Fakultas Informatika
School of Computing
Telkom University

KUG1C3
Dasar Algoritma dan Pemrograman

Sequential File

-RSM-

Definisi

- Sequential File (arsip sekuensial) adalah sekumpulan record (rekaman) yang disimpan dalam media penyimpanan sekunder komputer, yang dapat diakses secara berurutan mulai dari record pertama sampai dengan record terakhir, record per record secara searah.
- Record terakhir adalah record fiktif yang menandai akhir dari arsip. Pada beberapa implementasi, record fiktif ini disebut sebagai EOF (end of file).

Struktur Arsip Sekuensial

- Tiap record berisi : tipe data dasar atau tipe data terstruktur (bentukan)
- Tiap record memiliki struktur yang sama !!
- Elemen dari record disebut Field
- Ada sebuah atau lebih field yang disebut Key karena kekhususannya dalam proses
- Jika Key dari tiap record memiliki nilai yang tidak sama (unik), maka Key disebut Primary Key

Pengolahan Arsip Sekuensial

- Tiap record dapat diakses dan dibaca menurut urutannya dengan pemanggilan primitif akses yang tersedia.
- Perekaman record pada arsip sekuensial dapat dilakukan melalui primitif penulisan.
- Arsip sekuensial hanya dapat dilakukan pada salah satu modus operasi : diakses / dibaca atau ditulis. Tidak pernah keduanya dilakukan pada sebuah arsip disaat yang bersamaan.

Pendefinisian Arsip Sekuensial

```
Kamus
  type rekaman : <……>
  {sebuah type terdefinisi untuk
  setiap record}
  NamaArsip : SEQFILE of
     (*) <nama_rek> : rekaman
     (1) <mark>
```

Dengan catatan bahwa (*) mungkin kosong,1 rekaman atau lebih

Primitif-Primitif Akses untuk Arsip Sekuensial

```
procedure ASSIGN (Input NamaArsip, NamaFisik)
{Arsip sekuensial yang namanya dikenal dalam program
sebagai NamaArsip, secara fisik diberi nama NamaFisik
IS     : sembarang
FS     : Arsip dengan NamaArsip pada program siap dipakai
}

procedure OPEN (Input NamaArsip, <rekaman>)
{Arsip Sekuensial siap dibaca rekaman pertama yang
informasinya ada pada <rekaman> dapat diakses
IS     : sembarang
FS     : Informasi pada rekaman siap diakses, dengan
mengacu pada <rekaman>
}
```

Primitif-Primitif Akses untuk Arsip Sekuensial

```
procedure READ (Input NamaArsip, <rekaman>)
{Rekaman sesudah rekaman yang sedang 'current', yang dapat diakses
IS     : <rekaman> bukan merupakan mark (EOF), disebut sebagai current_rekaman
FS     : Arsip dimajukan satu rekaman, <rekaman> berisi informasi yang disimpan pada rekaman sesudah current_rekaman. Mungkin <rekaman> yang baru adalah mark (EOF)
}

procedure CLOSE (Input NamaArsip)
{Arsip sekuensial 'ditutup', tidak dapat diakses maupun ditulis
IS     : sembarang
FS     : Arsip tidak dapat diproses lagi
}
```

Primitif-Primitif Perekaman untuk Arsip Sekuensial

```
procedure Rewrite (Input/Output NamaArsip)
{Arsip sekuensial siap untuk direkam
IS     : sembarang
FS     : Arsip sekuensial yang bernama NamaArsip siap untuk direkam pada posisi pertamanya
}

procedure Write (Input/Output NamaArsip, <rekaman>)
{Data pada <rekaman> direkam pada posisi aktual arsip. Kemudian posisi dimajukan satu.
IS     : arsip sekuensial berada pada posisi yang telah siap menerima rekaman, <rekaman> bukan merupakan mark (EOF)
FS     : <rekaman> direkam pada posisi yang telah disiapkan, arsip diajukan satu posisi. Jika <rekaman> yang diisikan ke arsip adalah elemen fiktif yang dimaksudkan sebagai mark,  maka arsip tak dapat lagi direkami
}
```

Contoh 1

- Sebuah arsip sekuensial berisi data mahasiswa yang setiap rekamannya memuat data NIM, nama, dan nilai akhir mahasiswa.
  - Definisi arsip sekuensial

```
type rekaman : <    NIM : integer,
                    Nama : string,
                    Nilai : integer [0..100] >
ArsipMhs : SEQFILE of
(*) RekMhs : rekaman
(1) <9999999,'',0>
```

Domain setiap rekaman : sesuai dg domain masing-masing rekaman

Konstanta : sebuah rekaman misal <7473001, 'Juliete', 95>

Contoh 1

- Cara Akses Rekaman Pertama

  OPEN (ArsipMhs, RekMhs)
- Cara Akses

  READ (ArsipMhs, RekMhs)
- Cara menyiapkan untuk direkam

  REWRITE (ArsipMhs)
- Cara mengisi

  WRITE (ArsipMhs, RekMhs)

  WRITE (ArsipMhs, <7473002,'Davy',96>)

  WRITE (ArsipMhs, Rek1) {Rek1 bertipe rekaman}
- Cara mengisi akhir rekaman

  WRITE (ArsipMhs, <9999999,'',0>)

Contoh 2

- Sebuah arsip sekuensial berisi teks, maka setiap rekamannya adalah satu karakter. Misalnya MARK adalah '#‘.
  - Definisi arsip sekuensial

```
type rekaman : character
Dokumen : SEQFILE of
  (*) CC : rekaman
  (1) <‘#’>
```

Domain setiap rekaman : character

Konstanta : sebuah rekaman, misal <’J’>, <‘0’>, <‘#’>

Contoh 2

- Cara Akses Rekaman Pertama

  OPEN (Dokumen, CC)
- Cara Akses

  READ (Dokumen, CC)
- Cara menyiapkan untuk direkam

  REWRITE (Dokumen)
- Cara mengisi

  WRITE (Dokumen, CC)

  WRITE (Dokumen, <'D'>)

  WRITE (Dokumen, Kar) {Kar bertipe character}
- Cara mengisi akhir rekaman

  WRITE (Dokumen, <'#'>)

Pemrosesan Sebuah Arsip Sekuensial

- Jika setiap rekaman harus diproses dengan cara sama, pemrosesan arsip sekuensial dapat dilakukan dengan memakai skema pemrosesan sekuensial dengan mark.

  Contoh

  Dibaca sebuah arsip sekuensial bernama :

```
type rekaman : <NIM : integer, Nilai : integer [0..100]>
ArsipMhs : SEQFILE of
     (*) RekMhs : rekaman
     (1) <9999999, 0>
```

  Analisa : pemrosesan sekuensial dari elemen arsip sekuensial

  Model tanpa MARK, jika i adalah deret yang diproses, i berharga 1, 2, 3, .. N

  EOF adalah NIM = 9999999

  First-Elmt : OPEN (arsipMhs,<nrp,nilai>)

  Next-Elmt : READ (arsipMhs,<nrp,nilai>)

  Proses : membaca arsip sambil menghitung nilai rata-rata mahasiswa

Pemrosesan Sebuah Arsip Sekuensial

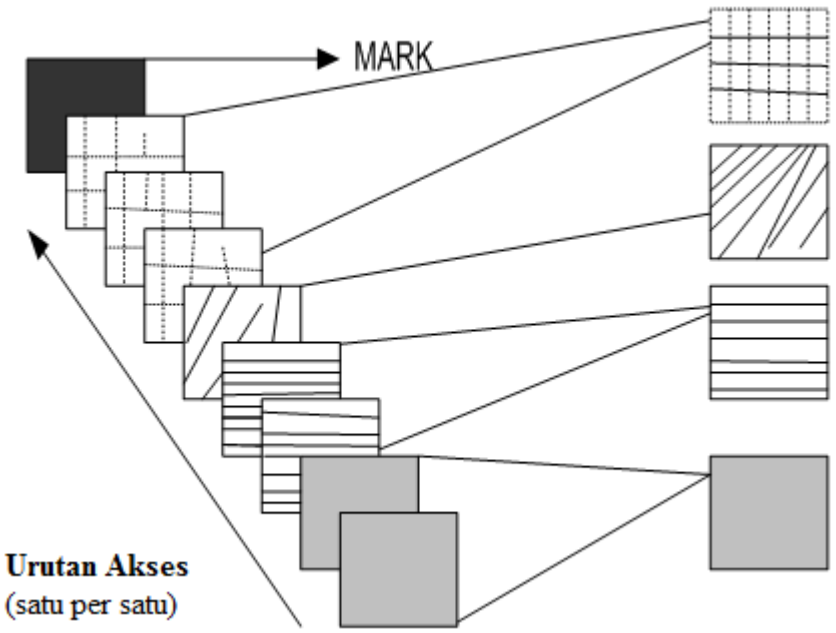
```
Program NILAIRATA_RATA
{model proses sekuensial dengan mark,
 dengan penanganan kasus kosong}:
Kamus:
    type rekaman : < NIM : integer, nilai:integer [0..100] >
   ArsipMhs : SEQFILE of
      (*) RekMhs : rekaman { setiap mahasiswa  punya 1 rekaman }
      (1) <9999999, 99>
   SumNil : integer  { jumlah nilai}
   JumMhs: integer  { jumlah mahasiswa }
Algoritma :
   OPEN(ArsipMhs, RekMhs) {First_Elmt}
   if (RekMhs.NIM =9999999) then
      output ('Arsip kosong')
   else
      SumNil ← 0 ; JumMhs ← 0   { Inisialisasi }
      repeat
        SumNil ← Sumnil + RekMhs.nilai; JumMhs ← JumMhs+1 {Proses }
        READ(ArsipMhs,RekMhs)        {Next_Elmt}
      until  (RekMhs.NIM=9999999)          {EOP}
      Output (Sum/JumMhs)                  {Terminasi}
   CLOSE (ArsipMhs)
```

Algoritma Konsolidasi

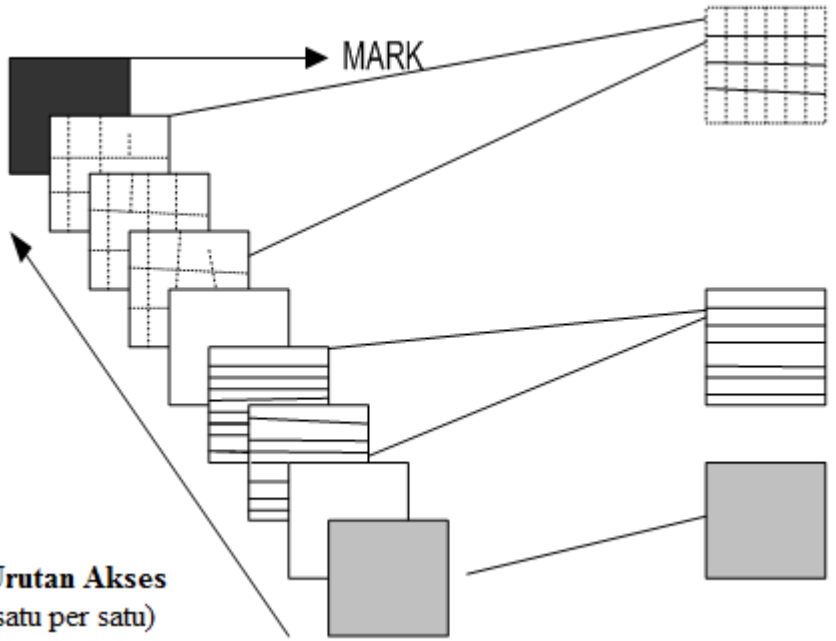
- Didefinisikan sebuah arsip sekuensial yang terurut, arsip tersebut mengandung kelompok-kelompok data dengan kunci yang sama yang harus diproses sebagai satu kesatuan.
- Ada dua model arsip semacam ini:
  - tanpa separator : kita mengenali adanya kelompok yang lain karena kunci berubah
  - dengan separator : ada rekaman tertentu yang memisahkan satu kelompok dan kelompok lainnya. Separator ini boleh satu rekaman atau lebih dari satu rekaman. Pada contoh berikut separator adalah ’kartu putih’.

Algoritma Konsolidasi

Tanpa Separator



Dengan Separator



Contoh Aplikasi 1

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

Contoh Aplikasi 1

```
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    }
Kamus
   type  Keytype : integer
   type  Valype :  integer [0..100]
   type rekaman : <  NIM: keytype,    { kunci }
                     Nilai:valtype    {nilai ujian} >
   ArsipMhs : SEQFILE of      {input, terurut menurut kunci }
      (*) RekMhs: rekaman
      (1) <9999999, 0>
   Current_NIM : integer { identifikasi kategori yg sedang diproses}
   SumNil : integer             { Jumlah nilai seluruh matakuliah seorg mhs}
   NKuliah : integer             { Jumlah matakuliah seorang mahasiswa }
Algoritma :
   {Inisialisasi : tidak ada}
   OPEN(ArsipMhs ,RekMhs)    {First_Elmt}
   while  (RekMhs.NIM ≠ 9999999)   {not EOP }  do
      {Proses satu kategori = 1 NIM }
      SumNil  ← 0            { Init_Categ}
      NKuliah  ← 0           { Init_Categ}
      Current_NIM ← RekMhs.NIM
      repeat
         SumNil  ← SumNil  + RekMhs.Nilai
         NKuliah  ← NKuliah + 1          {Proses_Current_Categ}
         READ (ArsipMhs ,RekMhs)
      until (Current_NIM ≠ RekMhs. NIM)
      { NIM ≠ Current_NIM ,
         NIM adalah elemen pertama dari Next_Categ }
       {Terminasi_Categ}
      output (Current_NIM,SumNil/NKuliah);
   CLOSE (ArsipMhs)
```

Contoh Aplikasi 2

- Idem contoh aplikasi 1, tetapi selain itu juga dikehendaki nilai rata-rata seluruh mahasiswa (jumlah nilai rata-rata setiap mahasiswa dibagi jumlah mahasiswa).

```
Program NILAIMAHASISWA2
{ dengan penanganan kasus kosong }
{ Input : sebuah arsip sekuensial berisi NIM dan nilai mahasiswa}
{ Proses :Mengelompokkan setiap kategori dan memrosesnya : menghitung
nilai rata-rata}
{ setiap mahasiswa  dan nilai rata-rata seluruh populasi mahasiswa
Output : NIM Nilai rata-rata setiap mahasiswa, dan nilai rata-rata
seluruh mahasiswa}
```

```
Kamus
   type  Keytype : integer
   type  Valype :  integer [0..100]
   type rekaman :  < NIM: keytype, {kunci }
                     Nilai:valtype { harga lain yang direkam } >
    ArsipMhs : SEQFILE of  { input, terurut menurut kunci }
      (*) RekMhs : rekaman
      (1) <9999999, 0>
   Current_NIM : integer { identifikasi kategori yg sedang diproses}
   SumNil : integer          { Jumlah nilai seluruh matakuliah seorg mhs}
   NKuliah : integer         { Jumlah matakuliah seorang mahasiswa }
   NilRata : integer         { Nilai rata-rata seorang mahasiswa }
   SumNilTot: integer    { Jumlah nilai seluruh matakuliah seorg mhs}
   NMhs : integer            { Jumlah matakuliah seluruh mahasiswa }
```

```
Algoritma :
   OPEN(ArsipMhs ,RekMhs)    {First_Elmt}
   if  (RekMhs.NIM = 9999999)    { EOP } then
      output ('Arsip kosong')
   else

      repeat
         {Proses satu kategori = 1 NIM }
         SumNil  ← 0  ; NKuliah  ← 0     { Init_Categ}
         Current_NIM ← RekMhs.NIM
         repeat
            SumNil  ← SumNil  + RekMhs.Nilai    {Proses}
            NKuliah  ← NKuliah + 1            {Proses_Current_Categ}
            READ (ArsipMhs,RekMhs)
         until (Current_NIM ≠ RekMhs.NIM)
         { NIM ≠ Current_NIM , RekMhs.NIM=elemen pertama Next_Categ }
         NilRata  ← SumNil/NKuliah
         SumNilTot  ← SumNilTot + NilRata
         NMhs ←  NMhs + 1
         output (Current_NIM,NilRata)           {Terminasi_Categ}

      until (RekMhs.NIM = 9999999)    { EOP }

      output (SumNilTot/NMhs);            { terminasi seluruh file }

   CLOSE (ArsipMhs)
```

Contoh Aplikasi 3

- Diberikan sebuah arsip teks yang dapat diakses secara sekuensial huruf per huruf.
- Hendak dihitung kata yang terpanjang dalam teks tersebut. Diandaikan bahwa teks hanya mengandung huruf dan ”blank”.
- Kata adalah sekumpulan huruf yang dipisahkan oleh satu atau beberapa blank.

```
Program KATATERPANJANG
{ Input : sebuah arsip sequential, yang mewakili sebuah teks, }
{ setiap rekaman adalah sebuah katakter}
{ Proses :Menghitung kata terpanjang dalam teks}
{ Output : Panjang kata maksimum     }
```

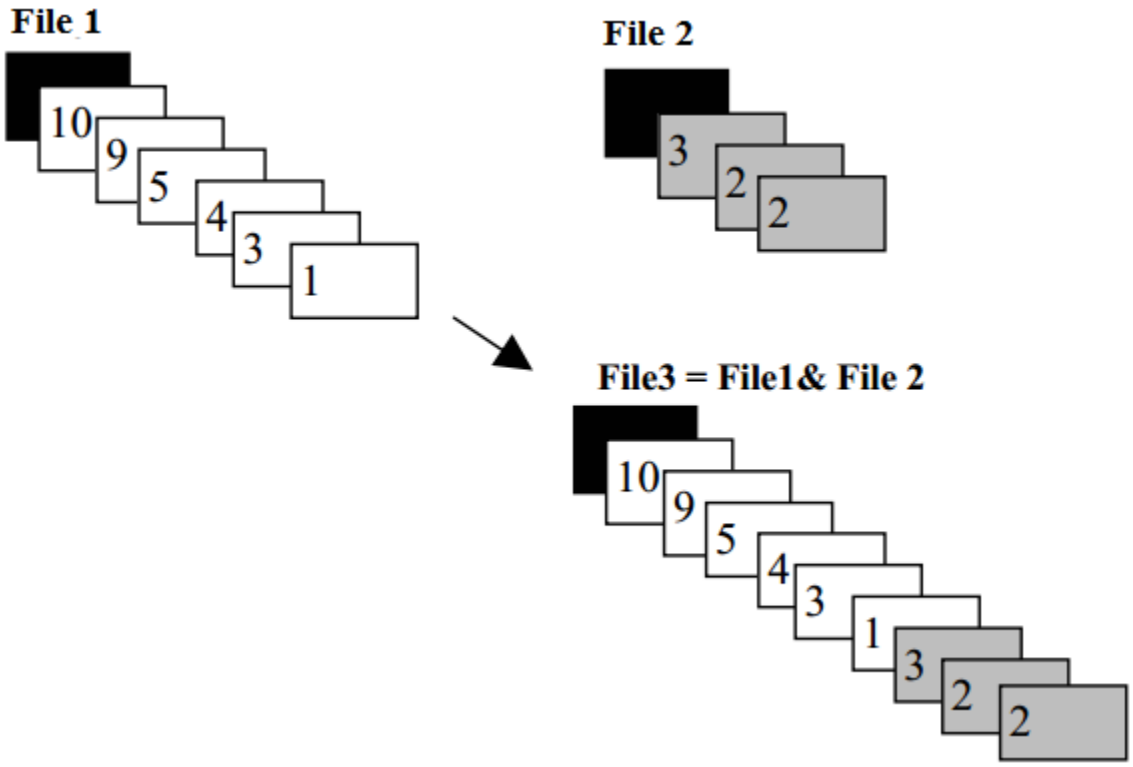
```
Kamus
   {Keytype : type dari elemen arsip yg menentukan apakah elemen tsb
separator atau bukan}
   {Valtype adalah type dari harga rekaman, di sini tidak ada }
   KeyType : character
   constant  mark :  character ='.'
   constant  blank :  character =' '
   type rekaman : Keytype
    ArsipIn : SEQFILE of {input, terurut menurut kunci }
       (*) CC: rekaman    {CC:sebuah karakter yang direkam)
       (1) <'.'>          { mark }
   PanjangKata : integer        { Panjang kata yang sedang dlm proses}
   MaxLength : integer           { Panjang kata maksimum }
   procedure  Inisialisasi
   procedure  Init_Categ        { Inisialisasi untuk satu kategori }
   procedure  Proses_Current_Categ  {Proses thd sebuah elemen kategori}
   procedure  Terminasi_Categ     {Terminasi sebuah  kategori}
   function  Separator  (K : Keytype) → boolean
   { true jika K adalah separator, di sini adalah blank : ' '}
```

```
Algoritma :
   OPEN(ArsipIn, CC)     {First_Elmt}
   if (CC=mark) then
      output ('Arsip kosong')
   else
      Maxlength  ← 0{Diandaikan kata minimum terdiri dari 1 huruf}
      repeat
         { Skip separator, jika ada}
         while   (CC ≠  mark)   and (CC  =  blank ) do
            {not EOP & Separator(Keyin) }
            READ (ArsipIn , CC)
         { CC bukan Separator , CC adalah huruf pertama dari
           sebuah kata atau Mark}
         PanjangKata  ←  0
         while  (CC ≠  mark)   and (CC ≠  blank )  do
            {not Separator(KeyIn) and not  EOP}
            PanjangKata  ← PanjangKata  + 1
            {Proses_Current_Categ}
            READ (ArsipIn , CC)
         { CC = blank   or CC = mark}
         if (MaxLength < PanjangKata)  then
                MaxLength ←  Panjang Kata            {Terminasi_Categ}
      until (CC = mark)    { EOP }
      {Terminasi proses,Maxlength=0 berarti Arsip hanya berisi blank}
      Output (MaxLength)
   CLOSE(ArsipIn)
```

Algoritma Pemrosesan Dua Arsip: Merging

- Merging adalah penggabungan dua buah arsip.
- Yang paling sederhana adalah jika arsip yang pertama dikonkatenasi ke arsip kedua (artinya data dari arsip ke dua ditambahkan setelah rekaman terakhir arsip pertama dan membentuk arsip yang baru)

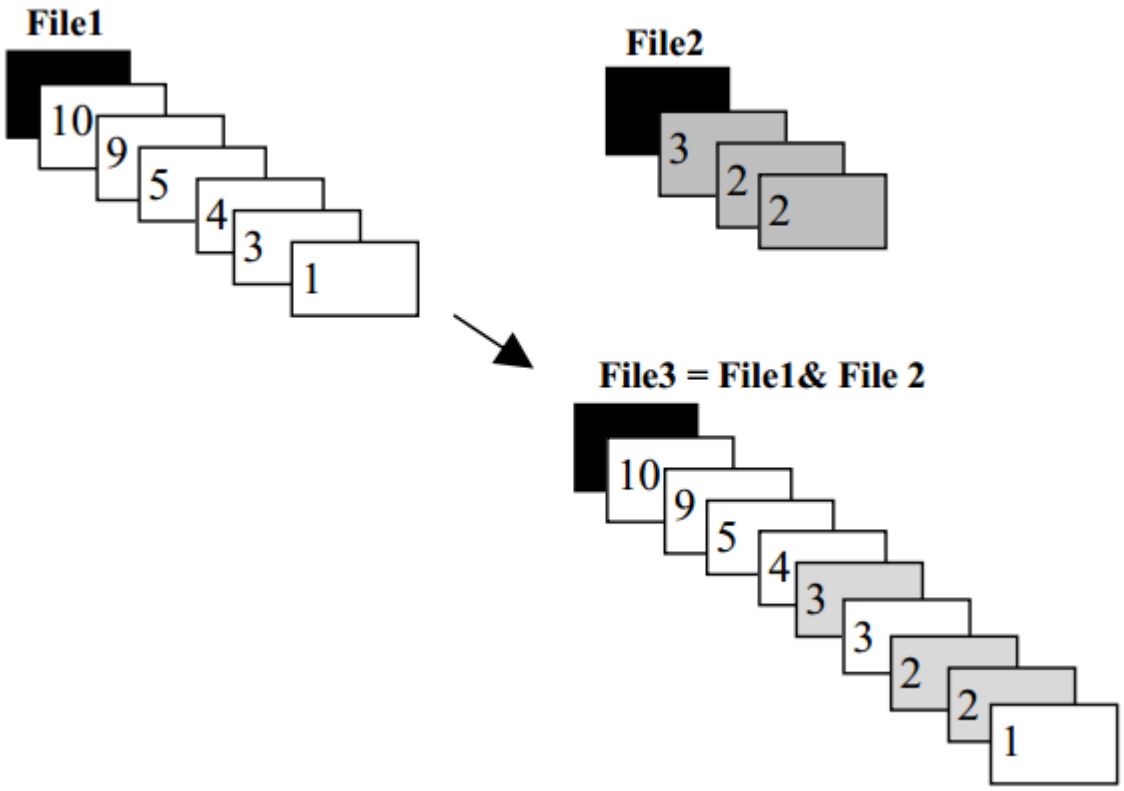
Algoritma Pemrosesan Dua Arsip: Merging



Algoritma Pemrosesan Dua Arsip: Merging

- Cara sebelumnya tak dapat dipakai jika kedua arsip sudah terurut, dan dikehendaki sebuah arsip hasil yang tetap terurut.
- Algoritma untuk penggabungan dua buah arsip terurut menjadi sebuah arsip yang terurut adalah:

Algoritma Pemrosesan Dua Arsip: Merging



Algoritma Merging 1 (Versi AND)

```
Program MERGING1
{ Input : dua  arsip sequential, terurut, sejenis}
{ Proses :Menggabung kedua arsip menjadi sebuah arsip yg terurut}
{        VERSI  AND  }
{ Output : Sequential file baru  yang terurut}
Kamus
   {  Keytype adalah suatu type dari kunci rekaman,
      Valtype adalah type dari harga rekaman}
   {  Keytype dan Valtype harus terdefinisi .
      Akhir arsip ditandai oleh mark dan Val}
   type rekaman : <  Key: keytype, {kunci }
                     Val:valtype   { harga lain yang direkam) >
   ArsipIn1 :  SEQFILE of {input, terurut menurut kunci }
      (*) RekIn1 : rekaman
      (1) <mark, Val>
   ArsipIn2 : SEQFILE of  {input, terurut menurut kunci }
      (*) RekIn2 : rekaman
      (1) <mark, Val>
   ArsipOut : SEQFILE of  {output, terurut menurut kunci }
      (*) RekOut : rekaman
      (1) <mark, Val>
Algoritma :
   OPEN(Arsip1, RekIn1)       {First_Elmt of Arsip1}
   OPEN(Arsip2, RekIn2)       {First_Elmt of Arsip2}
   REWRITE(ArsipOut)          {Menyiapkan arsip hasil: Arsip3}
   while  (RekIn1.Key ≠ mark)  and (RekIn2.Key≠mark) do
      depend on RekIn1.Key,RekIn2.Key:
         RekIn1.Key  ≠ RekIn2.Key : WRITE(ArsipOut, RekIn1)
                                    READ (ArsipIn1, RekIn1)
         RekIn1.Key > RekIn2.Key :  WRITE(ArsipOut, RekIn2)
                                    READ (ArsipIn2,RekIn2)
   { RekIn1.Key = mark or  RekIn2.Key = mark }
   while  (RekIn1.Key ≠ mark) do
      WRITE(ArsipOut, RekIn1)
      READ (ArsipIn1, RekIn1)
   { Akhir Arsip1, RekIn1.Key  =  mark }
   while  (RekIn2.Key ≠ mark) do
      WRITE(ArsipOut, RekIn2)
      READ (ArsipIn2,RekIn2)
   { Akhir Arsip2, RekIn2.Key  =  mark }
   WRITE(ArsipOut,<mark,Val>)
   CLOSE(ArsipIn1)
   CLOSE(ArsipIn2)
   CLOSE(ArsipOut)
```

Algoritma Merging 2 (Versi OR)

```
Program MERGING2
{ Input : dua  arsip sequential, terurut menaik menurut kunci,
sejenis, dan semua harga < Mark>}
{ Proses :Menggabung kedua arsip menjadi sebuah arsip yg terurut}
{        VERSI  OR }
{ Output : Sequential file baru  yang terurut}
```

```
Kamus
   {  Keytype adalah suatu type dari kunci rekaman,
      Valtype adalah type dari harga rekaman}
   {  Keytype dan Valtype harus terdefinisi .
      Akhir arsip ditandai oleh mark dan Val}
   type rekaman : <  Key: keytype, {kunci }
                     Val:valtype   { harga lain yang direkam) >
   ArsipIn1 : SEQFILE of {input, terurut menurut kunci }
      (*) RekIn1 : rekaman
      (1) <mark, Val>
   ArsipIn2 : SEQFILE of   {input, terurut menurut kunci }
      (*) RekIn2 : rekaman
      (1) <mark, Val>
   ArsipOut : SEQFILE of {output, terurut menurut kunci }
      (*) RekOut : rekaman
      (1) <mark, Val>
```

```
Algoritma :
   OPEN(ArsipIn1, RekIn1)    {First_Elmt of Arsip1}
   OPEN(ArsipIn2, RekIn2)    {First_Elmt of Arsip2}
   REWRITE (ArsipOut)  {Menyiapkan arsip hasil: Arsip3}
   while  (RekIn1.Key ≠ mark)  or (RekIn2.Key≠mark) do
      depend on RekIn1.Key,RekIn2.Key :
         RekIn1.Key  ≤ RekIn2.Key : WRITE(ArsipOut, RekIn1)
                                    READ (ArsipIn1, RekIn1)
         RekIn1.Key > RekIn2.Key :  WRITE(ArsipOut, RekIn2)
                                    READ (ArsipIn2,RekIn2)
   { RekIn1.Key = mark or  RekIn2.Key = mark }
   WRITE(ArsipOut, <mark,Val>)
   CLOSE(ArsipIn1)
   CLOSE(ArsipIn2)
   CLOSE(ArsipOut)
```

Algoritma Merging 2 (Versi OR): Catatan

- Versi OR ini teks algoritmanya lebih singkat daripada versi AND.
- Algoritma ini 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 nilai EOF yang ditentukan oleh sistem seperti pada kebanyakan sistem pengarsipan.

Algoritma Pemrosesan Dua Arsip: Updating dengan Transaction File

- Updating adalah mengubah harga rekaman yang ada pada sebuah master file dengan data dari transaction file.
- Berikut ini akan diberikan algoritma umum untuk meremajakan rekaman dari sebuah arsip sekuensial yang terurut dengan key unik (yang biasanya disebut dengan Master File).
- Peremajaan dilakukan terhadap rekaman yang ada, dilakukan berdasarkan arsip terurut lain (update file) dengan key yang tidak unik. Artinya satu rekaman pada Master File dapat mengalami 1 atau beberapa kali peremajaan.
- Hasil peremajaan dilakukan langsung terhadap Master File.
- Catatan:
  - Masalah harus dispesifikasikan secara cermat. Bagaimana jika ada peremajaan terhadap Master padahal key yang diremajakan tidak ada?
  - Contoh: peremajaan arsip saldo pada tabungan di bank, dengan perjanjian jumlah pada arsip update adalah negatif untuk pengambilan, positif untuk penabungan. Penabungan adalah dalam US$.

Algoritma Pemrosesan Dua Arsip: Updating dengan Transaction File

- Rekaman Master adalah <Key : integer, Saldo: integer>
- Rekaman Update adalah <Key : integer, Jumlah: integer>
- Arsip Master: <1,23> <3,34> <6,200> <16,10> <22,50> <30,0> <999,0>
- Arsip Update: <3,2> <3,4> <16,-10> <22,1> <25,50> <30,5> <999,0>
- Arsip Master Baru:<1,23> <3,40> <6,200> <16,0> <22,51> <30,5> <999,0>

Algoritma Updating dengan Transaction File

```
Program UPDATING
{ Input : dua  arsip sequential (MasterFile dan Transaction File),
terurut menaik menurut kunci yg sama}{ Proses :Meremajakan sebuah
field pada Master file berdasarkan Transaction File }
{ Output : Sequential file baru  yang terurut menaik menurut kunci}
{ Seq. processing terhadap Master file , Master File adalah
pengendali pd loop luar }
Kamus
{ Keytype: suatu type dari kunci rekaman,  Akhir arsip ditandai oleh mark}
   type rekMaster : < KeyM: integer, Saldo:integer>
   type rekTrans : < KeyT: integer, TransSaldo:integer>
    Master : SEQFILE of {input,  terurut menurut kunci }
      (*) RekM : rekMaster
      (1) <9999,0>
    Transaction : SEQFILE of {input, terurut menurut kunci }
      (*) RekT : rekTrans
      (1) <9999,0>
    NewMaster : SEQFILE of  { Output, terurut menurut kunci }
                          { Master file baru yg telah diremajakan }
      (*) rekNM : rekMaster
      (1) <9999,0>
   Newsaldo : integer
Algoritma :
      OPEN(Master, RekM)          {First_Elmt of Arsip Master}
      OPEN(Transaction, RekT){First_Elmt of Arsip Transaksi}
      REWRITE (NewMaster) {Menyiapkan arsip hasil  NewMaster}
      while  (RekM.Key ≠ 9999)   do
            while  (RekT.Key< RekM.Key) and  (RekT.Key ≠ 9999)  do
            {Error,  skip transaction file yg tidak ada pada master }
                  READ (Transaction, RekT)
      {RekT.Key ≥ RekM.Key  or  RekT.Key= 9999 }
      if (RekT.Key = RekM.Key) then     { updating }
         { Init Update }
         NewSaldo ← RekM.Saldo
         repeat
            NewSaldo   ← NewSaldo + RekT.TransSaldo
            READ (Transaction, RekT)
         until (RekT.Key  ≠ RekM.Key) or (RekT.Key =9999)
         { RekT.Key ≠RekM.Key or RekT.Key =9999 }
         { End_upDate }
         write(NewMaster, <RekM.Key,NewSaldo>)
      else{RekT.Key > RekM.Key,tidak ada update thd MasterFile,salin}
         write(NewMaster,<KeyM,Saldo>)
      READ (Master, RekM)   { Next Elmt of Master }
   WRITE(NewMaster, <9999,0>)
   CLOSE(Master);  CLOSE(Transaction)
   CLOSE(NewMaster)
```

Algoritma Pemrosesan Dua Arsip: Splitting

- Splitting adalah pemecahan sebuah arsip menjadi dua atau lebih arsip.
- Algoritmanya tergantung pada kriteria pemecahannya.
- Contoh:
  - Memisahkan sebuah arsip pegawai menjadi beberapa arsip sesuai dengan kode golongan
  - Memisahkan arsip data percobaan sesuai dengan kriteria data (misalnya yang layak dipakai dan yang harus dibuang)

Referensi

- Liem, Inggriani. 2007. Diktat Kuliah Dasar Pemrograman (Bagian Pemrograman Prosedural). Bandung: Institut Teknologi Bandung

Fakultas Informatika
School of Computing
Telkom University

THANK YOU