

Type Tstack = <wadah:array[1..10] of character, top:integer>

Procedure CreateStack(output S:Tstack) 2
{I.S: - ; F.S: S terdefinisi }

{Proses mengisi elemen wadah kosong dengan '_', top=0}

Function Infotop(S:Tstack) -> character 3
{mengembalikan nilai elemen puncak}

Function Top(S:Tstack) -> integer 4
{mengembalikan posisi puncak}

Function isEmptyStack(S:Tstack) -> boolean 5
{mengembalikan true bila S kosong}

Function isFullStack(S:Tstack) -> boolean 6
{mengembalikan true bila S penuh}

Procedure Push(input/output S:Tstack, input e:character) 7
{I.S: S,e terdefinisi, S mungkin kosong }

{F.S: S tetap, atau infotop(S)=e }

{Proses mengisi elemen e ke puncak S, bila belum penuh}

Procedure Pop(input/output S:Tstack, output e:character) 8
{I.S: S terdefinisi, mungkin kosong }

{F.S: S tetap, atau e berisi infotop(S) lama }

{Proses menghapus elemen e dari puncak S, bila belum kosong}

Procedure PrintStack(input S:Tstack) 9
{I.S:-; F.S:-; Proses: menampilkan info elemen S }

**tugas kelas:
membuat
realisasi/body
fungsi/operator
untuk ADT Stack.**

Procedure CreateStack(output S:Tstack)

{I.S: - ; F.S: S terdefinisi }

{Proses mengisi elemen wadah kosong dengan '_', top=0}

Kamus lokal

i : integer {iterator}

Algoritma

S.top <-- 0

i traversal [1...10]


S.wadah[i] <-- '_'

Function Infotop(S:Tstack) -> character
{mengembalikan nilai elemen puncak}

Kamus lokal
{ - }

Algoritma

if (S.top != 0) then
 --> S.wadah[S.top]
else
 --> '_'

A handwritten checkmark is drawn next to the 'else' branch. A horizontal line is drawn under the 'else' branch, extending to the right.

~~Function~~ Top(S:Tstack) -> integer
{mengembalikan posisi puncak}

Kamus Lokal

Algoritma
--> S.top

~~Function~~ isEmptyStack(S:Tstack) -> ~~boolean~~
{mengembalikan true bila S kosong}

Kamus Lokal

Algoritma

~~if~~ Top(S) = ~~0~~ then
 --> true
~~else~~
 --> false



~~Function~~ isFullStack(S:Tstack) -> ~~boolean~~
{mengembalikan true bila S penuh}

Kamus Lokal

Algoritma

```
if Top(S) = 10 then  
    -> true  
else  
    -> false
```

Procedure Push(input/output S:Tstack, input e:character)
{I.S: S,e terdefinisi, S mungkin kosong }
{F.S: S tetap, atau infotop(S)=e }
{Proses mengisi elemen e ke puncak S, bila belum penuh}

kamus lokal

algoritma

if (not isFullStack(S)) then
 S.top <- S.top + 1
 S.wadah[Top(S)] <- e

Procedure Pop(input/output S:Tstack, output e:character)

{I.S: S terdefinisi, mungkin kosong }

{F.S: S tetap, atau e berisi infotop(S) lama }

{Proses menghapus elemen e dari puncak S, bila belum kosong}

kamus lokal

algoritma

if (not isEmptyStack(S)) then

 e <- Infotop(S)

 S.wadah[Top(S)] <- '_'

 S.top <- S.top - 1

else {stack kosong}

 e <-- '_'

Procedure PrintStack(input S:Tstack)

{I.S:-; F.S:-; Proses: menampilkan info elemen S }

Kamus Lokal

i : integer {iterator}

Algoritma

i traversal [1...10]

output (S.wadah[i])

{ bg, outputnya ku lowercase-kan ya, sama di ungkep kurung }

```
for MatMuncul C,i : = { berarti huruf C tidak mu  
MatMuncul C,i : { berarti huruf C muncul }  
  output (C)  
  Count ← 1; output(i)  
  if (i = JumlahKar) then  
    { outputkan yg muncul ,  
      traversal menghitung dimana saja muncul }  
    j traversal [i +1..100]  
    if MatMuncul C,j then  
      Count ← Count + 1; output(j)  
    output ("Muncul ", Count, " kali")
```

program utama

