PENCARIAN BERUNTUN
(SEQUENTIAL SEARCHING)

a. Introduction
b. Tanpa Boolean
c. Dengan Boolean
d. Penggunaan dalam Fungsi

INTRODUCTION

- Merupakan algoritma pencarian yang paling sederhana.
- Proses → Membandingkan setiap elemen larik satu per satu secara beruntun, mulai dari elemen pertama, sampai elemen yang dicari ditemukan, atau seluruh elemen sudah diperiksa.

INTODUCTION [→]

13	16	14	21	76	21
1	2	3	4	5	6

- Misalkan nilai yang dicari adalah: X = 21
  Maka, elemen yang diperiksa: 13,16,14,21 (ditemukan!)
  Indeks larik yang dikembalikan: IDX = 4
- Misalkan nilai yang dicari adalah: X = 13
  Maka, elemen yang diperiksa: 13 (ditemukan!)
  Indeks larik yang dikembalikan: IDX = 1
- Misalkan nilai yang dicari adalah: X = 15
  Maka, elemen yang diperiksa: 13,16,14,21,76,21
  (tidak ditemukan!)
  Indeks larik yang dikembalikan: IDX = 0

INTODUCTION [→]

- Untuk semua algoritma pencarian berikutnya, kita menggunakan tetapan dan tipe data global berikut ini:

```
KAMUS
   const Nmaks = 100
   type Larik = array [1..Nmaks] of integer
```

```
Procedure BACA_LARIK (output A : larik, input N : integer)
Kamus
   k : integer
Algoritma
   for k ← 1 to N do
         input (A[k])
   endfor
```

TANPA BOOLEAN

- Misal: Terdapat larik ***L*** berukuran ***N*** elemen.
- Proses pembandingan dilakukan selama ***X*** tidak ditemukan (***L[k] ≠ X***) dan indeks larik belum sampai batas maksimal (***k ≠ N***).
- Nilai yang dikembalikan adalah indeks larik (***IDX***) tempat ***X*** ditemukan.
- Jika X tidak ditemukan, maka ***IDX*** diisi 0.

TANPA BOOLEAN [→]

```
Procedure CARI_IDX (input L : larik, input N : integer, input X : integer,
                    output IDX : integer)
Kamus
    k : integer

Algoritma
    k ← 1
    while (k < N) and (L[k] ≠ X) do
        k ← k + 1
    endwhile
    { k = N or L[k] = X }

    if L[k] = X then       {  X ditemukan }
        IDX ← k
    else
        IDX ← 0
    endif
```

TANPA BOOLEAN [→]

```
ALGORITMA_PENCARIAN
KAMUS
    const Nmaks = 100
    type larik : array [1..Nmaks] of integer
    A : larik
    X : integer
    IDX : integer

Procedure BACA_LARIK (output L: larik, input N: integer)
Procedure CARI_IDX (input L: larik, input N: integer, input X: integer, output IDX: integer)

ALGORITMA
    input (N)
    BACA_LARIK (A,N)

    input (X)
    CARI_IDX (A,N,X,IDX)
    if IDX = 0 then
        output (X, ' tidak ditemukan!')
    else
        output (X, ' ditemukan pada indeks larik ke- ' , IDX)
    endif
```


TANPA BOOLEAN [→]

```
Procedure CARI_X (input L : larik, input N : integer, input X : integer,
                  output ketemu : boolean)
Kamus
    k : integer

Algoritma
    k ← 1
    while (k < N) and (L[k] ≠ X) do
        k ← k + 1
    endwhile
    { k = N or L[k] = X }

    if L[k] = X then    {  X ditemukan }
        ketemu ← true
    else
        ketemu ← false
    endif
```

TANPA BOOLEAN [→]

```
ALGORITMA_PENCARIAN
KAMUS
    const Nmaks = 100
    type larik : array [1..Nmaks] of integer
    A : larik
    X : integer
    ada : boolean

Procedure BACA_LARIK (output L: larik, input N: integer)
Procedure CARI_X (input L: larik, input N: integer, input X: integer, output ketemu: boolean)

ALGORITMA
    input (N)
    BACA_LARIK (A,N)

    input (X)
    CARI_X (A,N,X,ada)
    if ada then
        output (X, ' ditemukan!')
    else
        output (X, ' tidak ditemukan!')
    endif
```

DENGAN BOOLEAN

- Variabel boolean ***ketemu*** diinisialisasi dengan nilai false.
- Pembandingan dilakukan mulai dari elemen ke- ***k*** = 1, 2, …, ***N***.
- Jika ***L[k]*** sama dengan ***X***, variabel ***ketemu*** diisi dengan nilai true dan proses pembandingan dihentikan.
- Sebailknya, jika ***L[k]*** tidak sama dengan ***X***, variabel ketemu tetap bernilai false, dan proses pembandingan dilanjutkan pada elemen berikutnya.
- Nilai yang dikembalikan adalah indeks larik (***IDX***) tempat ***X*** ditemukan. Jika ***X*** tidak ditemukan, ***IDX*** diisi dengan 0.

DENGAN BOOLEAN [→]

```
Procedure CARI_IDX (input L : larik, input N : integer, input X : integer,
                    output IDX : integer)
Kamus
    k : integer
    ketemu : boolean

Algoritma
    k ← 1
    ketemu ← false
    while (k ≤ N) and (not ketemu) do
        if L[k] = X then
            ketemu ← true
        else
            k ← k + 1
        endif
    endwhile
    { k > N or ketemu }

    if ketemu then      { X ditemukan }
        IDX ← k
    else
        IDX ← 0
    endif
```


DENGAN BOOLEAN [→]

```
Procedure CARI_X (input L : larik, input N : integer,
                  input X : integer, output ketemu : bolean)
Kamus
    k : integer

Algoritma
    k ← 1
    ketemu ← false
    while (k ≤ N) and (not ketemu) do
        if L[k] = X then
            ketemu ← true
        else
            k ← k + 1
        endif
    endwhile
    { k > N or ketemu }
```

PENGGUNAAN DALAM FUNGSI

- Secara umum, algoritma *sequential searching* tidak dinyatakan dalam prosedur, namun direalisasikan sebagai fungsi boolean atau fungsi integer yang mengembalikan indeks larik, seperti fungsi CARI_X dan CARI_IDX pada halaman berikut.

PENGGUNAAN DALAM FUNGSI [→]

```
Function CARI_IDX ( input L : larik, input N : integer,
                    input X : integer ) → integer
Kamus
    k : integer

Algoritma
    k ← 1
    while (k < N) and (L[k] ≠ X) do
        k ← k + 1
    endwhile
    { k = N or L[k] = X }

    if L[k] = X then      { X ditemukan }
        return k
    else
        return 0
    endif
```

PENGGUNAAN DALAM FUNGSI [→]

```
Function CARI_X (input L : larik, input N : integer,
                 input X : integer) → boolean

Kamus
    k : integer

Algoritma
    k ← 1
    while (k < N) and (L[k] ≠ X) do
        k ← k + 1
    endwhile
    { k = N or L[k] = X }

    if L[k] = X then     {  X ditemukan }
        return true
    else
        return false
    endif
```

15