Tehnici de programare

Stiva

ovidiu.banias@upt.ro

Cuprins

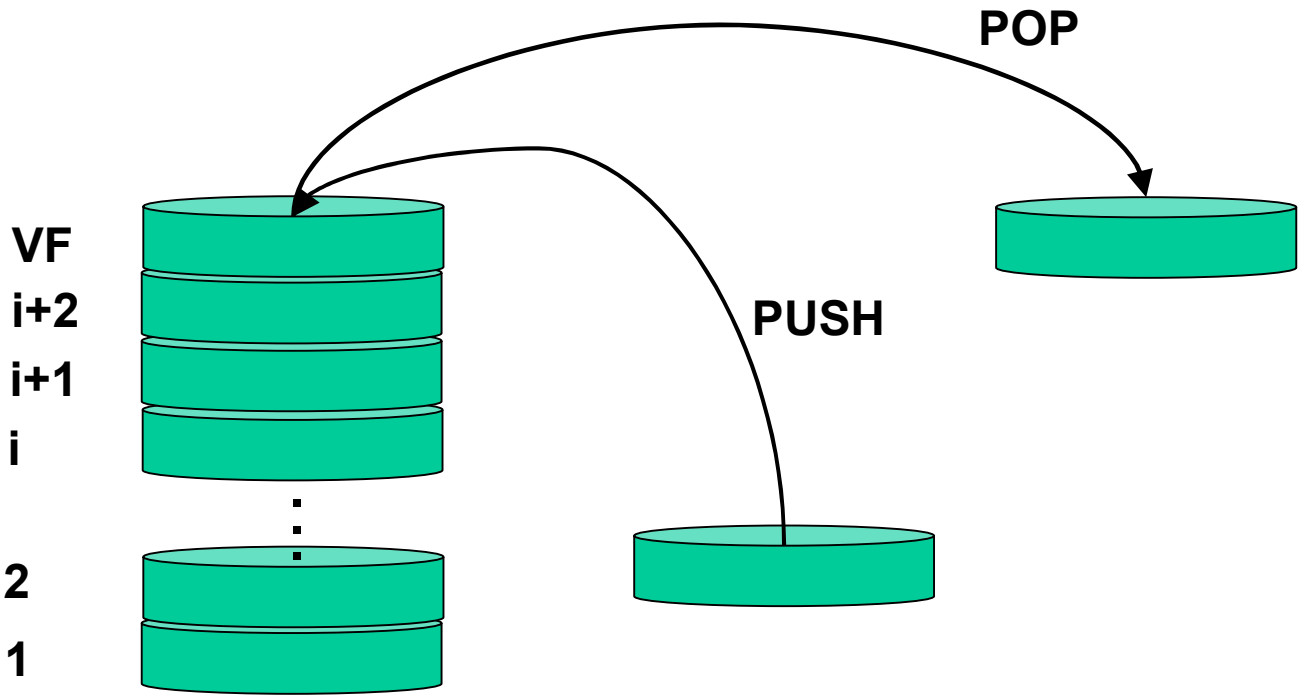
Stiva

Forma poloneză

Calculul unei expresii matematice

Stiva ca structură de date

Definiţie: structură de date abstractă, având prorietatea că operațiile de adăugare şi extragere se realizează numai din vârful stivei.
(LIFO – Last In First Out)



Stiva. Aplicabilitate

- Se folosesc vectori pt. simularea stivei
- Stă la baza recursivității

Funcția Manna Pnueli

$$f(x)=\begin{cases} x-1, \text{dacă } x \geq 12 \\ f(f(x+2)), \text{altfel} \end{cases}$$

Funcția Ackermann

$$f(x,y)=\begin{cases} x+1 & \text{dacă x} = 0 \\ ac(x-1,1) & \text{dacă } y=0 \\ ac(x-1,ac(x,y-1)) & \text{altfel} \end{cases}$$

Stiva. Manna Pnueli

Funcţia Manna Pnueli

$$f(x)=\begin{cases} x-1, \text{dacă } x \geq 12 \\ f(f(x+2)), \text{altfel} \end{cases}$$

$f(12)=11, f(13)=12, f(14)=13,...$

$$f(6)=f(f(8))=f(f(f(10)))=f(f(f(f(12))))=f(f(f(11)))=f(f(f(f(13))))=$$
$$=f(f(f(12)))=f(f(11))=f(f(f(13)))=f(f(12))=f(11)=f(f(13))=f(12)=11$$

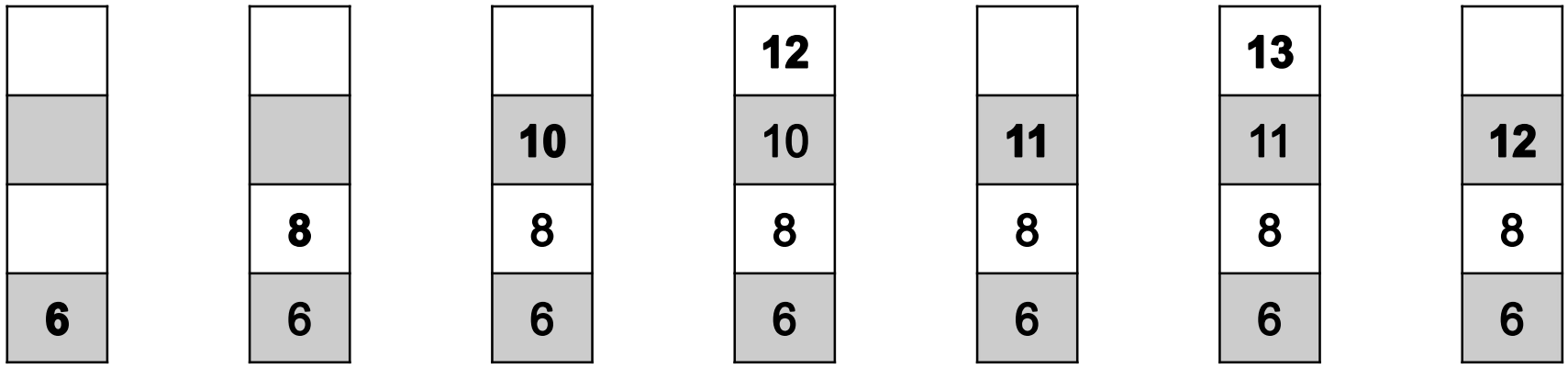
Implementare in C?

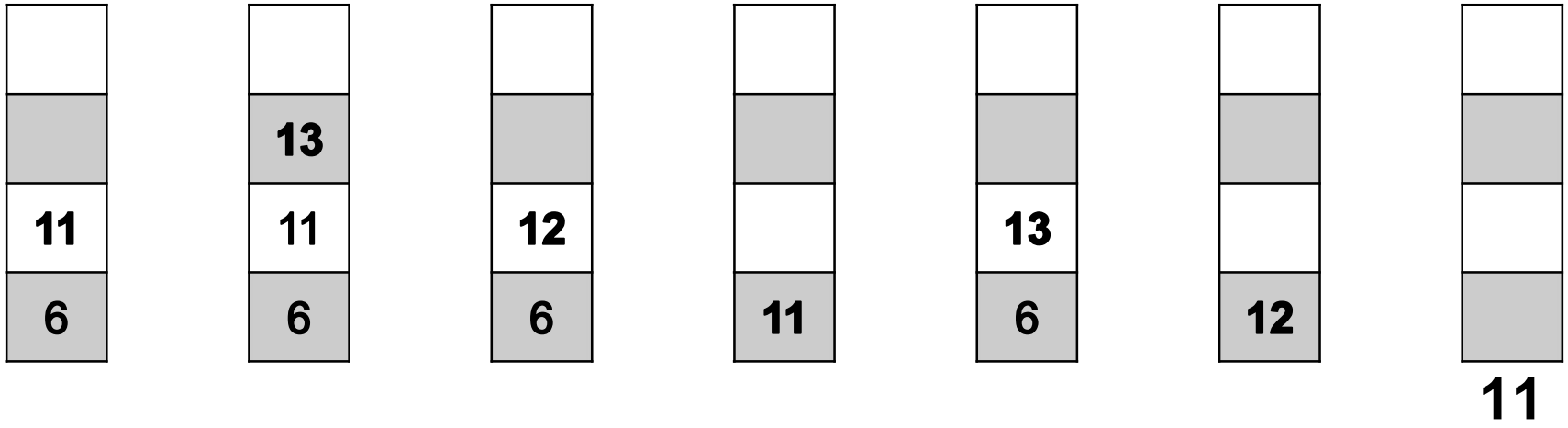
- se foloseşte o structură de date de tip stiva st
- se adaugă în stivă valoarea lui x
- pentru x<12, la apelul funției f se pune(PUSH) în stivă rezultatul x+2
- pentru x≥12, se scoate din stivă (POP) şi se modifică nou vârf al stivei cu x-1
- algoritmul se încheie când nu mai sunt elemente în stivă

Stiva. Manna Pnueli (cont)

$f(6) = ?$

$$f(x) = \begin{cases} x - 1, \text{dacă } x \geq 12 \\ f(f(x+2)), \text{altfel} \end{cases}$$





Stiva. Ackermann

Funcţia Ackermann

$$f(x,y)=\begin{cases} y+1 & \text{daca } x=0 \\ f(x-1,1) & \text{daca } y=0 \\ f(x-1,f(x,y-1)) & \text{altfel} \end{cases}$$

$$f(2,1)=f(1,f(2,0))=f(1,f(1,1))=f(1,f(0,f(1,0)))=$$
$$=f(1,f(0,f(0,1)))=f(1,f(0,2))=f(1,3)=f(0,f(1,2))=$$
$$=f(0,f(0,f(1,1)))=f(0,f(0,f(0,f(1,0))))=$$
$$f(0,f(0,f(0,f(0,1))))=f(0,f(0,f(0,2)))=f(0,f(0,3))=f(0,4)=5$$

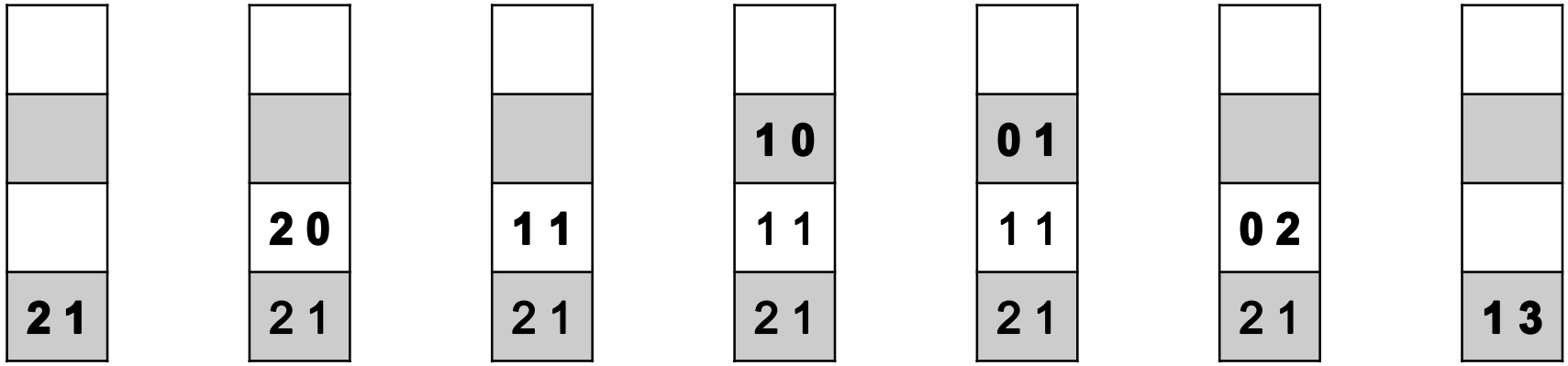
Implementare in C?

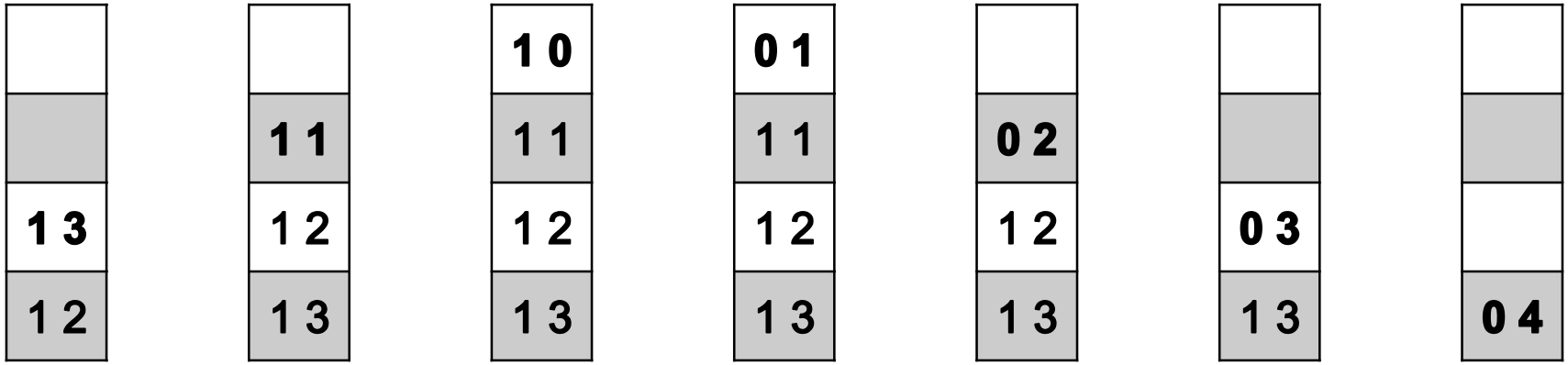
- se foloseşte o structură de date de tip stiva cu 2 elemente st
- se adaugă în stivă valoarile x şi y
- Pentru x şi y ≠0, la apelul funției f se pune în stivă (x,y-1)
- pentru y=0, se modifică vârful stivei cu valorile (x-1,1)
- pentru x=0, se scoate din stivă (x',y') şi se modifică nou vârf al stivei cu (x-1,y')

Stiva. Ackermann (cont)

$f(2,1) = ?$

$$f(x,y) = \begin{cases} y+1 & \text{daca } x = 0 \\ f(x-1,1) & \text{daca } y = 0 \\ f(x-1, f(x,y-1)) & \text{altfel} \end{cases}$$





Forma poloneză postfixată

Definiţie: FP postfixată este o notație matematică prin care orice operator urmează operanzii săi.

Fie E_1, E_2 expr. aritmetice şi $@$ un operator $= \{*, /, +, -\}$.

$$E_1 @ E_2 \overset{FP}{\rightarrow} E_1 E_2 @$$

$a+b$	forma normală	$(a+b)*(c-d)$	forma normală
ab +	forma postfixată	ab + cd - *	forma postfixată
+ ab	forma prefixată	* +ab - cd	forma prefixată

$a*(b-c)+d/(e+f*h)-i$ forma normală

*abc-*defh*+ /i- +* forma postfixată

Forma poloneză postfixată. Exemplificare

Problemă: Se dă o expresie aritmetică în formă normală. Să se afişeze expresia în formă poloneză postfixată.

Observații:

- se ține cont de ordinea efectuării operațiilor, vor fi setate priorități
- pentru operatorii care nu pot fi folosiți la un moment dat (datorită ordinii efectuării operațiilor), se va folosi stiva

---

Input $(a*(b-c)+d/(e+f*h)-i)$

Output $abc-*defh*+/i-+$

Forma poloneză postfixată. Rezolvare

1. Se definesc prioritățile operatorilor

   ‘(‘,’)’ – prioritate 0

   “*‘,’/’ – prioritate 1

   ‘+‘,’-’ – prioritate 2

2. Expresia matematică se citeşte caracter cu caracter şi este de forma **(E)**

3. Operanzii se introduc în vectorul **fp**

4. Operatorii se introduc in stiva **st**, apoi se transferă în **fp** cu excepția ‘(‘,’)’

5. În funcție de valoarea şi prioritatea operatorului din vârful stivei se fac următoarle operații:

- dacă (prioritate(op)==1), nu se face nici o operație suplimentară
- dacă (prioritate(op)==2), se scoate temporar operatorul op din vârful stivei, se transferă din stivă în fp toți operatorii cu prioritate 1, apoi se reintroduce in stivă operatorul op
- dacă (op==‘)’), se scot din stivă toți operatorii până când (op==‘(’) şi se adaugă in fp. Cu siguranță în acest moment, operatorii dintre paranteze vor avea aceeași prioritate!

Forma poloneză postfixată. Exemplu

Exemplu : $(a*(b-c)+d/(e+f*h)-i)$

st	(
fp	

st	(
fp	a

st	(*
fp	a

st	(*(
fp	a

st	(*(
fp	ab

st	(*(-
fp	ab

st	(*(-
fp	abc

st	(*(-)
fp	abc

st	(*
fp	abc-

st	(*+
fp	abc-

st	(+
fp	abc-*

st	(+
fp	abc-*d

st	(+/
fp	abc-*d

st	(+/(
fp	abc-*d

st	(+/(
fp	abc-*de

st	(+/(+
fp	abc-*de

st	(+/(+
fp	abc-*def

st	(+/(+*
fp	abc-*def

st	(+/(+*
fp	abc-*defh

st	(+/(+*)
fp	abc-*defh

st	(+/
fp	abc-*defh*+

st	(+/-
fp	abc-*defh*+

st	(+-
fp	abc-*defh*+/

st	(+-)
fp	abc-*defh*+i

st	
fp	abc-*defh*+/i-+

Forma poloneză postfixată. Exemplu

Exemplu : (8/4*2)

st	(
fp	

st	(
fp	8

st	(/
fp	84

st	(
fp	84/

st	(*
fp	84/

st	(*)
fp	84/2

st	()
fp	84/2*

st	(
fp	

st	(
fp	8

st	(/
fp	84

st	(/*
fp	84

st	(/*)
fp	842

st	()
fp	842*/

incorect

Calculul unei expresii aritmetice

Problemă: Se dă o expresie aritmetică în formă normală şi valorile fiecărui operand al expresiei. Să se calculeze rezultatul expresiei aritmetice.

Observaţie:

- Pornind de la forma poloneză postfixată se pot calcula uşor subexpresii de forma:

$$r_i = op_i op_{i+1} @, \text{ unde } @ = \{*, /, +, -\},\ i = \overline{1, n-1}$$

Input

$(a * (b + c))$
2
3
1

Output 8

cu semnificaţia a=2, b=3, c=1

Calculul unei expresii aritmetice. Rezolvare

1. Se transformă expresia aritmetică din formă normală în formă poloneză postfixată

2. Se parcurge **fp** de la stânga la dreapta

3. Dacă **fp[k]** este operand atunci se introduce în stiva **st**

4. Dacă **fp[k]** este operator atunci se scot din stiva **st** ultimii doi operanzi şi se introduce în stivă rezultatul expresiei

$$op_i @ op_{i+1}, \text{unde } @ - \text{operator}; op_i, op_{i+1} - \text{operanzi}$$

Calculul unei expresii aritmetice. Exemplu

Exemplu: $(a * (b - c) + d/(e + f * g) - h)$

2
3
3
30
6
4
1
6

$a = 2 \quad b = 3 \quad c = 3$
$d = 30 \quad e = 6 \quad f = 4$
$g = 1 \quad h = 6$

$$(a * (b - c) + d/(e + f * g) - h) \rightarrow abc - * defg * + /h - +$$

st	abc
op	-
r1	b-c=0

st	a0
op	*
r2	a*0=0

st	0defg
op	*
r3	4*1=4

st	0de4
op	+
r4	6+4=10

st	0d10
op	/
r5	30/10=3

st	03h
op	-
r6	3-6=-3

st	0-3
op	+
r7	0+(-3)=-3

$= -3$