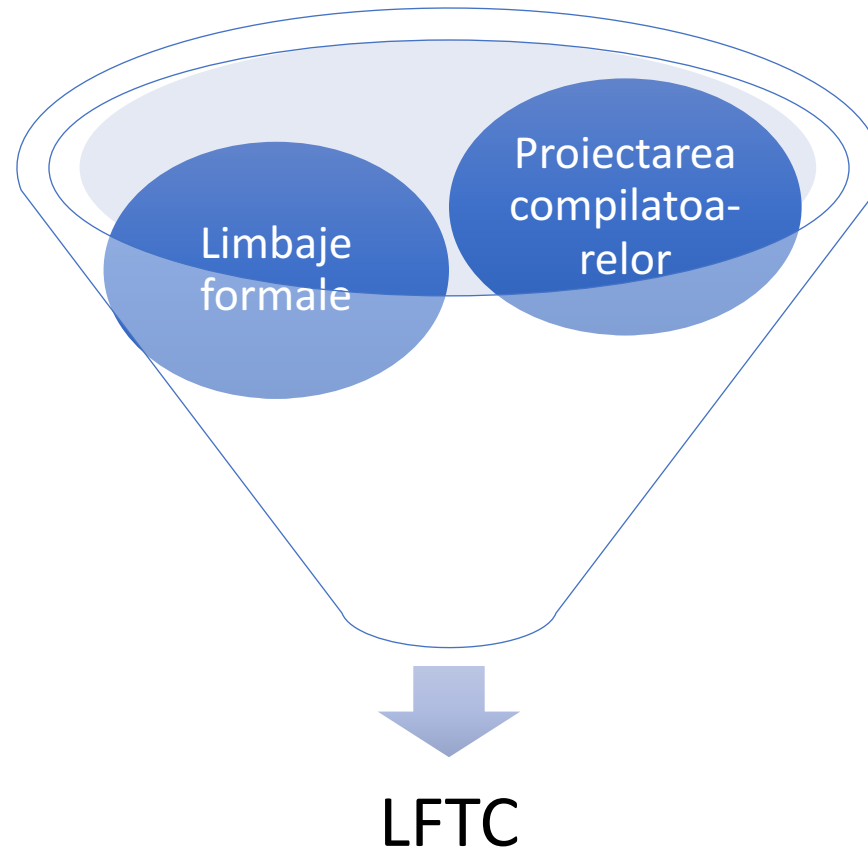


# Limbaje formale și tehnici de compilare

*Símona Motogna*

# De ce facem acest curs?



# Aspecte organizatorice

- **INFORMATICA**

- Curs – 2 h/ sapt
- Seminar – 2h/sapt
- Laborator - 2 h/sapt

10 prezențe – seminar  
12 prezențe - laborator

- **MATEMATICA - INFORMATICA**

- Curs – 2 h/ sapt
- Seminar – 1h/sapt
- Laborator - 1 h/sapt

5 prezențe – seminar  
6 prezențe - laborator

**PREZENȚA OBLIGATORIE**

# Aspecte organizatorice

- Nota finala =
  - 70% examen scris
  - + 20% lab
  - + 10% seminar

## Lab:

- toate laboratoarele predate
- nu se admit întârzieri mai mari de 2 săptămâni

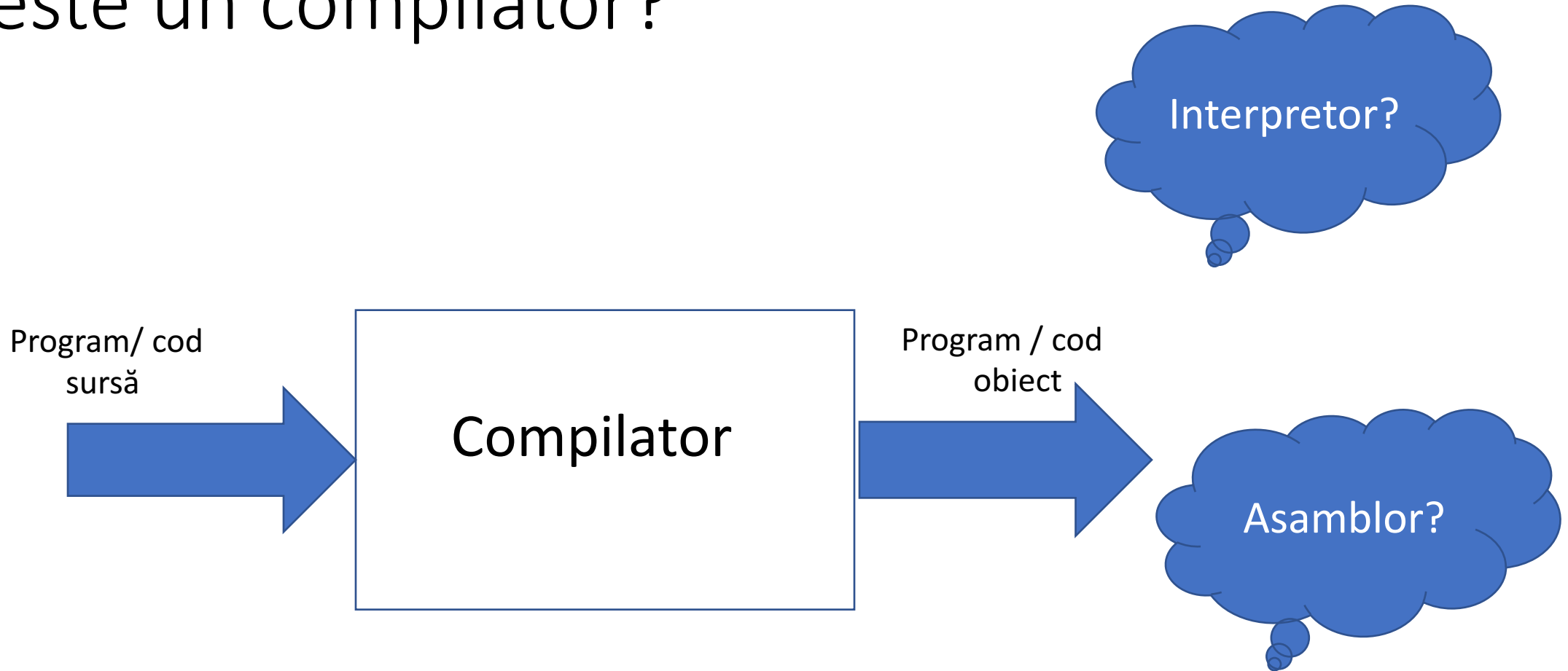
## Seminar:

- probleme rezolvate, ieșit la tablă, teme

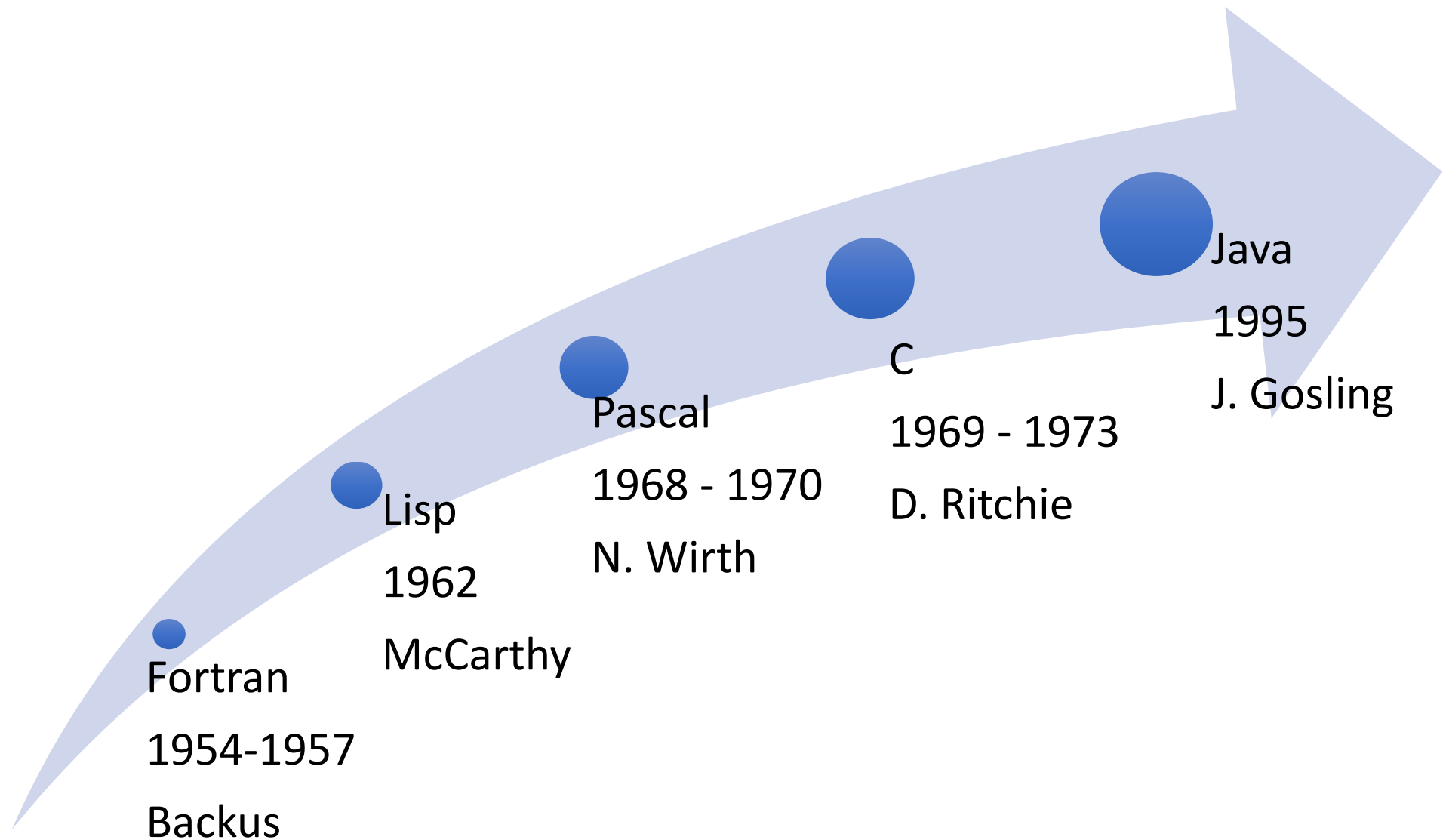
# Bibliografie

- Vezi fișa disciplinei

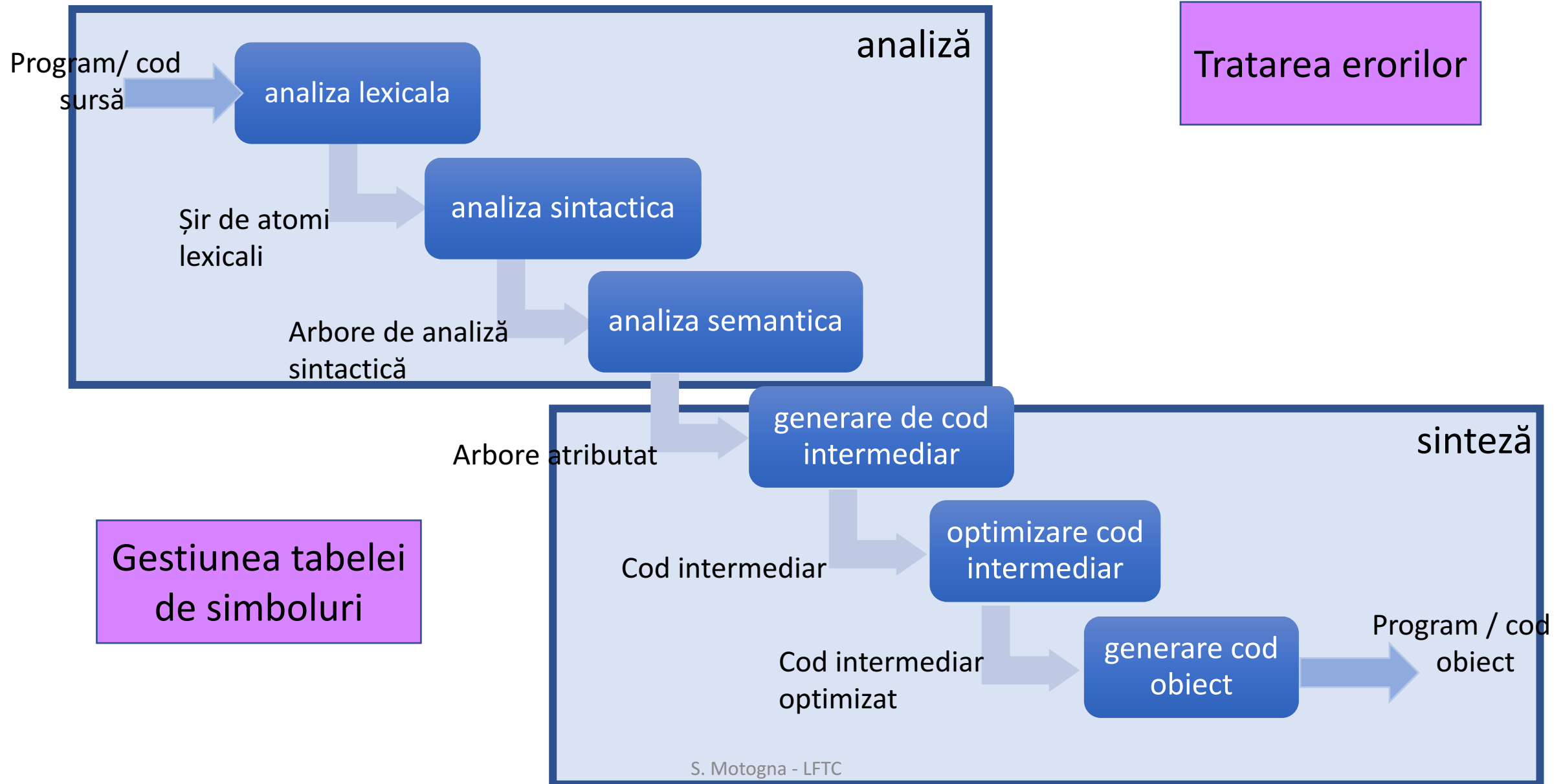
# Ce este un compilator?



# Un pic de istorie ...



# Structura unui compilator





# Cap. 1. Analiză lexicală

**Definiție** = privește programul sursă ca un șir de caractere, detectează atomii lexicali, pe care îi clasifică și codifică

INPUT: program sursă

OUTPUT: FIP + TS

*Algorithm AnalizaLexicala v1*

```
While (not.eof) do  
    detectare(atom);  
    clasificare(atom);  
    codificare(atom);  
End_while
```

# Detectare

Ana are mere.

- Separatori => **Obs 1)**


If  $(x==y)$   $\{x=y\}$

- Look-ahead => **Obs 2)**

# Clasificare

- Clase de atomi lexicali:
  - Identificatori
  - Constante
  - Cuvinte rezervate (cuvinte cheie)
  - Separatori
  - Operatori
- Dacă un atom lexical nu poate fi clasificat => EROARE LEXICALĂ

# Codificare

- Tabel de codificare
- Identificator, constantă => tabelă de simboluri (TS)
- FIP = forma internă a programului = șir de perechi
- Atom lexical – înlocuit cu pereche (cod, poziție in TS)  
  
identificator, constantă

## *Algorithm AnalizăLexicală v2*

```
While (not.eof) do  
    detectare(atom);  
    if atom este cuvânt rezervat sau operator sau separator  
        then genFIP(cod, 0)  
        else  
            if atom este identificador sau constantă  
                then indice = poz(atom, TS);  
                    genFIP(cod, indice)  
                else mesaj "Eroare lexicală"  
            endif  
        endif  
    endif  
endwhile
```

# Observații:

- genFIP = adaugă o pereche (cod, poziție) la FIP
- Poz(atom,TS) – caută *atom* în tabela de simboluri *TS*; dacă îl găsește returnează poziția; dacă nu, inserează *atom* în tabela *TS* și returnează poziția
- Ordinea în clasificare (cuvânt rezervat, apoi identificator)
- If-then-else imbricate => detectare eroare dacă un atom nu poate fi clasificat

# Tabela de simboluri

**Definiție** = conține toate informațiile colectate de fazele compilatorului relative la numele simbolice din programul sursă



identificatori, constante, etc.

Variante:

- tabelă de simboluri unică – conține toate numele simbolice
- tabele de simboluri distincte: TI (tabela identificatorilor) + TC (tabela constantelor)

# Organizare TS

*Observație:* căutare și inserare

- |   |            |
|---|------------|
| 1. Tabel neordonat – în ordinea detectării în codul sursă | $O(n)$     |
| 2. Tabel ordonat: ordine alfabetică (numerică)            | $O(\lg n)$ |
| 3. Arbore binar de căutare (echilibrat)                   | $O(\lg n)$ |
| 4. Tabel de dispersie                                     | $O(1)$     |



# Tabele de dispersie

- $K$  = mulțimea cheilor (nume simbolice)
- $A$  = mulțimea pozițiilor ( $|A| = m$ ;  $m$  – nr prim)

$$h : K \rightarrow A$$

$$h(k) = (\text{val}(k) \bmod m) + 1$$

- Conflicte:  $k_1 \neq k_2$  ,  $h(k_1) = h(k_2)$

# Domeniu de vizibilitate

- Fiecare domeniu de vizibilitate – propria TS
- Structura -> arbore de incluziune