


```
procedure populateTabelMark( input/output T: Tabel)
{I.S: T terdefinisi }
{F.S: T terisi sejumlah elemen }
{proses: mengisi elemen dengan masukan keyboard berulang hingga masukan bernilai <=0}
```

## Kamus Lokal

i : integer {iterator}  
x : integer {variabel penampung masukan keyboard}

## Algoritma

~~T.size <-- N~~   
i traversal 1..T.size  
  input x  
  if x > 0  
    return x = x - 1  
  else {x <= 0}  
    return x

i <-- 0  
input x {first element}  
while (x>0 AND i<10) do  
  i <-- i+1  
  T.wadah [ i ] <-- x  
  input x {next element}  
{end while}  
T.size <-- i

i <-- 0 {inisialisasi}  
iterate  
  input x  
stop (x<=0 OR i=10)  
  i <-- i+1 {next element}  
  T.wadah[ i ] <-- x  
T.size <-- i

function getNewTable() --> Tabel

{mengembalikan tabel dengan size=0, setiap elemen wadah=-999}

kamus lokal

i : integer {iterator} ✓

T : Tabel

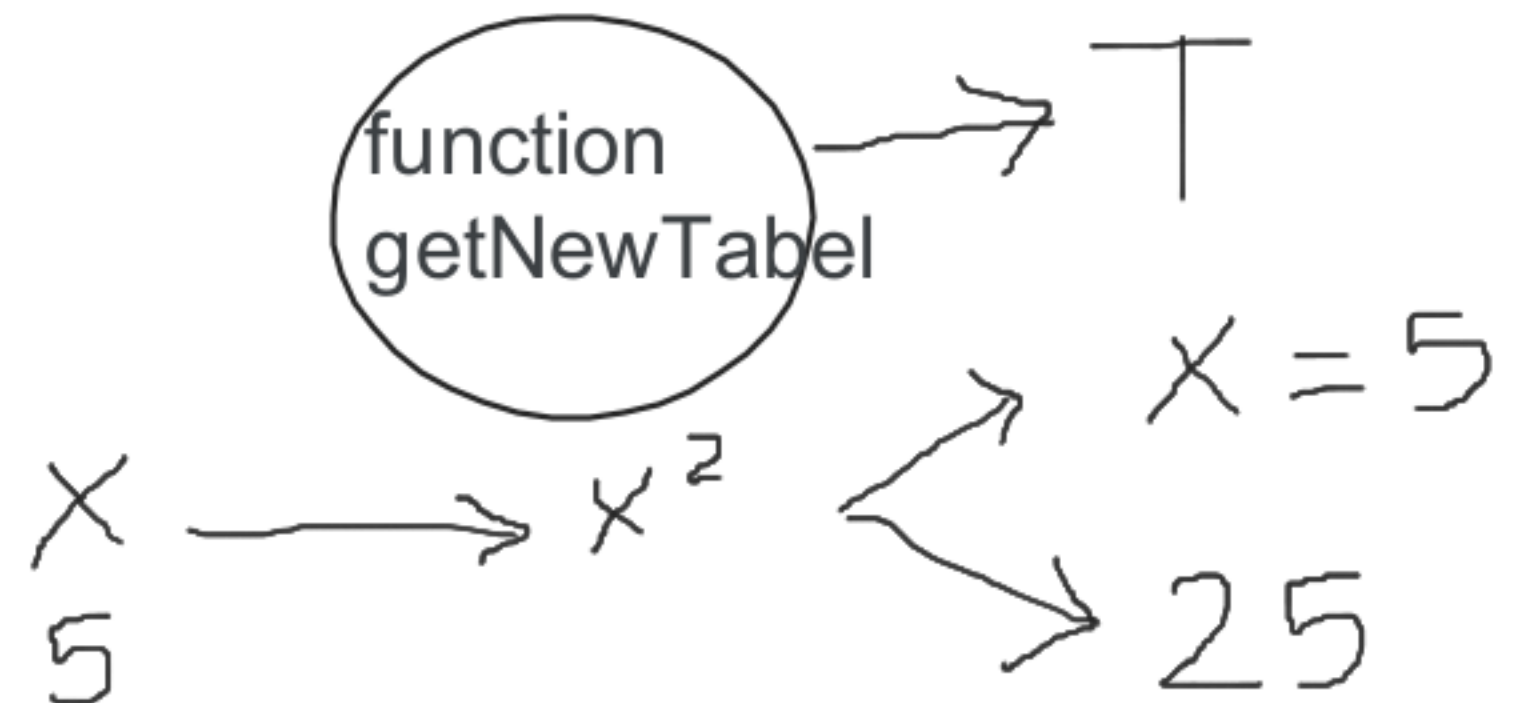
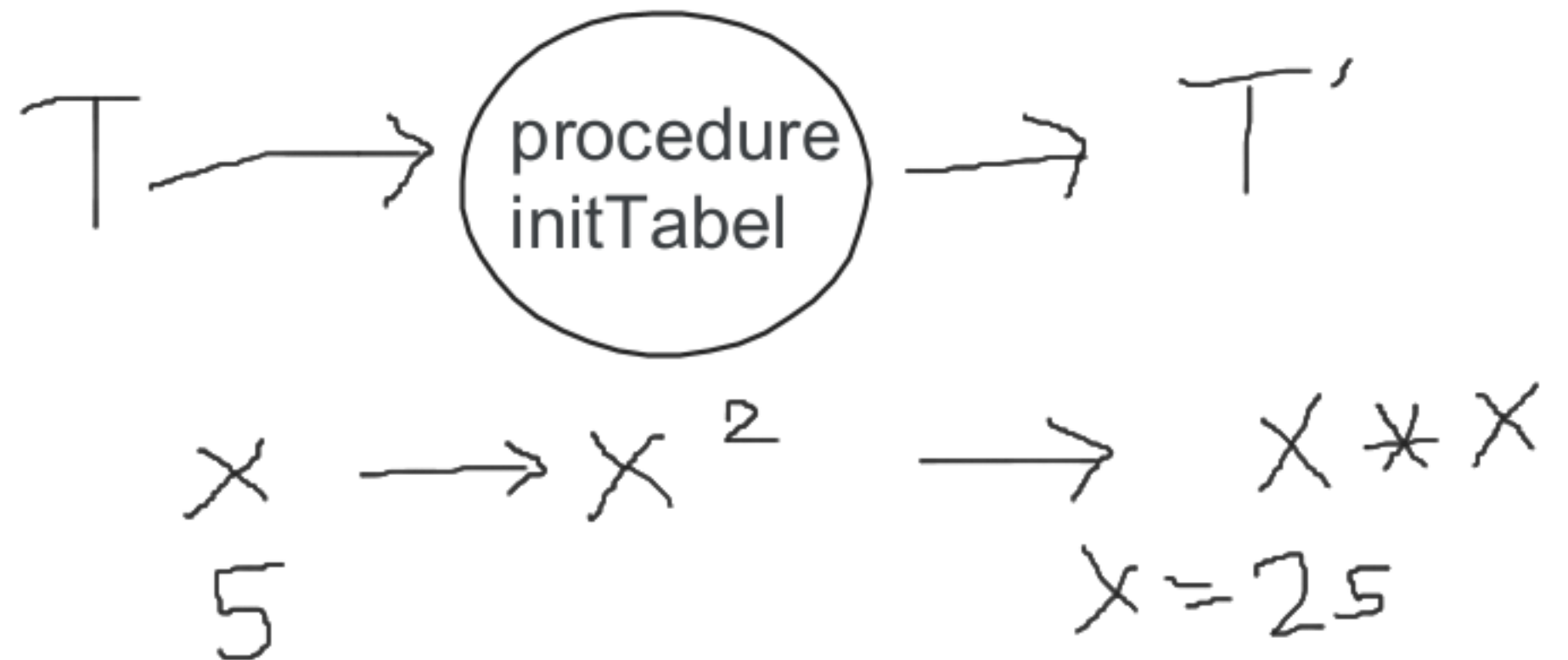
algoritma

T.size <-- 0

i traversal 1..10

T.wadah[i] <-- -999

--> T



Bismillah

Semoga bener aamiin...  
✓✓✓✓✓✓✓✓



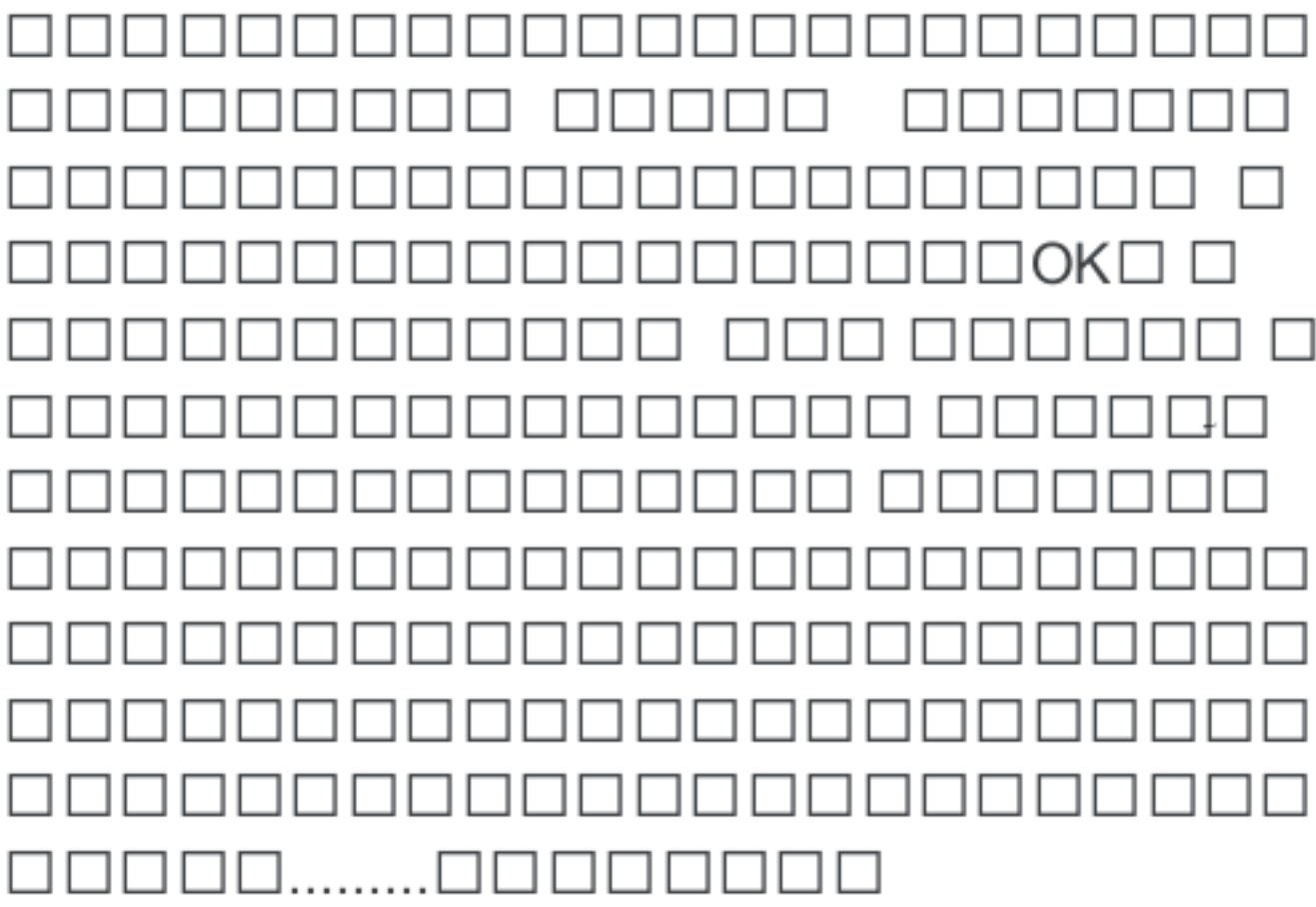
```
procedure viewTable ( input T:Tabel )  
  {I.S.: T terdefinisi}  
  {F.S.: beberapa elemen T telah tampil ke layar}  
  {Proses: menampilkan elemen T yang terisi ke layar}
```

```
procedure printTable ( input T:Tabel )  
  {I.S.: T terdefinisi}  
  {F.S.: semua elemen T telah tampil ke layar}  
  {Proses: menampilkan semua elemen T ke layar}  
kamus lokal
```

```
  i : integer {iterator}  
Algoritma  
  i traversal 1 ... 10  
  output T.wadah[ i ]
```

```
kamus lokal  
  i : integer {inisiator}  
  n : integer {indeks tabel yang ingin ditampilkan}
```

```
algoritma  
  i traversal 1..T.size ✓  
  if T.wadah[i] == T.wadah[n] then  
    output T.wadah[i]  
  else  
    i +=1
```



```

{ *** Test tabel kosong *** }
function isEmpty (T : Tabint) → boolean
{ Mengirimkan true jika tabel kosong, mengirimkan false jika tidak }

```

```

// Kamus Lokal
cek : boolean
// algoritma
if T.size != 0 then
    cek <-- false
else then
    cek <-- true
-->(cek)

```

jadinya --  
> ??? iya :')||  
kayaknya si  
begitu  
wkwk

(\\_\\_/\_/)

(= ' ' =)

(")\_(")

(\\_\\_/\_/)

(= ' ' =)

(")\_(")

(\\_\\_/\_/)

(= ' ' =)

(")\_(")

(\\_\\_/\_/)

(= ' ' =)

(\\_\\_/\_/)

```

function countEmpty(T: Tabel) --> integer
{I.S. : T terdefinisi}
{F.S. : mengembalikan banyak elemen T yang belum terisi}
{proses : menghitung banyaknya elemen T yang belum terisi}

```

kamus lokal

i : integer {iterator}

temp: ~~integer~~

```

algoritma
if T.size == 0 then
    temp <-- 0
else
    temp <-- 0
    i traversal 1..10
    if T.wadah[i] == 0 then
        temp <--- temp + 1
-->(temp)

```

- ggg

=

tabel kosong itu  
maksudnya, isi  
tabelnya 0 semua  
kan? kayaknya iya  
dah beda ngga sih o  
sama kosong tuh??  
|| iya 0 semua

function isFullTable( T: Tabel) -> boolean  
{mengembalikan True jika T penuh }

kamus lokal

i : integer (iterator)

full : boolean

algoritma

full <- true

i traversal 1..T.size

if T.wadah[i] == 0 then

full <- false

-> full

-> full

if T.size=10 then

full <-- true

else

full <-- false

-->full

function getSize( T: Tabel) -> integer  
{mengembalikan banyak elemen pengisi T }

kamus lokal

counter : integer

i : integer (iterator)

algoritma

counter <-- 0

i traversal 1..10

if T.wadah[i] != 0 then

counter <-- counter + 1

-> counter

--> T.size

dah bener  
kan ini????



procedure searchX1 (input T: Tabel, input X: integer, output Pos: integer )

{I.S.: T terdefinisi, X terdefinisi }

{F.S.: Pos berisi posisi pertama ketemu X di T.wadah, atau -999 jika tidak ketemu }

{Proses: mencari elemen bernilai X dalam T.wadah}

kamus lokal

i : integer{iterator}

~~pos: integer~~

algoritma

~~T.size <-- 0~~

i traversal 1...999

if T.wadah[ i ] != X then

pos <-- pos + 1

else

--> pos

bayangkan T=[2,3,1,6,4,9,3,0,0,0]

nilai yang dicari X = 4 , maka posisi ketemu= 5

Pos <-- -999

if (T.size != 0) then

i <-- 1

while (T.wadah[i] != X) AND (i<T.size) do

i <-- i + 1

{end while}

if (T.wadah[i] = X) then

Pos <--i

silakan yg menghapus bertanggung  
jawab mengisi kembali

```
procedure countX ( input T:Tabel, input X: integer, output Byk: integer )  
  {I.S.: T terdefinisi, X terdefinisi }  
  {F.S.: Byk berisi banyaknya X di T.wadah, atau 0 jika tidak ketemu }  
  {Proses: menghitung elemen bernilai X dalam T.wadah}
```

```
procedure countX ( input T:Tabel, input X: integer, output Byk: integer )  
  {I.S.: T terdefinisi, X terdefinisi }  
  {F.S.: Byk berisi banyaknya X di T.wadah, atau 0 jika tidak ketemu }  
  {Proses: menghitung elemen bernilai X dalam T.wadah}
```

Kamus Lokal

```
  i : integer { iterator }  
  sum : integer { jumlah dari input X pada tabel }
```

Algoritma

```
  sum <- 0 { inisial nilai sum }  
  i transversal 1..10  
    if T.wadah[ i ] = X then  
      sum <- sum + 1 { setiap kali menemukan nilai X pada wadah nilai sum bertambah 1 }
```

```
  Byk <- sum { memasukkan hasil sum ke Byk }
```

replace sum dengan Byk

halo kalo ada yang merasa perlu di ganti  
monggo ditambahin : )

```
function getMaxElm ( T: Tabel) -> integer
    {mengembalikan nilai elemen terbesar }
//kamus lokal
i : integer
max: integer
```

```
//algoritma
if (T.size>0) then

    max <-- T.wadah[1] {inisiasi max}
    i traversal 2..T.size
        if T.wadah[i] > max then
            max <-- T.wadah[i]
        --> max
else {tabel kosong}
    --> ???
```

```
function getMinElm ( T: Tabel) -> integer
    {mengembalikan nilai elemen terkecil, tapi bukan -999 }
//kamus lokal
i:integer {iterasi}
small:integer {temp storage}

//algoritma
small<--T.wadah[1] {inisiasi small}

i traversal 2...T.size
    if(T.wadah[i]<small & T.wadah[i]!=-999)then
        small<--T.wadah[i]

    --> small
```



```
procedure addX ( input/output T:Tabel, input X: integer )  
  {I.S.: T terdefinisi, X terdefinisi }  
  {F.S.: isi T.wadah bertambah 1 elemen jika belum penuh}  
  {Proses: mengisi elemen T.wadah dengan nilai X}
```

Kamus Lokal :  
 i : integer {iterator}

Algoritma : **then**  
if ~~T.size != 10~~ {cek tabel belum penuh (asumsi 10 sebagai nilai maksimal dari tabel)}  
 i traversal 1..10  
 if T.wadah[i] = 0 {Mencari indeks wadah yang berisikan 0} then  
 T.wadah[i] <-- X {Mengisi elemen T.wadah yang kosong dengan X}  
 T.size <-- T.size + 1 {menambah isi T.wadah sebanyak 1 elemen  
 karena masih kosong}

**T.size <- T.size+1**  
**T.wadah[T.size] <- X**

dah bener blom  
gais, ada yg mau  
ditambahin?

**mantap**

kerjain kapan lagi gais?

makan duls

procedure delX ( input/output T:Tabel, input X: integer )  
 {I.S.: T terdefinisi}  
 {F.S.: isi T.wadah berkurang 1 elemen jika belum kosong}  
 {Proses: menghapus 1 elemen bernilai X, geser semua elemen sisa}

j traversal i..9  
 T.wadah[j] <- T.wadah[j+1]  
 {end traversal, j=9}  
 T.wadah[10] <-- 0

## Kamus lokal

i: integer {iterator}

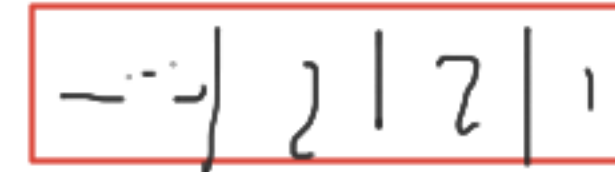
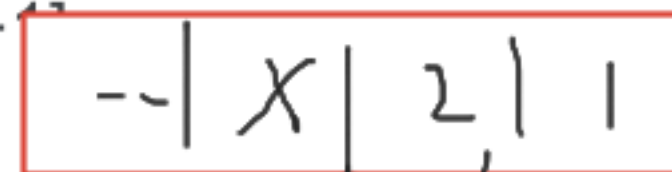
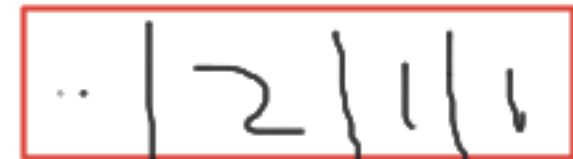
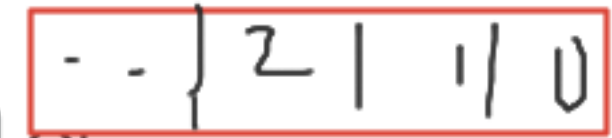
j: integer {iterator}

## Algoritma

```

if T.size != 0 then {Tabel tidak kosong}
  i <-- 1
  while i < T.size and T.wadah[i] != X do {cari indeks i dimana elemennya sama dengan X}
    i <-- i + 1
  {end while: i=T.size OR T.wadah[i]=X}
  if i < T.size then
    {ada elemen yang nilainya sama dengan X}
    j traversal i..T.size
    {Geser elemen ke kiri, elemen X dihapus}
    if j = 10 then
      {Index akhir, tidak ada elemen berikutnya}
      T.wadah[j] <-- 0
    else
      T.wadah[j] <-- T.wadah[j+1]
    T.size <-- T.size-1
  
```

anggap aja 3  
cell terakhir



procedure delAllX ( input/output T:Tabel, input X: integer )

{I.S.: T terdefinisi}

{F.S.: isi T.wadah berkurang semua elemen bernilai X jika belum kosong}

{Proses: menghapus semua elemen bernilai X, geser elemen sisa}

panggil delX sebanyak countX kali

procedure updateX ( input/output T:Tabel, input X: integer, input Y: integer )

{I.S.: T terdefinisi,  $Y \geq 0$ }

{F.S.: isi T.wadah bernilai X diganti nilainya dengan Y}

{Proses: mengganti elemen pertama bernilai X dengan nilai Y }



















