Array (Tabel)

Tim Pengajar KU1071

Sem. 1 2009-2010

Tujuan Perkuliahan

- Mahasiswa memahami makna dan kegunaan array (tabel)
- Mahasiswa dapat menggunakan notasi pendefinisian dan pengacuan array dengan benar hingga proses pencarian terhadap elemen array
- Mahasiswa dapat membuat program dengan menggunakan array

Array, Tabel Kontigu

- Type array adalah type yang mengacu kepada sebuah atau sekumpulan elemen melalui indeks
- Elemen dari array dapat diakses langsung jika dan hanya jika indeks terdefinisi
  - ditentukan harganya dan sesuai dengan domain yang didefinisikan untuk indeks tersebut
- Disebut juga sebagai **tabel, vektor** atau **larik**
- Merepresentasikan sekumpulan informasi yang bertype sama dan disimpan dengan urutan yang sesuai dengan definisi indeks **secara kontigu dalam memori** komputer
  - indeks harus suatu type yang mempunyai keterurutan (ada suksesor dan predesesor), misalnya type integer, karakter

Elemen tunggal vs Array

```
Program nilai_mata_kuliah
Kamus
   Mhs1,Mhs2,Mhs3,Mhs4,Mhs5: real
   rata2, nilaimax: real
Algoritma
   input(Mhs1,Mhs2,Mhs3,Mhs4,Mhs5)
   rata2←(Mhs1+Mhs2+Mhs3+Mhs4+Mhs5)/5
   nilaimax←max(max(max(max(Mhs1,Mhs2),Mhs3),Mhs4),Mhs5)
```

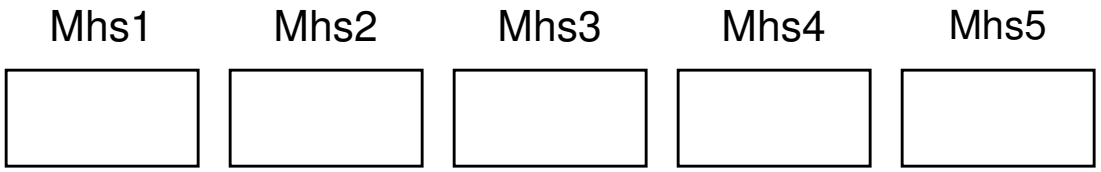
Bagaimana jika untuk 1000 Mhs?

Pembuatan 1 Mhs diwakili oleh 1 variabel TIDAK EFISIEN:

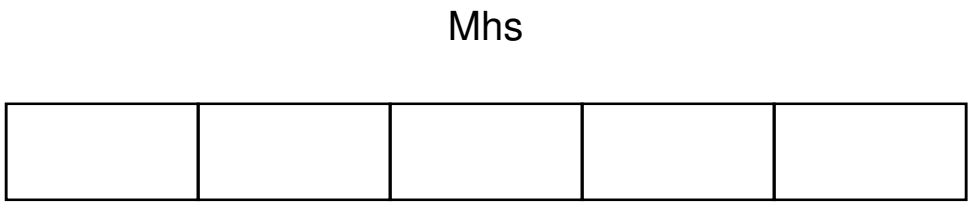
Mhs1, Mhs2, ..., Mhs1000

Solusi: penggunaan ARRAY

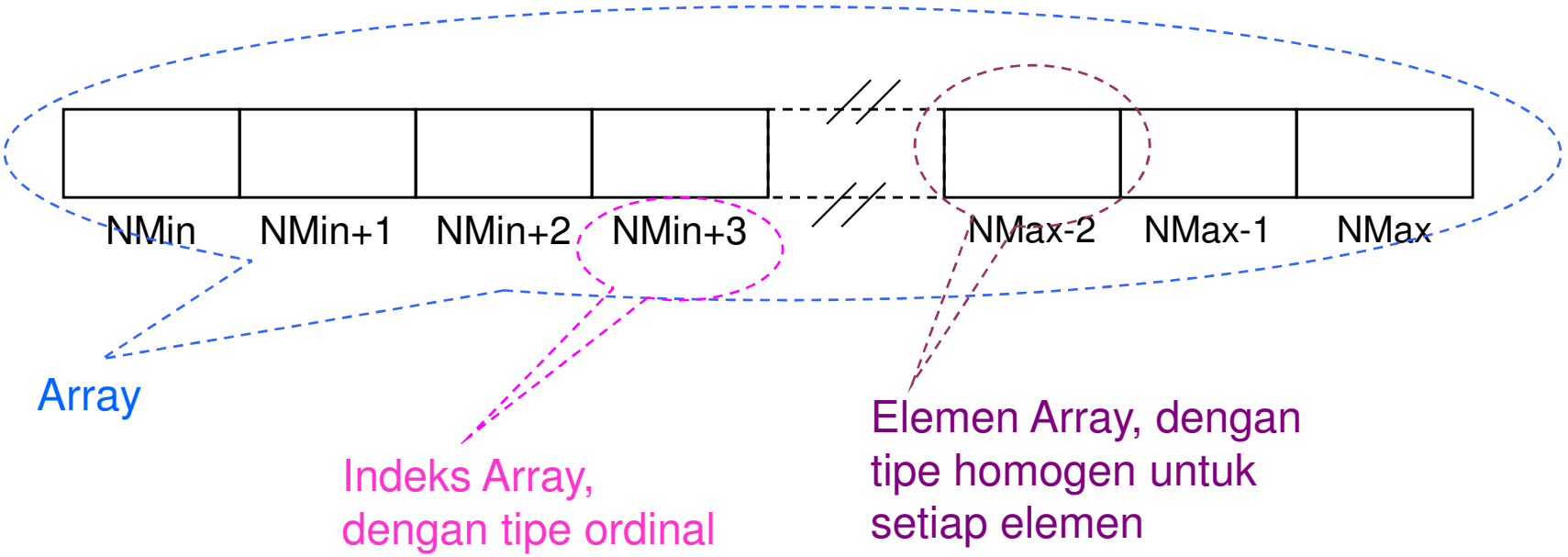
1 variabel menyimpan 1 nilai Mhs



1 variabel menyimpan 5 nilai Mhs
Type Array dengan 5 elemen



Array – contoh visualisasi



Contoh Penggunaan

```
KAMUS
   TabNamaHari : array [1..7] of string
   TabJumlahHari : array [1..12]  of integer
   type  Point : <x : integer, y : integer >
   type  Indeks : integer [1..10]
   TabTitikSurvey :  array [Indeks] of  Point
   TabFREK : array ['A'..'Z'] of integer
```

- **Domain :**
  - Domain array sesuai dengan pendefinisian indeks
  - Domain isi array sesuai dengan jenis array
- **Konstanta :**
  - Konstanta untuk seluruh array tidak terdefinisi
  - Konstanta hanya terdefinisi jika indeks dari array terdefinisi
- **Cara mengacu sebuah elemen:** melalui indeks

  $TabNamaHari_i$, jika i terdefinisi

  $TabNamaHari_7$

  $TabJumlahHari_3$

Contoh Pemakaian Array

Kasus-1: Nama Hari

Nama hari dalam minggu direpresentasi sebagai array. Tuliskan sebuah algoritma yang membaca hari ke berapa [1..7], kemudian menuliskan nama harinya.

```
Contoh :    Input : 1          Output "Senin"
            Input : 6          Output "Sabtu"
```

Program NamaHari

{ Mengisi TabelNamaHari yang akan memungkinkan untuk menuliskan nama hari : tabulasi eksplisit nama hari berdasarkan indeks HariKe... }

KAMUS

TabNamaHari : array [1..7] of string
procedure IsiTabHari { mengisi tabel nama hari }
HariKe : integer [1..7] {nomor dari hari}

ALGORITMA

IsiTabHari

{ Contoh pemanfaatan setelah Tabel TabNamaHari terdefinisi isinya }
Input (HariKe)
Output ($TabNamaHari_{HariKe}$)

Contoh Pemakaian Array
Kasus-1: Nama Hari (lanjutan)

procedure IsiTabHari
{ mengisi tabel nama hari }
{ I.S. : TabNamaHari tak terdefinisi }
{ F.S. : TabNamaHari siap dipakai, semua elemennya [1..7] sudah diisi }

Kamus Lokal :

ALGORITMA

$TabNamaHari_1 \leftarrow$ "Senin"
$TabNamaHari_2 \leftarrow$ "Selasa"
$TabNamaHari_3 \leftarrow$ "Rabu"
$TabNamaHari_4 \leftarrow$ "Kamis"
$TabNamaHari_5 \leftarrow$ "Jumat"
$TabNamaHari_6 \leftarrow$ "Sabtu"
$TabNamaHari_7 \leftarrow$ "Minggu"

Pemrosesan Sekuensial Pada Tabel

- Merupakan pemrosesan sekuensial tanpa mark
- Dimungkinkan adanya akses langsung jika indeks terdefinisi
  - First-Elmt adalah elemen tabel dengan indeks terkecil
  - Next-Elmt dicapai melalui suksesor indeks
- Model akses sekuensial tanpa mark
  - kondisi berhenti adalah jika indeks sudah mencapai harga indeks yang terbesar yang telah terdefinisi
- Tabel tidak mungkin “kosong”
  - jika kita mendefinisikan tabel, maka minimal mengandung sebuah elemen

Skema Pemrosesan Sekuensial

KAMUS UMUM PEMROSESAN TABEL

```
constant NMin: integer =1          {batas bawah}
constant NMax: integer =100        {batas atas}
type ElType : ...                  {suatu type terdefinisi, misalnya integer}
i : integer [NMin..NMax]
T : array [NMin..NMax] of ElType   {tabel dengan elemen bertype ElType}
procedure Inisialisasi {persiapan yang harus dilakukan sebelum pemrosesan}
procedure Proses (input  X : ElType)        {proses thd Current-Elmt tabel T}
procedure Terminasi     { "penutupan" setelah  pemrosesan selesai}
```

SKEMA PEMROSESAN TABEL T untuk indeks [NMin..NMax]

{Traversal Tabel T untuk Indeks bernilai NMin..NMax}

Skema :

```
Inisialisasi
i traversal [NMin..NMax]
                Proses(Ti)
Terminasi
```

Pemrosesan Sekuensial pada Tabel

Contoh 1: mengisi tabel, jumlah elemen diketahui

Program ISITABEL1

{ Traversal untuk mengisi, dengan membaca nilai setiap elemen tabel dari keyboard jika banyaknya elemen tabel yaitu N diketahui. Nilai yang dibaca akan disimpan di T_{NMin} s/d T_N. Nilai N harus dalam daerah nilai indeks yang valid }

KAMUS

```
constant NMin : integer  = 1 { NMin : batas bawah indeks}
constant NMax : integer = 100 { NMax : batas atas indeks}
i : integer [NMin..NMax]
T : array [NMin..NMax] of integer
N : integer
```

ALGORITMA

```
{Inisialisasi }
repeat
        input (N)
until (NMin ≤ N  ≤ NMax)
i traversal [NMin..N]
        input (Ti)
```

Catatan: N sebenarnya adalah indeks maksimum efektif. Tetapi, karena NMin=1, maka N juga berisi jumlah elemen tabel

Pemrosesan Sekuensial pada Tabel

Contoh 2: mengisi tabel, jumlah elemen tidak diketahui

Program ISITABEL2

{ Traversal untuk mengisi, dengan membaca nilai setiap elemen tabel dari keyboard yang diakhiri dengan 9999. Nilai yang dibaca akan disimpan di T_{NMin} s/d T_N, nilai N harus berada dalam daerah nilai indeks yang valid, atau 0 jika tabel kosong}

KAMUS

```
constant NMin : integer  = 1 { NMin : batas bawah indeks}
constant NMax : integer = 100 { NMax : batas atas indeks}
N : integer [0,NMin..NMax]        { indeks efektif tabel, 0 jika tabel kosong }
i : integer  [NMin..NMax+1]       { indeks untuk traversal tabel }
T : array [NMin..NMax] of integer
x : integer  { nilai yang dibaca  dan akan disimpan sebagai elemen tabel }
```

ALGORITMA

```
i ← NMin  {Inisialisasi }
input (x)  {First element }
while (x ≠9999) and (i<=NMax) do
        T_i ← x     { Proses }
        i ← i + 1
        input (x)  { Next element }
{x =9999 or i >NMax  }
if (i>NMax) then
   output ("Tabel sudah penuh ")
N ← i - 1
```

Jika pada saat input pertama kali sudah diisi "9999", maka N akan berisi NMin – 1. Karena NMin =1, proses ini aman (N diisi 0). Hati-hati jika NMin <> 1.

Table Lookup (*Searching*)

- Merupakan proses yang penting karena sering dilakukan terhadap sekumpulan data yang disimpan dalam tabel
- Ada beberapa variasi pencarian
  - Metoda mana yang dipakai menentukan kecepatan pencarian

```
KAMUS UMUM
   constant  NMax : integer  = 100
   type TabInt :  array  [1..NMax] of integer
{ jika diperlukan sebuah tabel, maka akan dibuat deklarasi sebagai berikut }
   T : TabInt              { tabel integer }
   N : integer             {indeks efektif, 1 ≤ N ≤ NMax}
```

Pencarian Berturutan (*Sequential Search*)

- Diketahui sebuah tabel berisi integer T[1..N], yang telah diisi
- Tuliskanlah algoritma yang menerima masukan sebuah X bernilai integer dan mencari apakah harga X ada dalam T secara sekuensial (berturutan)
  - Menghasilkan harga indeks IX dimana X diketemukan pertama kalinya, IX diberi harga 0 jika pencarian tidak ketemu
  - Pencarian segera dihentikan begitu harga pertama diketemukan

Contoh-1: N = 8, T berisi : { 1, 3, 5, -8, 12, 90, 3, 5 }, X = 5

- Pemeriksaan dilakukan terhadap {1,3,5}
- Output : IX = 3

Contoh-2: N = 4, T berisi : { 11, 3, 5, 8 }, X = 100

- Pemeriksaan dilakukan terhadap {11,3,5,8}
- Output : IX = 0

Algoritma Pencarian *Sequential*

Skema Pencarian Tanpa Boolean

procedure SEQSearchX1 (input T : TabInt, input N : integer,
output IX : integer, input X : integer)
{Mencari harga X dalam Tabel T [1..N] secara sekuensial mulai dari T_1. Hasilnya adalah indeks IX dimana T_{ix} = X (i terkecil), IX = 0 jika tidak ketemu}

Kamus Lokal :
i : integer [1..NMax] {indeks untuk pencarian }

ALGORITMA
i ← 1
while (i < N) and ($T_i \neq X$) do
i ← i + 1
{ i = N or T_i = X}
if (T_i = X) then
IX ← i
else { $T_i \neq X$ }
IX ← 0

Algoritma Pencarian *Sequential*

Skema Pencarian Dengan Boolean

```
procedure SEQSearchX2 (input T : TabInt, input N : integer,
                       input X : integer, output IX : integer, output Found: boolean)
{Mencari harga X dalam Tabel T [1..N] secara sekuensial mulai dari T1. Hasilnya
adalah indeks IX dimana Tix = X (i terkecil), IX = 0 jika tidak ketemu, dan sebuah
boolean Found (true jika ketemu) }
Kamus Lokal :
        i : integer  [1..N+1]          {indeks untuk pencarian }
ALGORITMA
Found ← false                  { awal pencarian, belum ketemu }
i ← 1
while  (i ≤ N) and (not  Found) do
        if (Ti = X) then
                Found ← true
        else
                i ← i + 1
{ i > N or Found }
if (Found) then
        IX ← i
else
        IX ← 0
```

Algoritma Pencarian *Sequential*

Skema Pencarian Dengan Boolean (salah)

procedure SEQSearchX3 (input T : TabInt, input N : integer,
input X : integer, output IX : integer, output Found: boolean)
{Mencari harga X dalam Tabel T [1..N] secara sekuensial mulai dari T_1. Hasilnya adalah indeks IX dimana T_{ix} = X (i terkecil), IX = 0 jika tidak ketemu, dan sebuah boolean Found (true jika ketemu) }

Kamus Lokal :
i : integer [1..N+1] {indeks untuk pencarian }

ALGORITMA
i ← 1
while (i ≤ N) and **(T_i ≠ X)** do
i ← i + 1
{ **i = N+1** or T_i = X }
if (i ≤ N) then
Found ← true; IX ← i
else { i > N }
Found ← false; IX ← 0

Dimana letak kesalahan algoritma ini?

Algoritma Pencarian *Sequential*
Pada Tabel Terurut

- Diketahui sebuah tabel bilangan integer T[1..N], dengan isi yang terurut membesar (untuk setiap i $\in$ [1..N-1], $T_i \leq T_{i+1}$
- Tuliskanlah algoritma, yang jika diberikan sebuah X bernilai integer akan mencari apakah harga X ada dalam T secara sekuensial mulai dari elemen pertama
  - Prosedur akan menghasilkan harga indeks IX dimana X diketemukan pertama kalinya, IX diberi harga 0 jika pencarian tidak ketemu
  - Pencarian segera dihentikan begitu harga pertama diketemukan
- Dengan memanfaatkan keterurutan, kondisi berhenti bisa lebih efisien

Contoh 1: N = 8, T berisi : { 1, 3, 5, 8, 12, 90, 311, 500}, X = 5
- Pemeriksaan dilakukan terhadap {1,3,5}
- Output : IX = 3

Contoh 2: N = 7, T berisi : { 11, 30, 50, 83,99,123,456}, X = 100
- Pemeriksaan dilakukan terhadap {11,30,50,83,99,123}
- Output : IX = 0

Algoritma Pencarian *Sequential*

Pada Tabel Terurut - Algoritma

procedure SEQSearchSorted (input T : TabInt, input N : integer,
input X : integer, output IX : integer)
{Mencari harga X dalam Tabel T [1..N] secara sekuensial mulai dr T_1}
{Hasilnya adalah indeks IX di mana T_{IX} =X; IX = 0 jika tidak ketemu }

Kamus Lokal :
i : integer [1..NMax] {indeks untuk pencarian }

ALGORITMA

```
i ← 1
while (i < N) and (Ti < X)
        i ← i + 1
{ i = N or Ti ≥ X }
if (Ti = X ) then
        IX ← i
else    { Ti ≠ X ➔ Ti > X }
        IX ← 0
```

Algoritma Pencarian *Sequential*
Dengan Sentinel

- Dengan teknik sentinel, sengaja dipasang suatu elemen fiktif setelah elemen terakhir tabel, yang disebut SENTINEL.
  - Elemen fiktif ini harganya sama dengan elemen yang dicari
  - Pencarian akan selalu ketemu; harus diperiksa lagi apakah posisi ketemu :
    - di antara elemen tabel yang sebenarnya, atau
    - sesudah elemen terakhir (berarti tidak ketemu, karena elemen fiktif)
  - Penempatan sentinel disesuaikan dengan arah pencarian
- Teknik sentinel sangat efisien, terutama jika pencarian dilakukan sebelum penyisipan sebuah elemen yang belum terdapat di dalam tabel

Contoh 1: N = 8, T berisi : { 1, 3, 5, 8, -12, 90, 3, 5}, X = 5
- T dijadikan { 1, 3, 5, 8, 12, 90, 3, 5, **5**}
- Pemeriksaan dilakukan terhadap {1,3,5}
- Output : IX = 3

Contoh 2: N = 4, T berisi : { 11, 3, 5, 8}, X = 100
- Akibatnya minimal ukuran array harus 5, berarti N dijadikan 5
- T dijadikan { 11, 3, 5, 8, **100**}
- Pemeriksaan dilakukan terhadap {11,3,5,8,100}
- Output : IX = 0

Algoritma Pencarian *Sequential*

Dengan Sentinel (lanjutan)

- Kamus umum untuk tabel dengan sentinel harus mengandung sebuah elemen tambahan

```
KAMUS UMUM
    constant  NMax : integer  = 100
    type TabInt :  array  [1..NMax+1] of integer
{ jika diperlukan sebuah tabel, maka akan dibuat deklarasi sebagai berikut }
    T : TabInt     {tabel integer}
    N : integer    {indeks efektif, maksimum tabel yang terdefinisi, 1≤N≤NMax}
```

Algoritma Pencarian *Sequential*

Dengan Sentinel - Algoritma

```
procedure SEQSearchWithSentinel (input T : TabInt, input N : integer,
                                 input X : integer, output IX : integer)
{ Mencari harga X dalam Tabel T[1..N] secara sekuensial mulai dari T1}
{ Hasilnya adalah indeks IX dimana TIX =X (IX terkecil) }
{ IX = 0 jika tidak ketemu. Sentinel diletakkan di TN+1        }
```

Kamus Lokal :

```
        i : integer [1..N+1] {indeks untuk pencarian }
```

ALGORITMA

```
TN+1 ← X            {pasang sentinel }
i ← 1
while  (Ti ≠ X) do   {tidak perlu test terhadap batas i, karena pasti berhenti}
        i ← i + 1
{ Ti = X; harus diperiksa apakah ketemunya di sentinel }

if  (i < N+1) then
        IX ← i {ketemu pada elemen tabel }
else    { i = N+1 }
        IX ← 0  {sentinel, berarti tidak ketemu }
```

Translasi ke Pascal

- Pendefinisian Array:

```
<nm_array> : array[<nmin>..<nmax>]
                of <tipe>;
```

Contoh:

```
TInt : array[1..100] of integer;
TPoint : array[1..MaxPoint] of Point;
```

- Pengacuan elemen Array:

```
<nm_array>[<index>]
```

Contoh:

```
TInt[5] := 10;
Hasil := Hasil + TInt[10];
Readln (TInt[i]);
```

Latihan

- Perhatikan kembali program ISITABEL1 (halaman 123). Buatlah program ISITABEL1a (yang merupakan revisi dari program ISITABEL1) yang menggunakan array dengan indeks minimum (NMin) 101 dan indeks maksimum (NMax) 200.
- Buatlah sebuah fungsi HitungRataTabel yang menghitung nilai rata-rata dari seluruh elemen array TabInt (array dengan elemen bertipe integer, merupakan parameter fungsi) yang elemennya berjumlah N (merupakan parameter fungsi).
- Buatlah sebuah prosedur KemunculanTerakhir yang mencari indeks array terakhir yang berisi suatu nilai. Parameter prosedur adalah TabInt (array dengan elemen integer), N (indeks efektif), X (nilai yang akan dicari), dan IX (indeks terakhir pada array yang bernilai X).

  Contoh: N=9, TabInt={4,3,7,6,3,8,4,3,6}, X=3, maka IX=8

Latihan di Rumah

- Semua algoritma yang dipelajari pada perkuliahan hari ini menggunakan tipe data integer. Cobalah untuk mengembangkan algoritma-algoritma yang ada agar dapat menangani array dengan elemen bertipe bentukan.