function size ( S : Stack ) 🡪 Index

{ mengembalikan jumlah elemen stack S. }

Dictionary

-

Algorithm

return S.top

endfunction

function peek ( S : Stack ) 🡪 Infotype

{ mengembalikan elemen teratas stack S, atau karakter X bila stack kosong. }

Dictionary

P : Infotype

function isEmpty( Stack ) 🡪 Boolean

Algorithm

P = 'X'

if not isEmpty( S ) then

P = S.info[S.top]

endif

return P

procedure pop ( in/out S : Stack, in/out P : Infotype)

{ I.S. Terdefinisi stack S yang mungkin kosong dan sebuah infotype P.

F.S. Bila S tidak kosong, P menyimpan elemen teratas yang dihapus

dari S dan jumlah elemen stack berkurang satu. }

Dictionary

function isEmpty( Stack ) 🡪 Boolean

Algorithm

if not isEmpty( S ) then

P = S.info[S.top]

S.top = S.top - 1

endif

endprocedure

procedure push ( in/out S : Stack, in P : Infotype)

{ I.S. Terdefinisi stack S yang mungkin penuh dan sebuah infotype P.

F.S. P menjadi elemen teratas S jika stack S belum penuh. }

Dictionary

function isFull( Stack ) 🡪 Boolean

Algorithm

if not isFull( S ) then

S.top = S.top + 1

S.info[S.top] = P

endif

endprocedure

function isFull ( S : Stack) 🡪 Boolean

{ mengembalikan True jika stack S penuh, atau False jka tidak. }

Dictionary

-

Algorithm

return S.top == MAXSIZE

endfunction

function isEmpty ( S : Stack) 🡪 Boolean

{ mengembalikan True jika stack S kosong, atau False, jika tidak. }

Dictionary

-

Algorithm

return S.top == 0

endfunction

function createStack () 🡪 Stack

{ mengembalikan objek stack dengan top terinisialisasi nol. }

Dictionary

S : Stack

Algorithm

S.top = 0

return S

endfunction

procedure swap( in/out S : Stack )

Kamus

R : Stack

P : Infotype

Algoritma

R = S

S = createStack()

while not (isEmpty(R)) do

pop(R,P)

push(S,P)

endwhile

endprocedure

procedure popth(in/out S: Stack, in Idx: Index, in/out P : Infotype)

Kamus

Algoritma

while S.top != idx do

pop(S,P)

endwhile

pop(S,P)

endprocedure

procedure printStack( in S : Stack )

Kamus

P : Infotype

Algoritma

while not isEmpty(S) do

pop(S,P)

output(P)

endwhile

endprocedure

function isPalindrome( in S : Stack )  Boolean

Kamus

R : Stack

P,Q : Infotype

Algoritma

R = S

swap(S)

while not isEmpty(S) do

if pop(S,P) != pop (R,Q) then

return false

endif

endwhile

return true

endfunction

Procedure pushSorted( in/out S : Stack, in P : Infotype )

Kamus

R : Stack

Q : Infortype

found : boolean

Algoritma

found = true

R = createStack()

while not isEmpty(S) and found do

if S.info[S.top] > P then // bottom ke top descending

pop(S,Q)

push(R,Q)

else

found = false

endif

endwhile

push(S,P)

while not isEmpty(R) do

pop(R,Q)

push(S,Q)

endwhile

endprocedure

A + B \* ( C - D ^ E / F + G \* H ) - I

A + B \* ( C - **^ D E** / F + G \* H ) - I

A + B \* ( C - **/** **^ D E F** + G \* H ) – I

A + B \* ( C - / ^ D E F + \* G H ) – I

A + B \* ( - C / ^ D E F + \* G H ) – I

A + B \* (+ - C / ^ D E F \* G H ) – I

A + \* B (+ - C / ^ D E F \* G H ) – I

+ A \* B (+ - C / ^ D E F \* G H ) – I

- + A \* B + - C / ^ D E F \* G H I