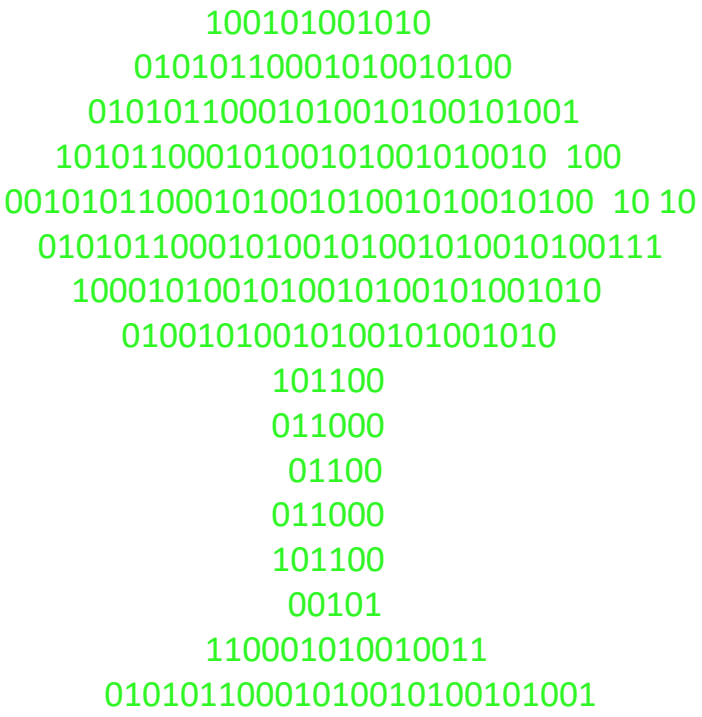
Structuri de date și algoritmi



Curs, IS – An II

Stive

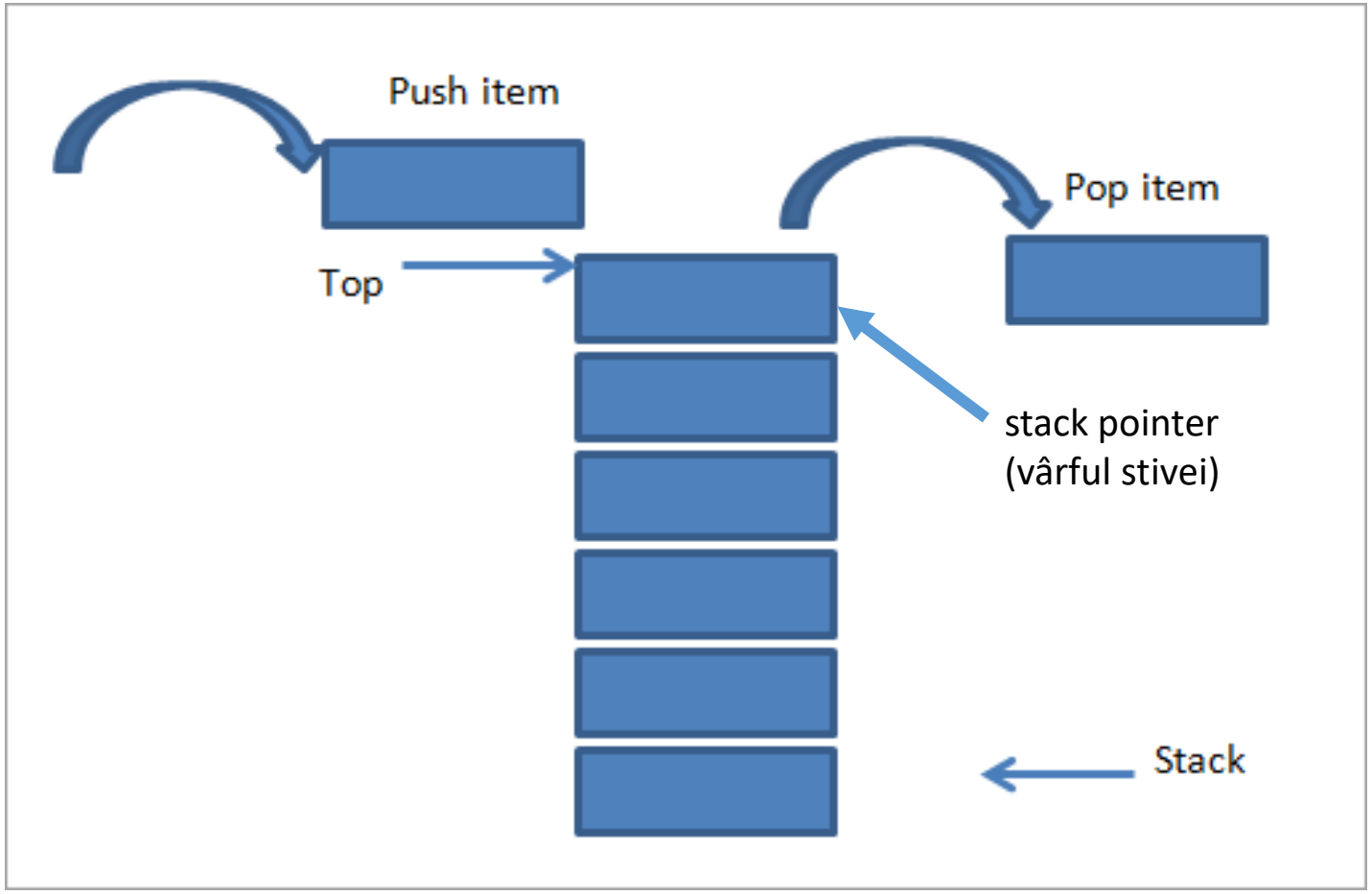
STIVE

- o listă specializată organizată pe principiul **LIFO** (Last In First Out) / **FILO** (First In Last Out):
  - Inserarea și ștergerea se realizează numai la un (același) capăt al listei

Operații de bază pe TAD Stivă (Stack):

- Push: adăugare element
- Pop: extragere element
- Top: returnarea valorii primului element
- Init: inițializează stiva (stiva vidă)
- IsEmpty: întoarce 1 (true) dacă stiva este goală și 0 (false) dacă are elemente

STIVE



STIVE

Utilizare:

- Inversarea unei secvențe (cuvânt, numere, etc.)
- Verificare dacă un cuvânt este palindrom
- Mecanismul de “undo” (modificările sunt păstrate într-o stivă)
- În algoritmi de tip backtracking (explorarea spațiului soluțiilor):
  - ex. Găsirea drumului într-un labirint – variantele de explorare se stochează într-o stivă
- Evaluarea expresiilor aritmetice
  - utilizând forma pre-fixată/post-fixată

STIVE

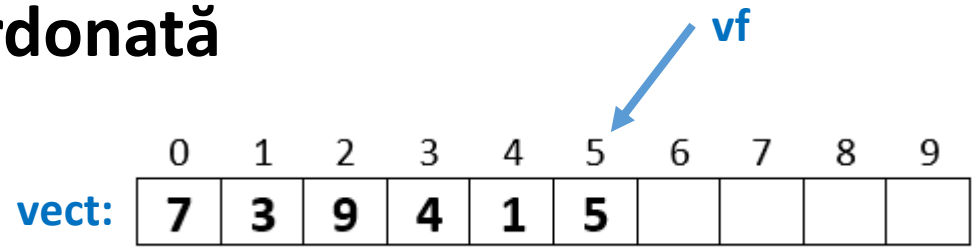
Reprezentarea în memorie:

A. Stiva ordonată (alocată static)

B. Stiva dinamică (alocată dinamic)

STIVE

A. Stiva ordonată

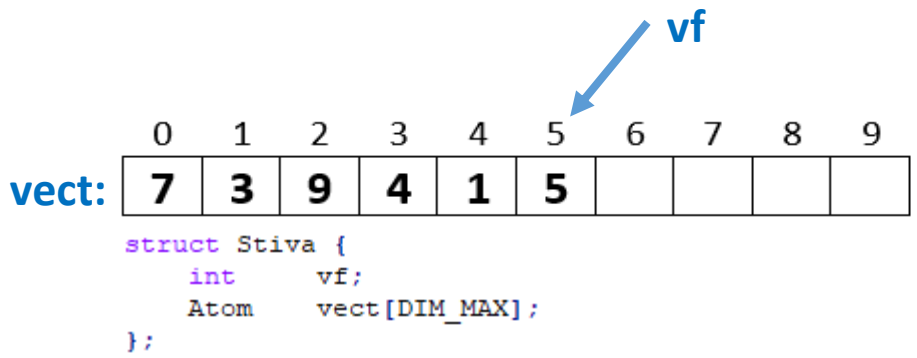



```
#define DIM_MAX 10

typedef ... Atom                    //ex: typedef int Atom

struct Stiva {
    int     vf;                     //stack pointer
    Atom    vect[DIM_MAX];          //tabloul ce contine elementele stivei
};
```

STIVE



A. Stiva ordonată

Operații în stiva ordonată

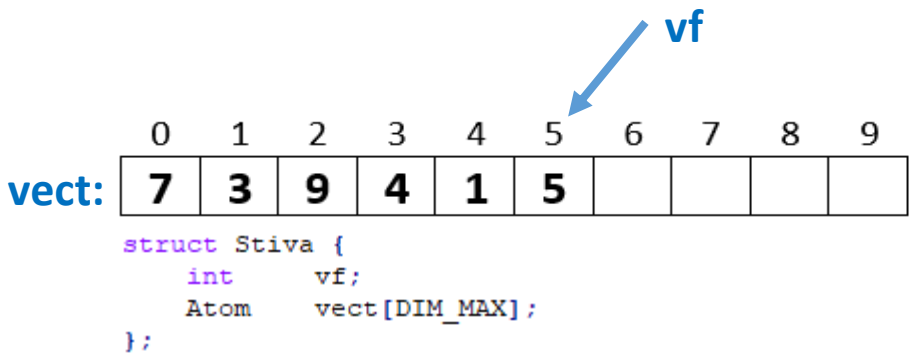
```
InitStack(s)
        s.vf := -1
End
```

```
IsFull(s)
        if(s.vf = DIM_MAX-1)
                return true
        end-if
        return false
End
```

```
IsEmpty(s)
        if(s.vf = -1)
                return true
        end-if
        return false
End
```

```
Top(s)
        return s.vect[s.vf]
End
```

STIVE



A. Stiva ordonată

Operații în stiva ordonată

```
Push(s, val)
        s.vf ++
        s.vect[s.vf] := val
End
```

```
Pop(s)
        s.vf--
End
```

Atenție la mecanismul de tratare a erorilor!

V1: lăsat în seama utilizatorului stivei (ca mai sus)

V2: integrat în implementarea operațiilor Top/Pop/Push

- returnare valoare de success la realizarea operației
- mesaj de eroare

STIVE

```
struct Stiva {
    int      vf;
    Atom     vect[DIM_MAX];
};
```

A. Stiva ordonată

Prototipurile funcțiilor in C++

```
void InitStack(Stiva &s);
bool isEmpty(const Stiva &s);
Atom Top(const Stiva &s);
void Push(Stiva &s, Atom val);
void Pop(Stiva &s);
```

Transmitere prin referință pentru a evita copierea parametrului (`s.vect` poate fi mare)!!!

```
InitStack(s)
        s.vf := -1
End
```

```
IsEmpty(s)
        if(s.vf = -1)
                return true
        end-if
        return false
End
```

```
IsFull(s)
        if(s.vf = DIM_MAX-1)
                return true
        end-if
        return false
End
```

```
Top(s)
        return s.vect[s.vf]
End
```

```
Push(s, val)
        s.vf ++
        s.vect[s.vf] := val
End
```

```
Pop(s)
        s.vf--
End
```

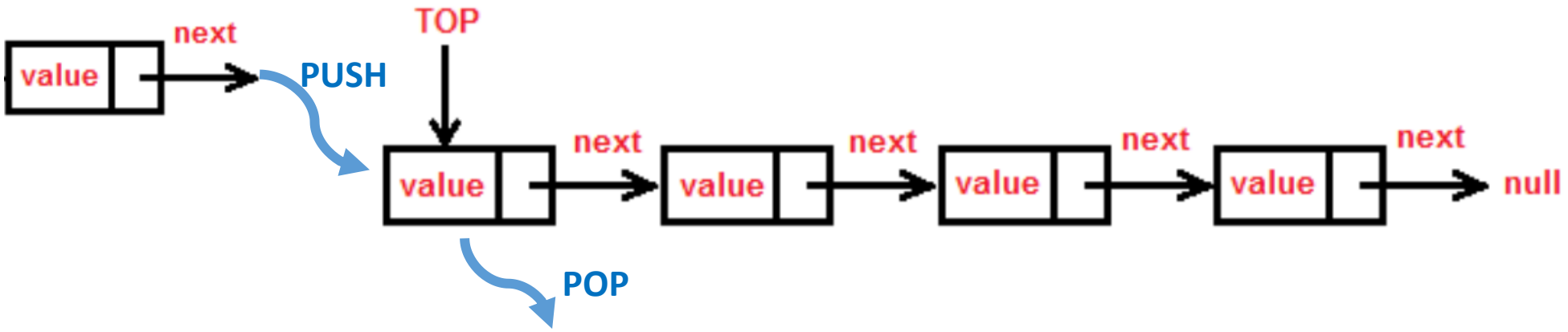
STIVE

```
struct Element {
    Atom    value;
    Element *next;
};

typedef Element* Stiva;
```

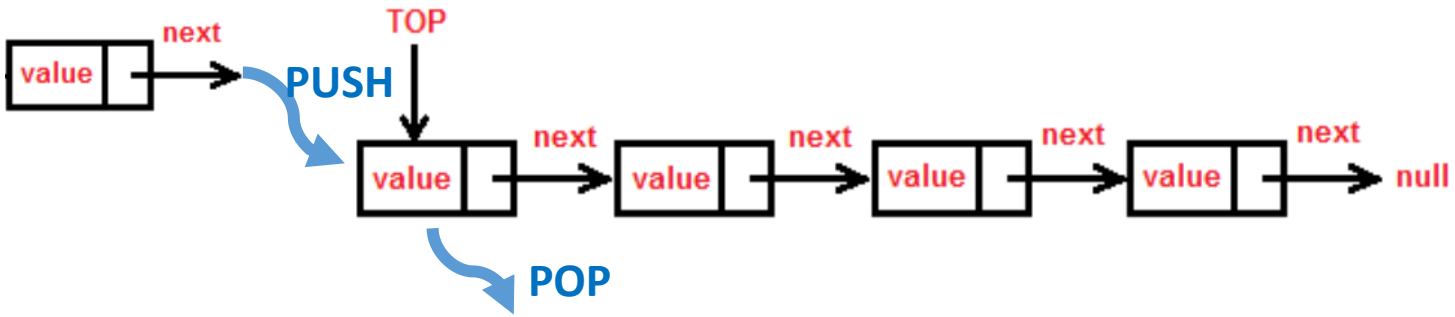
A. Stiva dinamică

- listă liniară simplu înlănțuită



- PUSH – inserare în fața listei
- POP – ștergerea primului element
- TOP – consultarea primului element

STIVE



A. Stiva dinamică

Operații în stiva dinamică

```
InitStack(s)
        s := 0
End
```

```
IsEmpty(s)
        if(s = 0)
                return true
        end-if
        return false
End
```

```
Top(s)
        return value(s)
End
```

```
Push(s, val)
        p := create_elem(val)
        value(p) := val
        next(p) := s
        s := p
End
```

```
Pop(s)
        p := s
        s := next(s)
        delete(p)
End
```

STIVE

A. Stiva dinamică

Operații în stiva dinamică

Atenție la mecanismul de tratare a erorilor!
V1: lăsat în seama utilizatorului stivei (ca în dreapta)

```
void InitStack(Stiva &s);
bool isEmpty(Stiva s);
Atom Top(Stiva s);
void Push(Stiva &s, Atom val);
void Pop(Stiva &s);
```

V2: integrat în implementarea operațiilor Top/Pop/Push
- returnare valoare de success la realizarea operației

```
bool Top(Stiva &s, Atom &val);
bool Push(Stiva &s, Atom val);
bool Pop(Stiva &s);
```

```
InitStack(s)
        s := 0
End
```

```
IsEmpty(s)
        if(s = 0)
                return true
        end-if
        return false
End
```

```
Top(s)
        return value(s)
End
```

```
Push(s, val)
        p := create_elem(val)
        value(p) := val
        next(p) := s
        s := p
End
```

```
Pop(s)
        p := s
        s := next(s)
        delete(p)
End
```

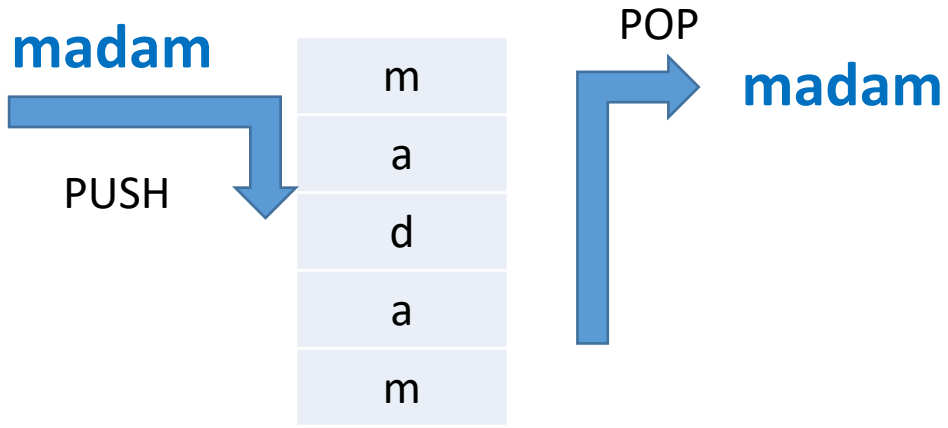
STIVE - utilizare

Biologie moleculară: Many molecular lengths between 4 and 8 nucleotides are palindromic as they correspond to nitrogenous sequences that read the same forwards as they do backward.

Verificare palindrom

abcdeedcba **123454321**

Secvență care citită invers este identică cu cea originală



STIVE - utilizare

Evaluarea expresiilor aritmetice

Ideea:

- rearanjarea expresiei a.î. să nu conțină paranteze,
- ordinea în care se efectuează operațiile să fie clară și evaluarea ușor de făcut pe calc.

Forma prefixată	**Forma infixată**	**Forma postfixată**
+ab	a+b	ab+
forma poloneză		*forma poloneză inversă*

STIVE - utilizare

Forma postfixată

ab+

Evaluarea expresiilor aritmetice

Forma postfixată:

- Operatorii apar în ordinea în care se execută operațiile la evaluarea expresiei
- Operatorii apar în urma operanzilor
- Evaluarea se face parcurgând expresia stg -> drt și executând operațiile (ținând cont de precedența lor)

Definiție

- Pentru orice operand (constantă sau variabilă) *E*, *E* este forma poloneză a operandului *E*
- Dacă *E* este o expresie de forma *E1 op E2* , f.p. a expresiei este *E1' E2' op* unde *E1'* si *E2'* sunt respectiv f.p. ale expresiilor *E1* si *E2*
- Dacă *E* este o expresie de forma (*E1*). f.p. a expresiei *E1* este de asemenea f.p. a expresiei *E*

STIVE - utilizare

Forma postfixată
ab+

Evaluarea expresiilor aritmetice

Forma postfixată:

Expresie	Forma postfixată
E (operand sau operator)	E’ = E
E1 *op* E2	E1’ E2’ *op*
(E1)	E1’

Expresie	Formă postfixată
a+b	ab+
4+5*5	455*+
4*2+3	42*3+
4*(2+3)	423+*
5+(2*3+4)/(6-4)	

STIVE - utilizare

Forma postfixată

ab+

Evaluarea expresiilor aritmetice

Forma postfixată – evaluare

- Se folosește o **stivă** de operanzi
- Se parcurge expresia
  - Dacă se întâlnește operand -> se adaugă în stivă
  - Daca se întâlnește operator
    - se extrag 2 operanzi din stivă,
    - se efectuează operația,
    - se depune rezultatul în stivă

STIVE - utilizare

Forma postfixată
ab+

Evaluarea expresiilor aritmetice

Forma postfixată:

```
// Expresia se afla intr-un tablou s cu elemente simboluri de
// tip operand sau operator binar
i=0
while (nu s-a terminat sirul s) do
    if (s[i] este operand) then
        depune s[i] in stiva
    else    if (s[i] este operator) then
                extrage din stiva doua simboluri t1 si t2;
                executa operatia s[i] asupra lui t1 si t2;
                depune rezultatul in stiva
            else semnalizeaza eroare
            endif
    endif
    i=i+1
end-while
```

STIVE - utilizare

Forma postfixată

ab+

Evaluarea expresiilor aritmetice

Trecerea din formă **infixată** în forma **postfixată**: a+b → ab+

- Se folosește o **stivă** pentru stocarea temporară a operatorilor
- Se parcurge expresia infixată
  - Daca se întâlnește operand → se adaugă în expresia postfixată
  - Daca se întâlnește operator
    - se scot din stivă toți operanzii cu precedența mai mare sau egală, până la '(' și se adaugă în expresia postfixată (exclusiv '(')
    - se adaugă în stivă operatorul curent
  - Dacă se întâlnește '(' → se adaugă în stivă
  - Dacă se întâlnește ')'
    - se scot din stivă toți operatorii până la '(' și se adaugă la expresia postfixată
    - se scoate din stivă '('

STIVE - utilizare

Forma postfixată
ab+

Evaluarea expresiilor aritmetice

Trecerea din formă **infixată** în formă **postfixată**: a+b → ab+

- Se folosește o **stivă** pentru stocarea temporară a operatorilor

EXEMPLU

5 + (2 * 3 + 4) / (6 - 4)