



# Skema Standar (Bag. 2)

Tim Pengajar

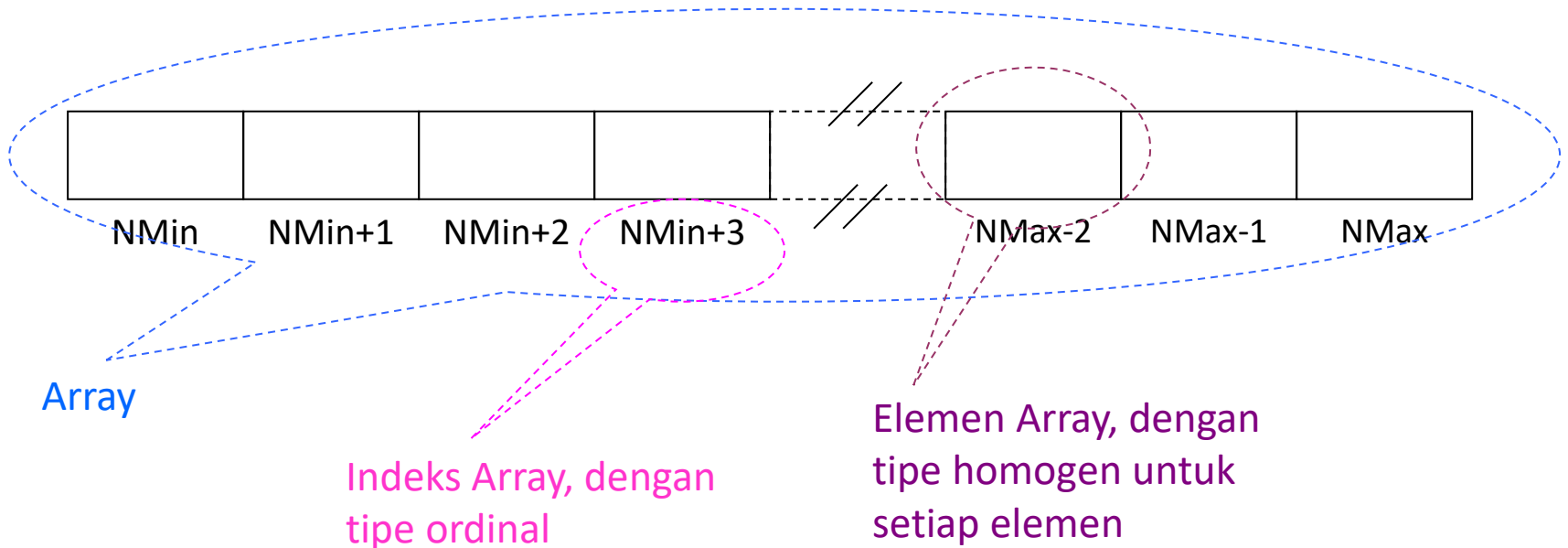
IF1210 Dasar Pemrograman

Sem. 2 2019/2020



# **SKEMA PEMROSESAN ARRAY/ TABEL**

# Array



- **Array** mendefinisikan **sekumpulan** (satu atau lebih) elemen **bertipe sama**
- Setiap elemen tersusun secara teratur (kontigu) dan dapat diakses dengan menggunakan **indeks**

# Deklarasi Array sbg. **Variabel** dalam Notasi Algoritmik (1)



- Deklarasi array sebagai variabel di **KAMUS**  
*nama-var* : array [*idmin*..*idmax*] of *type-elmt*
  - Deklarasi variabel array dengan nama *nama-var* dengan indeks elemen terkecil *idmin* dan indeks terbesar *idmax*
    - Indeks bertipe ordinal, misalnya integer. Indeks bisa dimulai dari nilai berapa pun (ini berbeda dengan beberapa bahasa pemrograman, misalnya Python)
  - Type elemen array ditentukan oleh *type-elmt*



# Deklarasi Array sbg. **Variabel** dalam Notasi Algoritmik (2)

- Cara akses sebuah elemen:

**nama-var**<sub>indeks</sub>

- Contoh deklarasi array

**Tab : array[1..100] of integer**

- Contoh akses elemen:

**output(Tab<sub>5</sub>)**

**x ← Tab<sub>1</sub> + Tab<sub>6</sub>**

**Tab<sub>9</sub> ← 9**

# Deklarasi Array sbg. **Type** dalam Notasi Algoritmik

- Array dapat menjadi salah satu komponen dalam type bentukan.
- **Contoh:** deklarasi type TabInt:

## **KAMUS**

constant NMax : integer = 100

**type** TabInt : **array** [1..NMax] **of** **integer**  
{ Variabel }

T : TabInt

## **ALGORITMA**

$T_1 \leftarrow 0$  { contoh cara akses }

- TabInt adalah type dengan komponen tunggal, yaitu sebuah array of integer dengan indeks dari 1 s.d. Nmax (konstanta)
- T adalah variabel bertipe TabInt

# Contoh Lain

## KAMUS

```
TabKata : array [1..100] of string
TabJumlahHari : array [1..12] of integer
type Point : < x : integer,
                y : integer>
TabTitikSurvey : array [1..10] of Point
```

- **Domain:**
  - Domain array sesuai dengan pendefinisian indeks
  - Domain isi array sesuai dengan jenis array
- Cara mengacu sebuah elemen: melalui indeks
  - TabKata<sub>i</sub>     {jika i terdefinisi}
  - TabKata<sub>7</sub>
  - TabJumlahHari<sub>3</sub>
  - TabTitikSurvey<sub>4</sub>.x {akses komponen x dari Point pada array elemen ke-4}



# Pemrosesan Sekuensial pada Array

- 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 ARRAY

**constant** NMin : integer = 1 { batas bawah }

**constant** NMax : integer = 100 { batas atas }

### type

ElType : ... { suatu type terdefinisi, misalnya integer }

{ Variabel }

i : integer[NMin..NMax]

T : array [NMin..NMax] of ElType { array berelemen ElType }

{ Deklarasi Prosedur }

procedure *Inisialisasi* { persiapan sebelum pemrosesan }

procedure *Proses* (input X : ElType) {proses current-elmt array T}

procedure *Terminasi* { penutupan setelah pemrosesan selesai }

{ **SKEMA** PEMROSESAN ARRAY T untuk indeks [NMin..NMax] }

{ Traversal Array T untuk indeks bernilai NMin..NMax }

### { **Skema** }

*Inisialisasi*

i traversal[Nmin..Nmax]

*Proses*( $T_i$ )

*Terminasi*

# Skema Pemrosesan Sekuensial



## KAMUS UMUM PEMROSESAN ARRAY

**constant** NMin : integer = 1 { batas bawah }

**constant** NMax : integer = 100 { batas atas }

### type

ElType : ... { suatu type terdefinisi, misalnya integer }  
{ Variabel }

i : integer[NMin..NMax]

T : array [NMin..NMax] of ElType { array berelemen ElType }

{ Deklarasi Prosedur }

procedure *Inisialisasi* { persiapan sebelum pemrosesan }

procedure *Proses* (input X : ElType) {proses current-elmt array T}

procedure *Terminasi* { penutupan setelah pemrosesan selesai }

{ **SKEMA** PEMROSESAN ARRAY T untuk indeks [NMin..NMax] }

{ Traversal Array T untuk indeks bernilai NMin..NMax }

### { Skema }

*Inisialisasi*

i traversal[Nmin..Nmax]

*Proses* ( $T_i$ )

*Terminasi*

Diasumsikan semua elemen array  
T sudah terisi (terdefinisi)



# Skema Pengisian dan Penulisan Isi Array (Skema Traversal terhadap Array)

# 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  $T_{N_{Min}}$  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 (N >= NMin) and (N <= NMax);  
{ Pengisian array dari pembacaan dari keyboard }  
i traversal [Nmin..N]  
    input ( $T_i$ )
```

# 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  $T_{N_{Min}}$  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 (N >= NMin) and (N <= NMax);  
{ Pengisian array dari pembacaan dari keyboard }  
i traversal [Nmin..N]  
    input ( $T_i$ )
```

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

# Mengisi Tabel, Jumlah Elemen Tidak Diketahui (1)



## **Program** ISITABEL2

{ Traversal untuk mengisi, dengan membaca nilai setiap elemen tabel dari keyboard yang diakhiri dengan 9999. Nilai yang dibaca akan disimpan di  $T_{N_{Min}}$  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 }

i : integer[NMin..NMax]

T : array [NMin..NMax] of integer

N : integer { indeks efektif tabel, 0 jika tabel kosong }

x : integer { nilai yg dibaca & akan disimpan sbg elemen tabel }

## **ALGORITMA**

... { next slide }

# Mengisi Tabel, Jumlah Elemen Tidak Diketahui (2)



## Algoritma

```
i ← NMin           { Inisialisasi }
i ← x         { Proses }
    i ← i + 1
    output ("Tabel sudah penuh")
N ← i - 1
```

# Mengisi Tabel, Jumlah Elemen Tidak Diketahui (2)



## Algoritma

```
i ← NMin           { Inisialisasi }  
input (x)          { First element }  
  
while (x ≠ 9999) and (i ≤ NMax) do  
    Ti ← 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 read 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.



# Menuliskan Isi Tabel Secara Mundur



**Program** TULISTABELMundur

{Menuliskan isi tabel dari indeks terbesar ke indeks terkecil}

## KAMUS

**constant** NMin : integer = 1

**constant** NMax : integer = 100

TabelInt : array [NMin..NMax] of integer

i : integer[NMin..NMax]

N : integer { ukuran efektif tabel yang terisi 1..N }

## ALGORITMA

{ Di titik ini: TabelInt[NMin..NMax] sudah diisi,  
algoritma berikut hanya menuliskan mundur }

i traversal [NMax..NMin]

output ( $T_i$ )



# Skema Pencarian Nilai Ekstrim dalam Array (Nilai Maksimum/Minimum)

# Pencarian Nilai Ekstrim

- Kamus umum yang digunakan:

## 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,  $1 \leq N \leq Nmax$  }

- Pada algoritma berikut diasumsikan array tidak kosong
  - Nilai ekstrim pada tabel kosong tidak terdefinisi

# Pencarian Nilai Maksimum (1)

## Versi mengembalikan NILAI maksimum



```
procedure MAX1 (input T : TabInt, input N : integer,  
                 output MAX : integer)
```

```
{ Pencarian harga maksimum:
```

```
  I.S. Tabel T tidak kosong, karena jika kosong maka maks tidak  
      terdefinisi,  $N > 0$ 
```

```
  F.S. Menghasilkan harga Maksimum MAX dari tabel  $T_{1..N}$  secara  
      sekuensial mulai dari indeks  $1..N$  }
```

### Kamus Lokal

```
  i : integer { indeks untuk pencarian }
```

### Algoritma

```
MAX  $\leftarrow$   $T_1$  { inisialisasi,  $T_1$  diasumsikan adl. nilai maks }  
i  $\leftarrow$  2 { perbandingan nilai maks dimulai dari elemen ke-2 }  
while (i  $\leq$  N) do  
    if (MAX  $<$   $T_i$ ) then  
        MAX  $\leftarrow$   $T_i$   
    i  $\leftarrow$  i + 1  
{ i > N : semua elemen sudah selesai diperiksa }
```

# Pencarian Nilai Maksimum (1)

## Versi mengembalikan NILAI maksimum



```
procedure MAX1 (input T : TabInt, input N : integer,  
                 output MAX : integer)
```

```
{ Pencarian harga maksimum:
```

```
  I.S. Tabel T tidak kosong, karena jika kosong maka maks tidak  
      terdefinisi,  $N > 0$ 
```

```
  F.S. Menghasilkan harga Maksimum MAX dari tabel  $T_{1..N}$  secara  
      sekuensial mulai dari indeks  $1..N$  }
```

### Kamus Lokal

```
  i : integer { indeks untuk pencarian }
```

Nilai yang dihasilkan adalah nilai maksimum, indeks tempat nilai maksimum tidak diketahui

### Algoritma

```
MAX  $\leftarrow$   $T_1$  { inisialisasi,  $T_1$  diasumsikan adl. nilai maks }  
i  $\leftarrow$  2 { perbandingan nilai maks dimulai dari elemen ke-2 }  
while (i  $\leq$  N) do  
  if (MAX  $<$   $T_i$ ) then  
    MAX  $\leftarrow$   $T_i$   
  i  $\leftarrow$  i + 1  
{ i > N : semua elemen sudah selesai diperiksa }
```

Elemen pertama tabel diproses secara khusus

# Pencarian Nilai Maksimum (2)

## Versi mengembalikan INDEKS maksimum

```
procedure MAX2 (input T : TabInt, input N : integer,  
                 output IMax : integer)  
{ Pencarian indeks dengan harga Maksimum  
  I.S. Tabel tidak kosong, karena jika kosong maka maks tidak  
    terdefinisi,  $N > 0$   
  F.S. Menghasilkan indeks IMax terkecil, dengan harga  
     $T_{IMax}$  dalam Tabel  $T_{1..N}$  adalah maksimum }
```

### Kamus Lokal

i : integer

### Algoritma

```
IMax  $\leftarrow$  1  
i  $\leftarrow$  2  
while (i  $\leq$  N) do  
    if ( $T_{IMax} < T_i$ ) then  
        IMax  $\leftarrow$  i  
        i  $\leftarrow$  i + 1  
{ i > N : semua elemen sudah selesai diperiksa }
```

# Pencarian Nilai Maksimum (2)

## Versi mengembalikan INDEKS maksimum

```
procedure MAX2 (input T : TabInt, input N : integer,  
                 output IMax : integer)  
{ Pencarian indeks dengan harga Maksimum  
  I.S. Tabel tidak kosong, karena jika kosong maka maks tidak  
    terdefinisi,  $N > 0$   
  F.S. Menghasilkan indeks IMax terkecil, dengan harga  
     $T_{IMax}$  dalam Tabel  $T_{1..N}$  adalah maksimum }
```

### Kamus Lokal

i : integer

Tidak menghasilkan nilai maksimum  
melainkan indeks dimana nilai maksimum  
berada

### Algoritma

IMax  $\leftarrow$  1

i  $\leftarrow$  2

```
while (i <= N) do  
  if ( $T_{IMax} < T_i$ ) then  
    IMax  $\leftarrow$  i  
  i  $\leftarrow$  i + 1
```

```
{ i > N : semua elemen sudah selesai diperiksa }
```

Elemen pertama tabel diproses secara khusus

# Pencarian Nilai Maksimum (3)

## Versi maksimum dari bil. positif (v.1)



```
procedure MAXPOS (input T : TabInt, input N : integer,  
                   output Max : integer)  
{ Pencarian harga Maksimum di antara bilangan positif  
  I.S. Tabel mungkin kosong: N=0, semua elemen tabel T bernilai  
    positif  
  F.S. Max berisi nilai elemen tabel yang maksimum. Jika kosong,  
    Max = -9999  
  Menghasilkan harga Maksimum (MAX) Tabel  $T_{1..N}$  secara  
    sekuensial mulai dari  $T_1$  }
```

### Kamus Lokal

i : integer

### Algoritma

```
Max  $\leftarrow$  -9999 { inisialisasi dgn nilai yg pasti digantikan!  
                  misal nilai minimum representasi integer }  
i traversal [1..N]  
  if (Max <  $T_i$ ) then  
    Max  $\leftarrow$   $T_i$   
{ i = ??, semua elemen sudah selesai diperiksa }
```



# Pencarian Nilai Maksimum (3)

## Versi maksimum dari bil. positif (v.1)



```
procedure MAXPOS (input T : TabInt, input N : integer,  
                  output Max : integer)  
{ Pencarian harga Maksimum di antara bilangan positif  
  I.S. Tabel mungkin kosong: N=0, semua elemen tabel T bernilai  
    positif  
  F.S. Max berisi nilai elemen tabel yang maksimum. Jika kosong,  
    Max = -9999  
  Menghasilkan harga Maksimum (MAX) Tabel  $T_{1..N}$  secara  
    sekuensial mulai dari  $T_1$  }
```

### Kamus Lokal

i : integer

### Algoritma

```
Max ← -9999 { inisialisasi dgn nilai yg pasti digantikan!  
              misal nilai minimum representasi integer }
```

```
i traversal [1..N]  
  if (Max <  $T_i$ ) then  
    Max ←  $T_i$ 
```

```
{ i = ??, semua elemen sudah selesai diperiksa }
```

Semua elemen tabel diperiksa dengan cara yang sama. Oleh sebab itu, nilai MAX harus diinisialisasi dengan nilai yang sudah pasti akan digantikan oleh nilai yang ada di dalam tabel

Pengulangan ini tidak aman untuk seluruh kasus. Carilah letak permasalahannya.

# Pencarian Nilai Maksimum (4)

## Versi maksimum dari bil. positif (v.2)



```
procedure MAXPOS (input T : TabInt, input N : integer,  
                  output Max : integer)  
{ Pencarian harga Maksimum di antara bilangan positif  
  I.S. Tabel mungkin kosong: N=0, semua elemen tabel T bernilai  
    positif  
  F.S. Max berisi nilai elemen tabel yang maksimum. Jika kosong,  
    Max = -9999  
  Menghasilkan harga Maksimum (MAX) Tabel  $T_{1..N}$  secara  
    sekuensial mulai dari  $T_1$  }
```

### Kamus Lokal

i : integer

### Algoritma

```
Max  $\leftarrow$  -9999 { inisialisasi dgn nilai yg pasti digantikan!  
                  misal nilai minimum representasi integer }  
i  $\leftarrow$  1  
while (i <= N) do  
    if (Max <  $T_i$ ) then  
        Max  $\leftarrow$   $T_i$   
        i  $\leftarrow$  i + 1;  
{ i = N+1, semua elemen sudah selesai diperiksa }
```



# **SKEMA PENCARIAN PADA ARRAY/ TABEL**

# *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}
```

# Algoritma Pencarian *Sequential*

## Skema Pencarian Tanpa Boolean



```
procedure SEQSearchX1 (input T : TabInt, input N : integer,  
                        input X : integer, output IX : integer)  
{ Mencari harga X dalam Tabel T [1..N] secara sekuensial mulai  
  dari  $T_1$ . Hasilnya adalah indeks IX di mana  $T_{IX} = X$  (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 Tanpa Boolean



```
procedure SEQSearchX1 (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 di mana TIX = 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<sub>i</sub> ≠ X) do

    i ← i + 1

{ i = N or T<sub>i</sub> = X }

if (T<sub>i</sub> = X) then

    IX ← i

else { T<sub>i</sub> ≠ X }

    IX ← 0

Harus dipastikan bahwa T tidak kosong, sehingga tidak ada pemeriksaan T<sub>i</sub> yang tidak terdefinisi

# 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  $T_1$ . Hasilnya adalah indeks IX di mana  $T[IX] = X$  (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

```
Found  $\leftarrow$  false { awal pencarian, belum ketemu }
```

```
i  $\leftarrow$  1
```

```
while (i <= N) and (not(Found)) do
```

```
    if ( $T_i = X$ ) then
```

```
        Found  $\leftarrow$  true
```

```
    else
```

```
        i  $\leftarrow$  i + 1
```

```
{ i > N or Found }
```

```
if (Found) then
```

```
    IX  $\leftarrow$  i
```

```
else
```

```
    IX  $\leftarrow$  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  $T_1$ . Hasilnya adalah indeks IX di mana  $T[IX] = X$  (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

Found  $\leftarrow$  false { awal pencarian, belum ketemu }

i  $\leftarrow$  1

while (i <= N) and (not(Found)) do

if ( $T_i = X$ ) then  
        Found  $\leftarrow$  true

else

        i  $\leftarrow$  i + 1

{ i > N or Found }

if (Found) then

        IX  $\leftarrow$  i

else

        IX  $\leftarrow$  0

T boleh kosong karena tidak ada pemeriksaan nilai  $T[i]$  pada kondisi pengulangan



# Algoritma Pencarian Sequential Pada Tabel Terurut (1)



- Diketahui sebuah tabel bilangan integer  $T_{1..N}$ , dengan isi yang terurut membesar  
 $\forall i \in [1..N-1], T_i \leq T_{i+1}$
- 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  $IX$  di mana  $X$  diketemukan pertama kalinya,  $IX$  diberi harga 0 jika pencarian tidak ketemu
  - Pencarian segera dihentikan begitu harga pertama diketemukan

# Algoritma Pencarian Sequential Pada Tabel Terurut (2)



- 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



```
procedure SEQSearchSorted (input T : TabInt, input N : integer,  
                           input X : integer,  
                           output IX : integer)  
{ Mencari harga X dalam Tabel T1..N secara sekuensial mulai  
  dari T1. Hasilnya adalah indeks IX di mana TIX = X, IX = 0  
  jika tidak ketemu. }
```

## Kamus Lokal

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

## ALGORITMA

```
i ← 1  
while (i < N) and (Ti < X) do  
    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 (1)



- 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

# Algoritma Pencarian Sequential Dengan Sentinel (2)



- 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 (3)



- 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 di mana TIX = X  
  (i 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
```

```
  { T[i] = X, diperiksa apakah ketemunya di sentinel }
```

```
  if (i < N+1) then
```

```
    IX ← i { ketemu pada elemen tabel }
```

```
  else { i = N+1, ditemukan di sentinel, berarti tak ketemu }
```

```
    IX ← 0
```



# Catatan Implementasi di Python

- Di Python ada berbagai jenis struktur data yang mirip dengan struktur array
- Untuk perkuliahan IF1210 akan digunakan *collection type* **list** untuk merepresentasikan array
  - List merupakan salah satu struktur data dasar pada Python 3
  - Ada beberapa jenis struktur data array lain yang memerlukan library khusus; dipersilakan untuk mempelajari mandiri jika tertarik



# Catatan Implementasi di Python

## Array sbg. Variabel



- Contoh notasi algoritmik:

### **KAMUS**

```
constant NMax : integer = 100  
T : array [0..NMax-1] of integer
```

### **ALGORITMA**

```
T0 ← 0 { contoh cara akses elemen array ke-0 }
```

- Translasi ke Python:

### **# KAMUS**

```
NMax = 100 # constant NMax : int  
T = [0 for i in range(NMax)] # deklarasi array
```

### **# ALGORITMA**

```
T[0] = 0 # contoh cara akses elemen array ke-0
```

# Catatan Implementasi di Python

## Array sbg. Type



- Contoh notasi algoritmik:

### KAMUS

constant NMax : integer = 100

type TabInt : array [0..NMax-1] of integer

{ Variabel }

T : TabInt

### ALGORITMA

$T_0 \leftarrow 0$  { contoh cara akses elemen array ke-0 }

- Translasi ke Python:
  - Tidak ada translasi type TabInt di Python
  - Untuk itu, T dapat dideklarasikan sbg variabel biasa spt pada slide sebelumnya

# Rangkuman

- Apa yang sudah dipelajari hari ini?
  - Skema standar: skema pemrosesan array
    - pengisian dan penulisan isi array (skema traversal terhadap Array)
    - pencarian nilai ekstrim
    - pencarian (searching/look-up): searching tanpa boolean, dengan boolean, menggunakan sentinel, searching pada tabel terurut



# **SELAMAT BELAJAR**