Minggu 09

Pencarian pada Array 2

Algoritma Pemrograman – CII1F4

Fakultas Informatika

2021

Outline

- Pembahasan Latihan Soal Minggu 8 (Tambahan)
- Pencarian Pada Array Terurut (Binary Search)
- Studi Kasus Binary Search
- Studi Kasus Pencarian pada Array (Lanjutan)

Pembahasan Latihan Soal Minggu 8

(Tambahan)

Pembahasan Soal 1: Mahasiswa

Sebuah program digunakan untuk melaporkan data wisudawan di Universitas Telkom. Data yang disimpan adalah nama, nim, eprt, semester, dan ipk.

Masukan terdiri dari beberapa baris, yang masing-masing barisnya menyatakan nama, nim, eprt, jumlah semester dan ipk dari seorang wisudawan. Masukan berakhir apabila nim adalah “none”.

Keluaran berupa 3 bilangan yang menyatakan eprt tertinggi, ipk terendah, dan rata-rata semester lulusan.

Buatlah program dengan menggunakan subprogram

```
1. procedure untuk pengisian array wisudawan
2. function untuk mencari:
   a. eprt tertinggi
   b. ipk terendah
   c. rata-rata semester lulusan
3. Program utamanya
4. Asumsikan kapasitas dari arraynya adalah 1000.
```

Solusi Soal 1 [a]

- Definisi kan dulu tipe array dan tipe bentukan pertama kali pada program utama.

  Pada kasus ini adalah tipe array bentukan lulusan, asumsi ada 1000 elemen

```
program mahasiswa
kamus
    constant nMAX : integer = 1000
    type lulusan <
        nama, nim : string
        semester : integer
        eprt,ipk : real
    >
    type tabLulus : array [0..nMAX-1] of lulusan
```

- Selalu kerjakan algoritma program utama terakhir, setelah semua subprogram selesai diisi

Solusi Soal 1 [b]

```
procedure isiData(in/out t:tabLulus, n:integer)
{I.S. data lulusan telah siap pada piranti masukan
 Proses: data lulusan dimasukan ke dalam array t, selama nim != “none” dan array belum penuh
 F.S. array t berisi n data mahasiswa atau lulusan }
kamus
    nim : integer
algoritma
    n <- 0
    input(nim)
    while nim != "none" and n <= nMAX do
        n <- n + 1
        t[n-1].nim <- nim
        input(t[n-1].nama, t[n-1].eprt, t[n-1].semester,t[n-1].ipk) {input data yang tersisa}
        {indeks t selalu n dikurangi 1, contoh: untuk mahasiswa ke-2 (n=2) maka indeksnya t
         ke-2 adalah 1 (t[1]). karena indeks array t dimulai dari 0}

        input(nim)                    {input data nim selanjutnya}
    endwhile
endprocedure
```

Solusi Soal 1 [c]

```
function maxEPRT(t:tabLulus, n : integer) -> real
{diberikan array t yang berisi n data lulusan,
untuk mengembalikan nilai EPRT tertinggi}
kamus
    max : real
    i : integer
algoritma
    max <- t[0].eprt
    i <- 1
    while i < n do
        if max < t[i].eprt then
            max <- t[i].eprt
        endif
        i <- i + 1
    endwhile
    return max
endfunction
```

```
function minIPK(t:tabLulus, n : integer) -> real
{diberikan array t yang berisi n data lulusan, untuk
mengembalikan nilai IPK terendah}
kamus
    min : lulusan
    {versi lain, min tipenya lulusan bukan real}
    i : integer
algoritma
    min <- t[0]      {disimpan data orang pertama}
    i <- 1
    while i < n do
        if min.ipk < t[i].ipk then    {perhatikan min.ipk}
            min <- t[i]
        endif
        i <- i + 1
    endwhile
    return min.ipk    {perhatikan min.ipk}
endfunction
```

Solusi Soal 1 [d]

```
function rataan(t: tabLulus, n : integer) -> real
{diberikan array t yang berisi n data lulusan,
untuk mengembalikan rata-rata semester}
kamus
    jum : real
    i : integer
algoritma
    jum <- 0
    i <- 1
    while i <= n do
        jum <- jum + t[i].semester
        i <- i + 1
    endwhile
    return jum / n
endfunction
```

```
program mahasiswa
kamus
    constant nMAX : integer = 1000
    type lulusan <
        nama, nim : string
        semester : integer
        eprt,ipk : real >

    type tabLulus : array [0..nMAX-1] of lulusan
    dataMHS : tabLulus
    nMHS : integer
    eprtTertinggi, ipkTerendah, rerataSemester : real
algoritma
    isiData(dataMHS, nMHS)
    eprtTertinggi <- maxEPRT(dataMHS,nMHS)
    ipkTerendah <- minIPK(dataMHS,nMHS)
    rerataSemester <- rataan(dataMHS,nMHS)
    print(eprtTertinggi, ipkTerendah, rerataSemester)
endprogram
```

Pembahasan Soal 2: Bunga

Diasumsikan suatu array dengan kapasitas 100 telah berisi sejumlah N data bunga. Buatlah subprogram berikut:

a. procedure untuk melakukan rename nama bunga tertentu.

b. procedure delete data bunga dengan nama tertentu. (Geser elemen array untuk mengisi elemen yang kosong setelah proses delete tersebut)

Catatan:

- Tambahkan function pencarian untuk mempermudah proses rename dan delete.
- Tampilan “Bunga tidak ditemukan” apabila nama bunga yang dicari tidak ada.

Jawaban:

- Yang disimpan adalah daftar nama bunga, sehingga yang perlu dibuat adalah tipe array of string saja

```
kamus
        type tabBunga : array [0..100] of string
```

Solusi Soal 2 [a]

Subprogram untuk pencarian bunga, nilai yang dikembalikan adalah indeks hasil pencarian

```
function cari(t:tabBunga, n:integer, bunga:string) -> integer
{diberikan array t yang berisi n data dan sebuah nama bunga, untuk mengembalikan
indeks bunga di dalam t apabila ketemu, atau -1 apabila sebaliknya}
kamus
    i, found : integer
algoritma
    found <- -1
    i <- 0
    while i < n and found == -1 do
        if t[i] == bunga then
            found <- i
        endif
        i <- i + 1
    endwhile
    return found
endfunction
```

```
function cari2(t:tabBunga, n:integer, bunga:string) -> integer
{Variasi lain algoritma Sequential Search}
kamus
    i : integer
algoritma
    i <- 0
    while i < n and t[i] != bunga do
        i <- i + 1
    endwhile
    {data ketemu jika i < n, dan tidak ketemu jika i == n}
    if i < n then      {kenapa bukan t[i] == bunga???}
        return i
    else
        return -1
    endif
endfunction
```

Solusi Soal 2 [b]

```
procedure rename(in/out t:tabBunga, in n:integer, X:string)
{IS. terdefinisi suatu array t yang berisi n nama bunga, dan suatu
string X
 FS. merename bunga dengan nama X apabila ditemukan, atau
menampilkan "Bunga tidak ditemukan"}
kamus
    found : integer
algoritma
    found <- cari(t,n,X)
    if found == -1 then
        print("Bunga tidak ditemukan")
    else
        input(t[found])
    endif
endprocedure
```

Solusi Soal 2 [c]

```
procedure delete(in/out t:tabBunga, n:integer, in X:string)
{IS. terdefinisi suatu array t yang berisi n nama bunga, dan suatu string X
 FS. menghapus bunga dengan nama X apabila ditemukan, atau menampilkan "Bunga tidak ditemukan"}
kamus
    found, i : integer
algoritma
    found <- cari(t,n,X)
    if found == -1 then
        print("Bunga tidak ditemukan")
    else
        {timpa data yang mau dihapus dengan data setelahnya, data ke-i dengan data ke i+1}
        i <- found
        while i <= n-2 do    {indeks data terakhir adalah n-1 akan menimpan data ke n-2}
            t[i] <- t[i+1]
            i <- i + 1
        endwhile
        n <- n - 1           {jumlah data berkurang 1, karena proses hapus}
    endif
endprocedure
```

Bagaimana apabila X adalah data terakhir (indeks ke n-1) ?

Pencarian Pada Array Terurut

(Binary Search)

Pengantar Binary Search

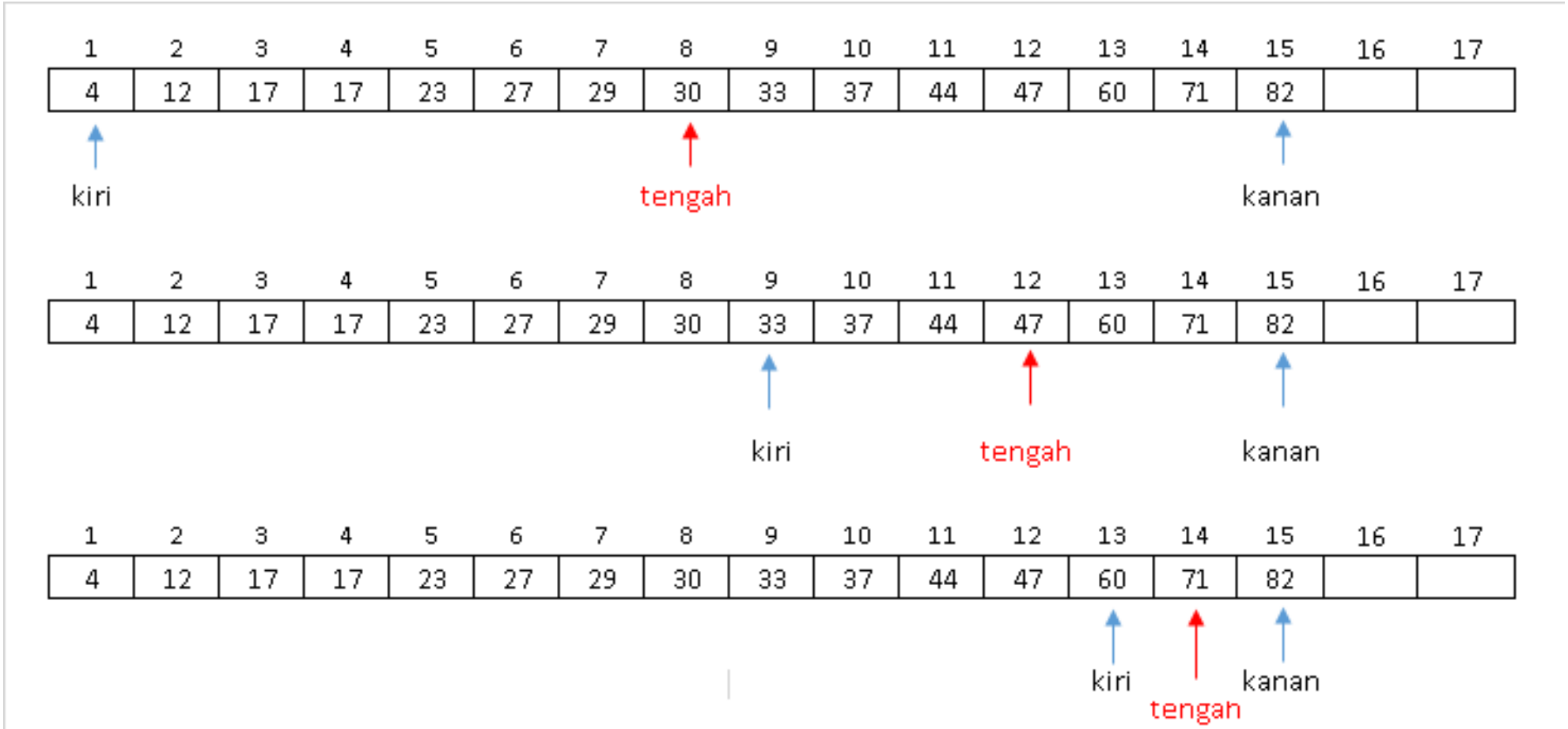
- Algoritma pencarian yang dipelajari sebelumnya dilakukan pada tabel acak. Walaupun bisa diaplikasikan ke tabel terurut, tapi tidak optimal.

- Perhatikan tabel terurut di bawah ini. Misalkan akan dicari X=26, dengan algoritma **Sequential Search**, proses pencarian akan dilakukan sampai elemen terakhir. Seharusnya pencarian berakhir di indeks ke-6, karena sudah bisa diketahui bahwa nilai 26 tidak ada pada tabel.

1	2	3	4	5	6	7	8	9	10	11	12	13	14
4	12	17	17	23	27	29	30	33	37	44	47	60	81

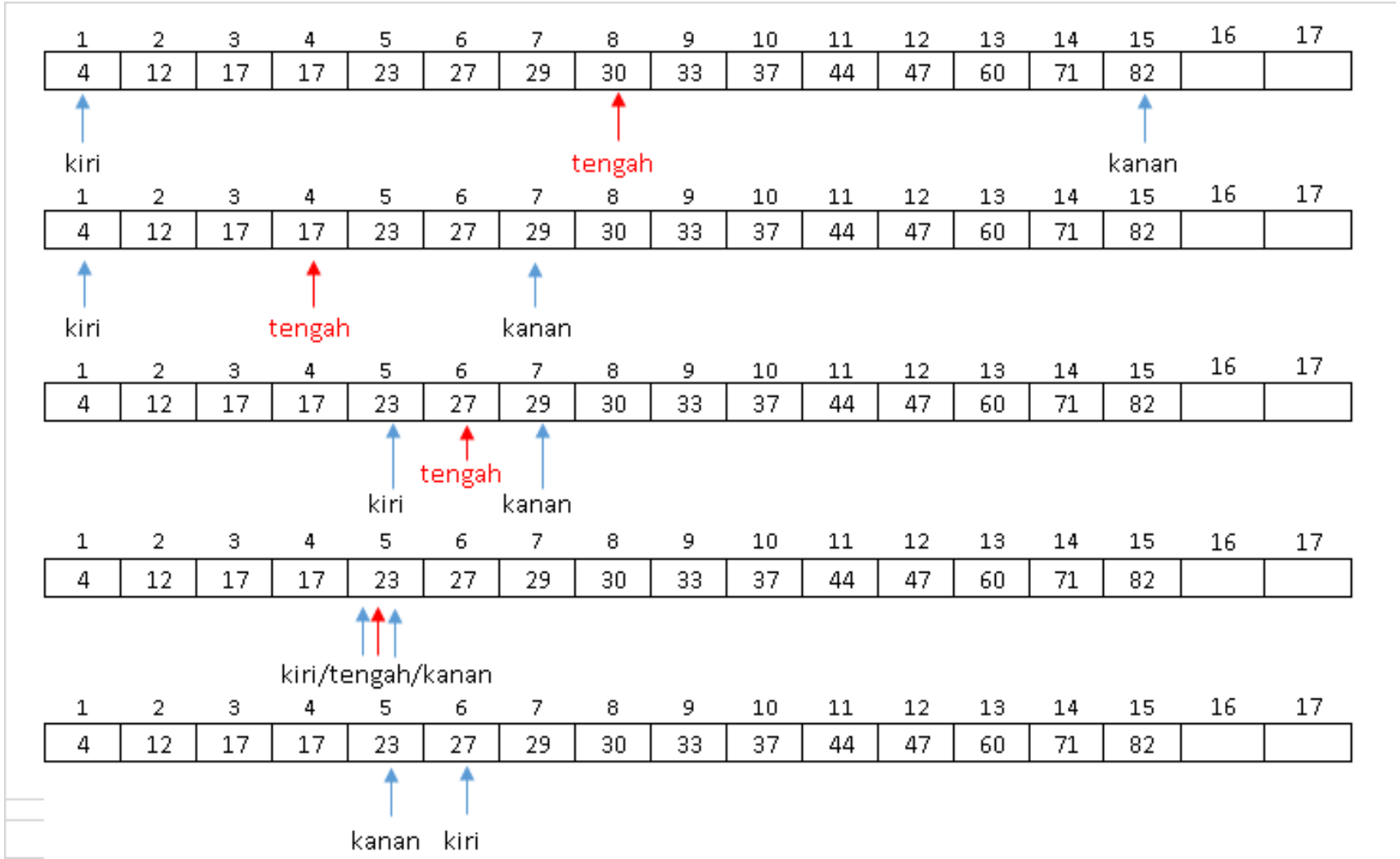
Bagaimana dengan Binary Search?

Perhatikan ilustrasi di bawah ini untuk menemukan angka 71. Hanya perlu tiga iterasi untuk menemukan angka 71.



Bagaimana dengan Binary Search?

Perhatikan ilustrasi di bawah ini untuk menemukan angka 25



Tidak ada angka 25 pada tabel

Algoritma Binary Search 1

```
type tabInt : array [1..1000] of integer    {indeks dari 1}

function binSearch1(tab:tabInt, n, x : integer) -> boolean
{mengembalikan TRUE apabila x ada di dalam array tab yang berisi n bilangan,
tab terurut membesar atau ASCENDING}
kamus
    left, right, mid : integer
algoritma
    left <- 1          {0 apabila indeks dari 0}
    right <- N         {N-1 apabila indeks dari 0}
    mid <- (left + right) div 2
    while left <= right and tab[mid] != x do
        if x < tab[mid] then
            right <- mid - 1
        else
            left <- mid + 1
        endif
        mid <- (left + right) div 2
    endwhile
    return tab[mid] == x
endfunction
```

Algoritma Binary Search 2

```
function binSearch2(tab:tabInt, n, x : integer) -> integer
{mengembalikan indeks pencarian apabila x ada di dalam array tab yang berisi n
bilangan atau -1 apabila tidak ditemukan, tab terurut membesar atau ASCENDING}
kamus
    left, right, mid : integer
    found : integer
algoritma
    left <- 1          {0 apabila indeks dari 0}
    right <- N         {N-1 apabila indeks dari 0}
    found <- -1
    while left <= right and found == -1 do
        mid <- (left + right) div 2
        if x < tab[mid] then
            right <- mid - 1
        else if x > tab[mid] then
            left <- mid + 1
        else
            found <- mid
        endif
    endwhile
    return found
endfunction
```

Bagaimana jika tab terurut DESCENDING?

Studi Kasus 1: Array Mahasiswa

```
function cariMhs(A:tabMHS, N: integer, NIM : integer) -> integer
{mengembalikan indeks pencarian mahasiswa pada array A dengan NIM tertentu. -1
apabila tidak ditemukan. Array A terurut secara DESCENDING berdasarkan NIM}
kamus
    left, right, mid : integer
    found : integer
algoritma
    left <- 0
    right <- N-1
    found <- -1
    while left <= right and found == -1 do
        mid <- (left + right) div 2
        if NIM > A[mid].nim then
            right <- mid - 1
        else if NIM < A[mid].nim then
            left <- mid + 1
        else
            found <- mid
        endif
    endwhile
    return found
endfunction
```

A terurut berdasarkan NIM dan pencarian berdasarkan NIM

```
constant nMAX : integer = 1000
type Mahasiswa <
    nama: string
    nim : integer
    eprt,ipk : real
>
type tabMHS : array [0..nMAX-1] of Mahasiswa
{indeks dari 0}
```

Studi Kasus 2: Array Mahasiswa

```
function cariMhs(A:tabMHS, N: integer, IPK : real) -> integer
{mengembalikan indeks pencarian mahasiswa pada array A dengan IPK tertentu. -1
apabila tidak ditemukan. Array A terurut secara DESCENDING berdasarkan NIM}
kamus
    k : integer
algoritma
    k <- 0
    while k < N and A[k].ipk != IPK do
        k <- k + 1
    endwhile
    if k < N then
        return k
    else
        return -1
    endif
endfunction
```

Jika A terurut berdasarkan NIM dan pencarian berdasarkan IPK.

Maka gunakan Sequential Search

Kesimpulan

1. Sequential Search tidak optimal digunakan untuk data yang terurut.
2. Algoritma Binary Search untuk data terurut ascending dan descending berbeda (dalam hal bergesernya indeks left dan right)
3. Untuk array of tipe bentukan, maka data yang menjadi kata kunci pencarian harus sama dengan acuan pengurutan array.

   Contoh:

   - Apabila array mahasiswa terurut berdasarkan IPK, maka kata kunci untuk pencarian dengan binary search adalah IPK juga.
   - Apabila array mahasiswa terurut berdasarkan NIM, maka pencarian berdasarkan IPK hanya bisa dilakukan menggunakan sequential search

Studi Kasus Pencarian pada Array

(Lanjutan)

Contoh Soal : Mode

Sebuah array (kapasitas 1080) digunakan untuk menampung sekumpulan bilangan bulat. Dimana bilangan-bilangan tersebut bernilai 0 s.d. 10.

Buatlah subprogram yang digunakan untuk mencari modus dari kumpulan bilangan tersebut! Modus adalah bilangan yang paling banyak muncul, Asumsi hanya terdapat satu bilangan dengan kemunculan terbanyak.

```
{Deklarasi konstanta dan type ------------------------------------------ }
constant nMAX : integer = 1080
type tabInt : array [0..nMAX-1] of integer

{Daftar Subprogram ----------------------------------------------------- }
function frekuensi(t:tabInt, n:integer, x:integer) -> integer
{mengembalikan banyaknya kemunculan x pada array t yang berisi n bilangan}

procedure membuatTabel(in t1:tabInt, n:integer, in/out t2:tabInt, m:integer)
{I.S. terdefinisi array t1 yang berisi n buah bilangan bulat
 F.S. t2 berisi frekuensi bilangan 0 hingga m, m = 10, bilangan i memiliki frekuensi di t2[i]}

function modus(t:tabInt, n:integer) -> integer
{mengembalikan nilai yang paling banyak muncul pada array t yang berisi n bilangan bulat}
```

Pembahasan Soal : Mode [a]

- Ide Pencarian Modus: Buat daftar frekuensi dari semua bilangan yang muncul, kemudian dari daftar tersebut cari yang memiliki frekuensi paling banyak
- Contoh:

Array	8	5	2	4	9	5	5	1	2	0	6	5	3	8

- Daftar Frekuensi (asumsi bilangan 0 sampai 10)

Tabel Frekuensi	1	1	2	1	1	4	1	0	2	1	0
indek/bilangan (0-10)	0	1	2	3	4	5	6	7	8	9	10

- Frekuensi terbanyak adalah bilangan 5, dengan kemunculan 4,

  maka **modus adalah 5**

Pembahasan Soal : Mode [b]

```
function frekuensi(t:tabInt, n:integer, x:integer) -> integer
{mengembalikan banyaknya kemunculan x pada array t yang berisi n bilangan}
kamus
    count, i : integer
algoritma
    count <- 0
    i <- 0
    while i < n do
        if x == t[i] then
            count <- count + 1
        endif
        i <- i + 1
    endwhile
    return count
endfunction
```

```
procedure membuatTabel(in t1:tabInt, n:integer, in/out t2:tabInt,
m:integer)
{I.S. terdefinisi array t1 yang berisi n buah bilangan bulat
 F.S. t2 berisi frekuensi bilangan 0 hingga m, m = 10, bilangan i
memiliki frekuensi di t2[i] }
kamus
    count,i : integer
algoritma
    i <- 0
    m <- 10
    while i <= m do
        t2[i] <- frekuensi(t1,n,i)
        i <- i + 1
    endwhile
endprocedure
```

Pembahasan Soal : Mode [c]

```
function modus(t:tabInt, n:integer) -> integer
{mengembalikan nilai yang paling banyak muncul pada array t yang berisi n bilangan bulat}
kamus
    tabFreg : tabInt
    m, idx_max, i : integer
algoritma
    membuatTabel(t,n,tabFreg,m)    {membuat table frekuensi dari 0 sampai m}
    idx_max <- 0                   {Mencari indeks dari frekuensi terbanyak}
    i <- idx_max + 1
    while i <= m do
        if tabFreg[idx_max] < tabFreg[i] then
            idx_max <- i
        endif
        i <- i + 1
    endwhile
    return idx_max
Endfunction
```

Soal 1 : Anggota Himpunan

Sebuah tipe himpunan adalah array of integer yang berkapasitas 100.

Buatlah subprogram untuk mengecek suatu himpunan adalah valid atau tidak. (Himpunan dikatakan valid apabila tidak ada anggota himpunan yang sama)

Contoh :

```
A = {11, 28, 33, 64, 95, 16} adalah valid, sedangkan
B = {11, 28, 33, 64, 95, 16, 100, 28, 33, 64, 95, 16} tidak valid
```

Petunjuk:

1. Buat tabel frekuensi, beri asumsi range bilangan pada anggota himpunan
2. Cek table tersebut, harusnya maksimal jumlah kemunculannya setiap bilangan yang ada pada himpunan adalah 1.
3. Gunakan subprogram sesuai kebutuhan dengan tepat

Soal 2 : Irisan Himpunan

Sebuah tipe himpunan adalah array of integer yang berkapasitas 100.

Buatlah subprogram untuk mencari irisan dari dua buah himpunan

```
A = {11, 28, 33, 64, 95, 16, 100, 15} dan B = {3, 11, 7, 28, 33, 6}
Irisan A dan B adalah {11, 28, 33}
```

Petunjuk:

1. Asumsi himpunan valid
2. Cek setiap anggota himpunan array A apakah ada di dalam array B (atau sebaliknya)
3. Apabila ada, maka tambahkan anggota tersebut ke array Irisan.
4. Gunakan subprogram sesuai kebutuhan dengan tepat

Soal 3 : Gabungan Himpunan

Sebuah tipe himpunan adalah array of integer yang berkapasitas 100.

Buatlah subprogram untuk mencari irisan dari dua buah himpunan

```
A = {11, 28, 33, 64, 95, 16, 100, 15} dan B = {3, 11, 7, 28, 33, 6}
Gabungan A dan B adalah {11, 28, 33, 64, 95, 16, 100, 15, 3, 7, 6}
```

Petunjuk:

1. Asumsi himpunan valid
2. Copy isi array A ke array Gabungan
3. Cek setiap anggota array B, apakah merupakan anggota himpunan array Gabungan.
4. Apabila tidak, maka tambahkan anggota array B tersebut ke array Gabungan.
5. Gunakan subprogram sesuai kebutuhan dengan tepat

TERIMA KASIH