

ADATSZERKEZET (ASZ, DS) (PL. tömb, láncolt lista)

- ADATOK TÁROLÁSÁRA, ELRENDEZÉSÉRE

- Hozzáférés, módosítás elemi műveletei (PL. tömb:  $A[i]$ ,  
 $A[i]:=...$ )

ADATTÍPUS = ASZ + MŰVELETEK

ABSTRACT DATA TYPE (ADT)

matematikai struktúra + műveletei

† (PL. halmaz, sorozat, zsinór)

vagy szemliletes kép

KONKRÉT ADATTÍPUS (OSZTÁLY)

adathasok + metódusok

## Stack (verem)

- $A : \mathcal{T}[]$  //  $\mathcal{T}$  is some known type ;  $A.length$  is the physical constant
  - $m_0 : \mathbb{N}_+ := 16$  // size of the stack, its default is  $m_0$ .
  - $n : \mathbb{N}$  //  $n \in 0..A.length$  is the actual size of the stack
- +  $\text{Stack}(m : \mathbb{N}_+ := m_0) \{ A := \text{new } \mathcal{T}[m] ; n := 0 \}$  // create empty stack
- +  $\sim \text{Stack}() \{ \text{delete } A \}$
- +  $\text{push}(x : \mathcal{T})$  // push  $x$  onto the top of the stack
- +  $\text{pop}() : \mathcal{T}$  // remove and return the top element of the stack
- +  $\text{top}() : \mathcal{T}$  // return the top element of the stack
- +  $\text{isEmpty}() : \mathbb{B} \{ \text{return } n = 0 \}$
- +  $\text{setEmpty}() \{ n := 0 \}$  // reinitialize the stack

LIFO törn  
(last in  
first out)

reverse()

$v : \text{Stack}$

$\text{read}(x)$

$v.\text{push}(x)$

$\neg v.\text{isEmpty}()$

$\text{write}(v.\text{pop}())$

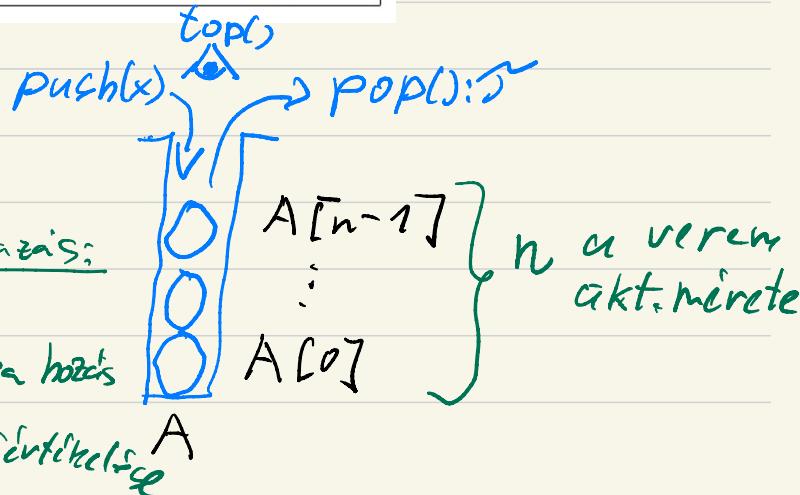
VERMEK:

Néhány alkalmazás:

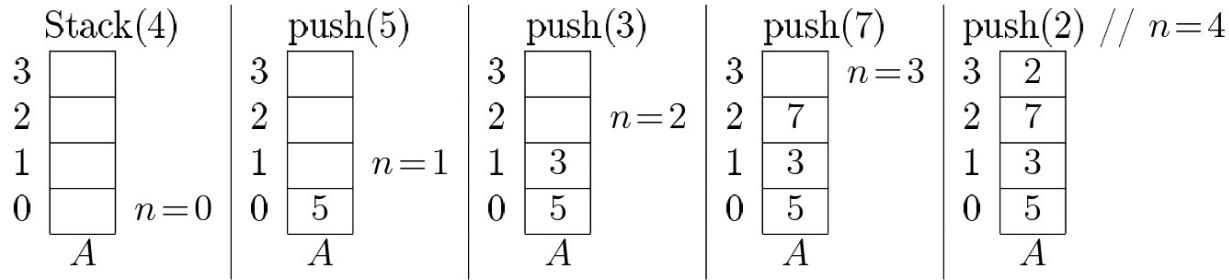
call stack

postfix formára hozás

postfix formák kiértékelése



### Néhány veremművelet



Stack::push( $x : \mathcal{T}$ )

doubleFullArray(& $A : \mathcal{T}[]$ )

$n = A.length$

doubleFullArray( $A$ )	SKIP
$A[n] := x$	
$n++$	

$B : \mathcal{T}[] := \text{new } \mathcal{T}[2 * A.length]$

$i := 0$  to  $A.length - 1$

$B[i] := A[i]$

**delete**  $A$  ;    $A := B$

Stack::pop(): $\mathcal{T}$

$n > 0$

$n--$	
<b>return</b> $A[n]$	StackUnderflow

Stack::top(): $\mathcal{T}$

$n > 0$

<b>return</b> $A[n - 1]$	StackUnderflow
--------------------------	----------------

$MT_{\text{push}}(n) \in \Theta(1)$

$MT_{\text{push}}(n) \in \Theta(n)$

$AT_{\text{push}}(n) \in O(1)$

(Amortizált)

műveletek száma ( $t_n$ )

$MT_{\text{op}}(n) \in \Theta(1)$   
( $\text{op} \neq \text{push}$ )

# FIFO tar(first in, first out)

## Queue (Sor)

-  $Z : \mathcal{T}[]$  //  $\mathcal{T}$  is some known type ;  $Z.length$  is the physical  
constant  $m0 : \mathbb{N}_+ := 16$  // length of the queue, its default is  $m0$ .  
 $n : \mathbb{N}$  //  $n \in 0..Z.length$  is the actual length of the queue  
 $k : \mathbb{N}$  //  $k \in 0..(Z.length - 1)$  : the starting position of the queue in  $Z$

+ Queue( $m : \mathbb{N}_+ := m0$ ) {  $Z := \text{new } \mathcal{T}[m]$  ;  $n := 0$  ;  $k := 0$  }  
// create an empty queue

+ add( $x : \mathcal{T}$ ) // join  $x$  to the end of the queue

+ rem() :  $\mathcal{T}$  // remove and return the first element of the queue

+ first() :  $\mathcal{T}$  // return the first element of the queue

+ length() :  $\mathbb{N}$  { **return**  $n$  }

+ isEmpty() :  $\mathbb{B}$  { **return**  $n = 0$  }

+ ~ Queue() { **delete**  $Z$  }

+ setEmpty() {  $n := 0$  } // reinitialize the queue

$\langle Z[k], Z[(k+1) \bmod m], \dots, Z[(k+n-1) \bmod m] \rangle$  } az ábrazolt  
 $(n > 0)$   
 $n = 0 : \langle \rangle$

$(m = Z.length)$

## Egy sor néhány művelete

Queue(4)

0 1 2 3

$k$

$n = 0$

rem() : 3

0 1 2 3

$k$

$n = 0$

add(5)

0 1 2 3

$k$

$n = 1$

add(7)

0 1 2 3

$k$

$n = 1$

add(3)

0 1 2 3

$k$

$n = 2$

add(2)

0 1 2 3

$k$

$n = 2$

rem() : 5

0 1 2 3

$k$

$n = 1$

add(4)

0 1 2 3

$k$

$n = 3$

$mT_{add}(n) \in \Theta(n)$

$M\bar{T}_{add}(n) \in \Theta(n)$

$A\bar{T}_{add}(n) \in \Theta(1)$

$M\bar{T}_{op}(n) \in \Theta(1)$   
 $(op \neq add)$

Queue::add( $x : \mathcal{T}$ )

$n = Z.length$

doubleFullQueueArray( $Z, k$ )

SKIP

$Z[(k + n) \bmod Z.length] := x$

$n++$

$A\bar{T}_{op}(n) \in \Theta(1)$

$M\bar{T}_{op}(n) \in \Theta(1)$   
 (tetszőleges op.)

Queue::rem() :  $\mathcal{T}$

$n > 0$

$n --$

$i := k$

$k := (k + 1) \bmod Z.length$

**return**  $Z[i]$

QueueUnderflow

Queue::first() :  $\mathcal{T}$

$n > 0$

**return**  $Z[k]$

QueueUnderflow

SOROK:

Néhány alkalmazás:

- IO pufferek
- üggyfelszolgálati programok (üzletek, bankok stb.)
- Folyamat ütemezés
- Fák szintenkörű bejárása (ebbőn a fóliánban)
- Szélességi keresés graffiton (hálózatokon)
- Legrövidebb út keresése a legáltalánosabb esetben

...

