa blank.

Contoh Skema Konsolidasi Dengan Separator (2)

Program KataTerpanjang

```
{ Input : sebuah arsip sequential, mewakili sebuah teks }
{ Proses : Menghitung panjang kata maksimum dalam teks }
{ Output : Panjang kata maksimum }
```

KAMUS

```
type keytype : character { type dari kunci rekaman }
type rekaman : keytype
constant mark : rekaman = '.', { akhir arsip ditandai oleh mark }
constant blank : rekaman = ' ', { separator kata }

ArsipIn : SEQFILE of
    (*) CC : character { file of rekaman }
    (1) mark
PanjangKata : integer { Panjang kata yang sedang diproses }
MaxLength : integer { Panjang kata maksimum }
```

ALGORITMA

```

assign (ArsipIn, "datakata.txt")
open (ArsipIn, CC)
{ Skip separator, bisa lebih dari 1 }
while (CC ≠ mark) and (CC = blank) do
    read (ArsipIn, CC)
{ CC = mark or CC ≠ blank }
{ CC bukan separator,
  CC : elemen pertama dari sebuah kata atau mark }
if (CC = mark) then
    output ("Arsip Kosong")
else { ArsipIn tidak kosong }
    MaxLength ← 0 { Asumsi: kata minimum terdiri atas 1 huruf }
    repeat
        PanjangKata ← 0
        while (CC ≠ mark) and (CC ≠ blank) do
            { Proses 1 huruf }
            PanjangKata ← PanjangKata + 1
            read (ArsipIn, CC)
        { CC = mark or CC = blank }
        if (MaxLength < PanjangKata) then MaxLength ← PanjangKata

        { Skip separator, bisa lebih dari 1 }
        while (CC ≠ mark) and (CC = blank) do
            read (ArsipIn, CC)
        { CC = mark or CC ≠ blank }
        { CC bukan separator,
          CC : elemen pertama dari sebuah kata atau mark }

    until (CC = mark)
    output ("Panjang kata maksimum = ", MaxLength)
close (ArsipIn)

```

Skema
Konsolidasi
v. Dengan
Separator (2)

Skema Konsolidasi pada Array

- Ide skema konsolidasi [dengan/tanpa] separator dapat diberlakukan untuk melakukan konsolidasi pada struktur data koleksi spt. **array**
- Contoh persoalan: diketahui data array of nilaiMhs dengan nilaiMhs adalah type bentukan sbb.

type nilaiMhs : <NIM:string, Nilai:integer>

NIM	Nilai
13215001	90
13215001	85
13515010	88
13515010	93
13515010	80
13515010	71

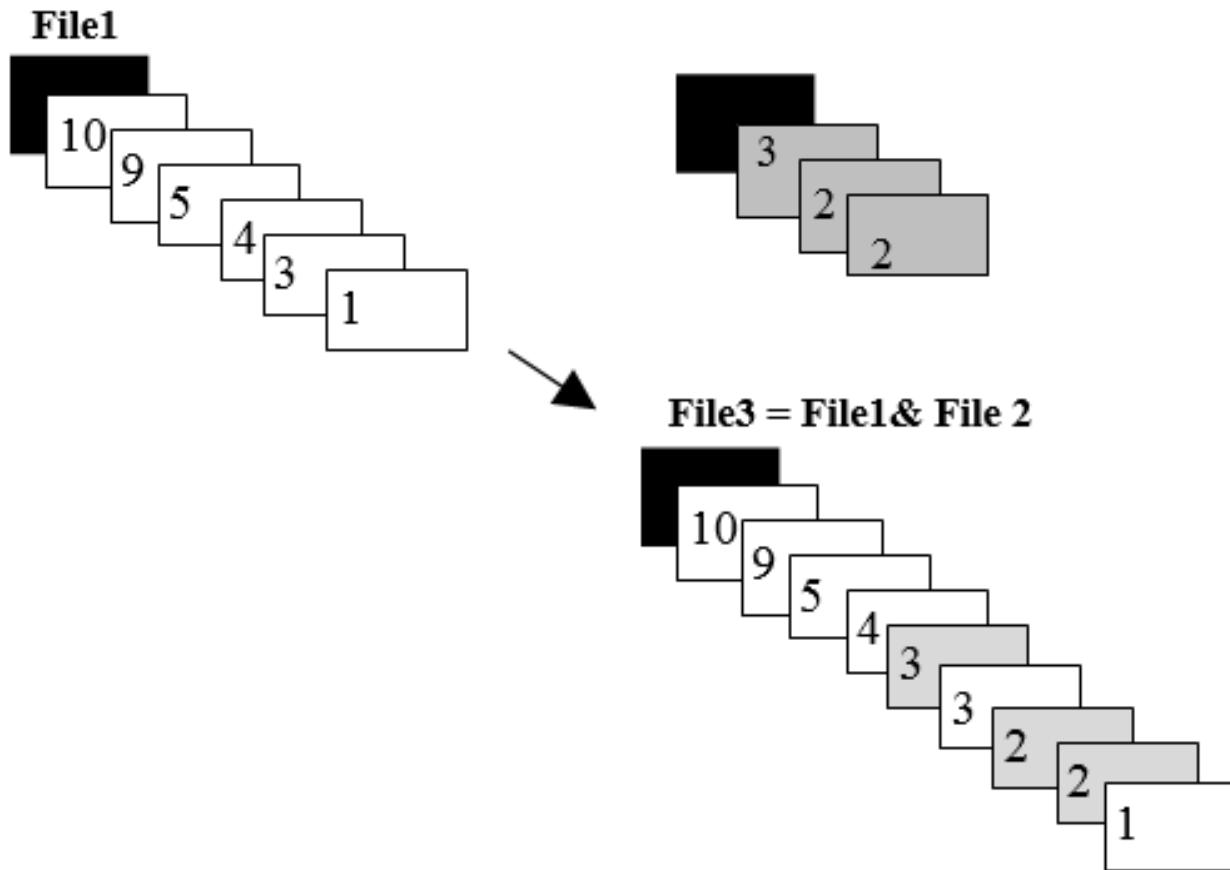
- Dengan memanfaatkan ide skema konsolidasi, tuliskan daftar NIM dan nilai rata-rata untuk semua nilainya.
 - Apa yang menjadi **mark**?
- Sebagai contoh, array di samping akan mencetak:
 13215001 88
 13515010 83

SKEMA MERGING 2 FILE

Merging (1)

- **Merging:** penggabungan dua buah arsip.
- Paling sederhana adalah jika arsip yang pertama "dikonkatensi" ke arsip kedua
 - artinya data dari arsip ke dua ditambahkan setelah rekaman terakhir arsip pertama dan membentuk arsip yang baru
- Tak dapat dipakai **jika kedua arsip sudah terurut**, dan dikehendaki sebuah **arsip hasil yang tetap terurut**.
- Akan dibahas skema untuk **penggabungan dua buah arsip terurut** menjadi **sebuah arsip yang terurut**

Merging (2)



Skema Merging v1 – AND (1)

```
{ SKEMA Merging versi dengan AND }
{ Input  : Dua arsip sequential, terurut, sejenis }
{ Proses : Menggabung kedua arsip menjadi sebuah arsip yg terurut
            VERSI AND }
{ Output : Sequential file baru yang terurut }
```

KAMUS

```
type keytype : ... { keytype adalah suatu type dari kunci rekaman }
type valtype : ... { valtype adalah type dari harga rekaman }
type rekaman : < KeyIn : keytype, { kunci }
                  ValIn : valtype > { harga lain yang direkam }
constant mark : rekaman = <...,...> { akhir arsip ditandai oleh mark }
```

```
ArsipIn1 : SEQFILE of { input, terurut menurut kunci }
            (*) RekIn1 : rekaman
            (1) mark
ArsipIn2 : SEQFILE of { input, terurut menurut kunci }
            (*) RekIn2 : rekaman
            (1) mark
ArsipOut : SEQFILE of { input, terurut menurut kunci }
            (*) RekOut : rekaman
            (1) mark
```

ALGORITMA

```

assign (ArsipIn1, ...)
open (ArsipIn1, RekIn1)
assign (ArsipIn2, ...)
open (ArsipIn2, RekIn2)
assign (ArsipOut, ...)
rewrite (ArsipOut)

```

```

while (RekIn1.KeyIn  $\neq$  mark.KeyIn) and (RekIn2.KeyIn  $\neq$  mark.KeyIn) do
    if (RekIn1.KeyIn  $\leq$  RekIn2.KeyIn) then
        write (ArsipOut, RekIn1)
        read (ArsipIn1, RekIn1)
    else { RekIn1.KeyIn  $>$  RekIn2.KeyIn }
        write (ArsipOut, RekIn2)
        read (ArsipIn2, RekIn2)
    { RekIn1 = mark or RekIn2 = mark }

```

```

while (RekIn1.KeyIn  $\neq$  mark.KeyIn) do
    write (ArsipOut, RekIn1)
    read (ArsipIn1, RekIn1)
    { Akhir ArsipIn1, RekIn1 = mark }
while (RekIn2.KeyIn  $\neq$  mark.KeyIn) do
    write (ArsipOut, RekIn2)
    read (ArsipIn2, RekIn2)
    { Akhir ArsipIn2, RekIn2 = mark }

```

```

close (ArsipIn1)
close (ArsipIn2)
close (ArsipOut)

```

Skema
Merging v1 –
AND (2)

Skema Merging v2 – OR (1)

```
{ SKEMA Merging versi dengan OR }
{ Input  : Dua arsip sequential, terurut, sejenis }
{ Proses : Menggabung kedua arsip menjadi sebuah arsip yg terurut
          VERSI OR }
{ Output : Sequential file baru yang terurut }
```

KAMUS

```
type keytype : ... { keytype adalah suatu type dari kunci rekaman }
type valtype : ... { valtype adalah type dari harga rekaman }
type rekaman : < KeyIn : keytype, { kunci }
                ValIn : valtype > { harga lain yang direkam }
constant mark : rekaman = <...,...> { akhir arsip ditandai oleh mark }
```

```
ArsipIn1 : SEQFILE of           { input, terurut menurut kunci }
            (*) RekIn1 : rekaman
            (1) mark
ArsipIn2 : SEQFILE of           { input, terurut menurut kunci }
            (*) RekIn2 : rekaman
            (1) mark
ArsipOut  : SEQFILE of           { output, terurut menurut kunci }
            (*) RekOut : rekaman
            (1) mark
```

Skema Merging v2 – OR (2)

ALGORITMA

```
assign (ArsipIn1, ...)  
open (ArsipIn1, RekIn1)  
assign (ArsipIn2, ...)  
open (ArsipIn2, RekIn2)  
assign (ArsipOut, ...)  
rewrite (ArsipOut)  
  
while (RekIn1.KeyIn  $\neq$  mark.KeyIn) or (RekIn2.KeyIn  $\neq$  mark.KeyIn) do  
    if (RekIn1.KeyIn  $\leq$  RekIn2.KeyIn) then  
        write (ArsipOut, RekIn1)  
        read (ArsipIn1, RekIn1)  
    else { RekIn1.KeyIn  $>$  RekIn2.KeyIn }  
        write (ArsipOut, RekIn2)  
        read (ArsipIn2, RekIn2)  
    { RekIn1 = mark and RekIn2 = mark }  
  
close (ArsipIn1)  
close (ArsipIn2)  
close (ArsipOut)
```

Skema Merging v2 – OR (3)

- Hanya benar jika **mark** adalah suatu *nilai khusus yang dipakai untuk “menahan” maju ke rekaman berikutnya* sehingga arsip yang belum habis akan terproses sampai habis
- Versi ini tidak dapat digunakan secara umum karena kekhususan nilai mark tersebut
- Bahkan tak dapat digunakan sama sekali jika mark adalah suatu nilai EOP yang ditentukan oleh sistem

Skema Lain

- *Update dengan transaction file*
- *Splitting*
- Tidak dibahas, silakan dipelajari sendiri → lihat “Diktat Dasar Pemrograman Bagian: Pemrograman Prosedural”

Latihan

Latihan-1

- Buatlah program yang membaca file sekuensial *rekaman.dat* berisi data bertipe **rekamanMHS** sbb:

type rekamanMHS : < NIM : string, nilai : integer>

- Selanjutnya, tuliskan ke file sekuensial lain *rekaman1.dat* dengan spesifikasi hanya menulis rekaman yang berisi komponen nilai ≥ 80 .
- File sekuensial diakhiri mark berupa : NIM = "99999999" dan nilai = 99.

Latihan-2

- Buatlah program yang membaca file sekuensial *rekaman.dat* yang berisi data bertipe **rekamanMHS** (spt. pada slide Latihan-1)
- Selanjutnya, tuliskan hanya bagian *nilai*-nya saja ke file sekuensial *nilai.dat* lain (berisi data bertipe integer).
- File *nilai.dat* diakhiri mark, yaitu nilai 999.

Latihan-3 (1)

- Diketahui file datapenjualan.dat yang berisi data bertipe penjualan sbb.

```
type penjualan : <Kategori : string,  
                  KdBarang : string,  
                  Jumlah   : integer>
```

- Data pada file terurut berdasarkan Kategori
- Buatlah program untuk mendapatkan total barang terjual per Kategori. Gunakan skema konsolidasi tanpa separator.
-

Latihan-3 (2)

- Asumsikan terdefinisi sebuah function bernama EOP untuk memeriksa apakah rekaman adalah mark atau bukan

function EOP (rek : penjualan) → boolean

{ Menghasilkan true jika rek adalah mark, asumsi mark terdefinisi }

- Contoh isi file:

- Akan tercetak di layar: buku – 91
atk – 332

```
<buku,AX210,40>  
<buku,RT234,51>  
<atk,RE445,101>  
<atk,TY309,231>  
< mark >
```

Latihan-4

- Diketahui 2 buah file data mahasiswa misalnya *rekmhs1.dat* dan *rekmhs2.dat* yang menyimpan data bertipe *rekamanMHS* (spt. pada slide Latihan-1).
- Data pada kedua file terurut menurut *NIM*.
- Tuliskan hasil penggabungan kedua file ke dalam sebuah file bernama *rekmhs.dat*.
- Gunakan skema *merging* versi AND.

Latihan-5

- Diketahui data array of nilaiMhs dengan nilaiMhs adalah type bentukan sbb. (array terurut berdasarkan NIM):

type nilaiMhs : <NIM:string, Nilai:integer>

NIM	Nilai
13215001	90
13215001	85
13515010	88
13515010	93
13515010	80
13515010	71

- Dengan memanfaatkan ide skema konsolidasi, tuliskan daftar NIM dan nilai rata-rata untuk semua nilainya.
 - Apa yang menjadi **mark**?
- Sebagai contoh, array di samping akan mencetak:

13215001 88
13515010 83

Selamat Belajar