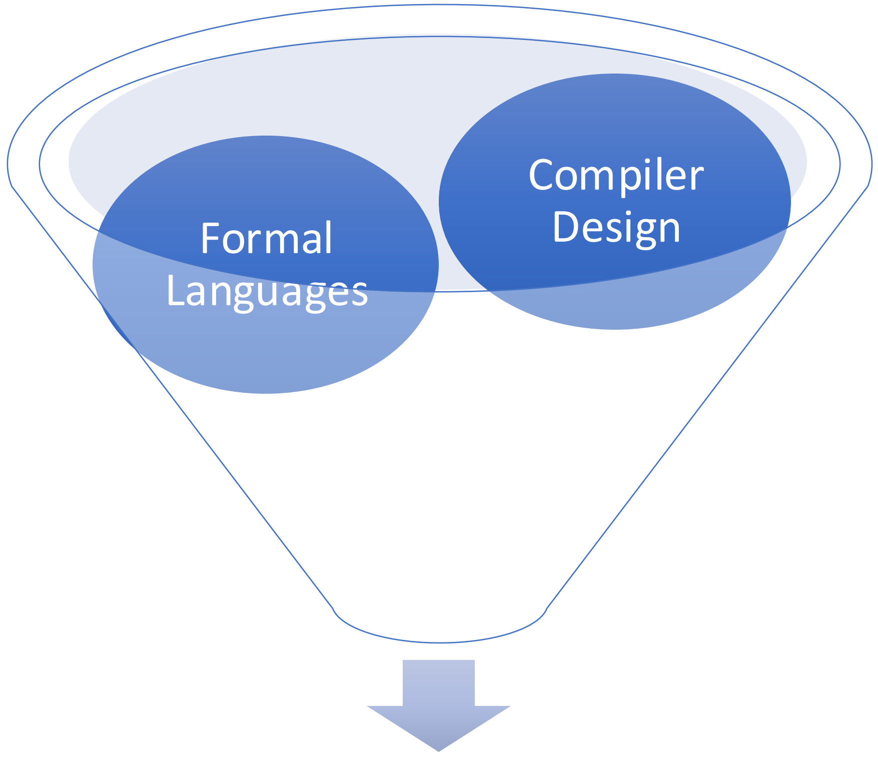
Formal languages and Compiler Design

Simona Motogna

Why?



FLCD

Teams Channel

dh0ue7t

Organization Issues

MATE - INFO

- Course – 2 h/ week
- Seminar – 1h/week
- Laboratory - 1h/week

PRESENCE IS MANDATORY

5 presences – seminar
6 presences - lab

ARTIFICIAL INTELLIGENCE

- Course – 2 h/ week
- Seminar – 2h/week
- Laboratory - 2h/week

PRESENCE IS MANDATORY

10 presences – seminar
12 presences - lab

Organization Issues

- Final grade =

  60% written exam

  + 30% lab

  + 10% seminar

Lab:

- all laboratory assignments are mandatory
- delays NO more than 2 weeks

Seminar:

- solved problems, answers (blackboard)

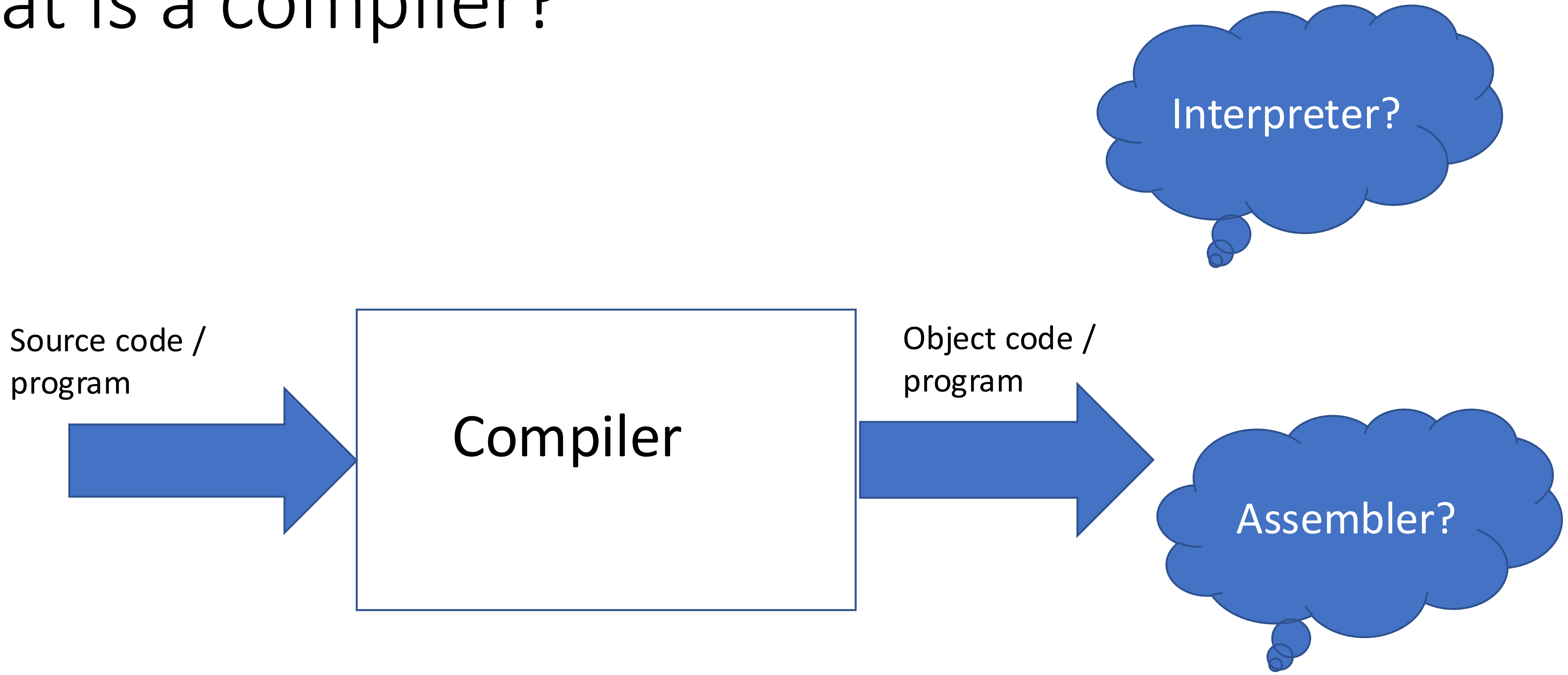
Requirements

- Lab assignments are supposed to be completed during lab and evaluated during lab
- Labs must be uploaded as specified by academic staff at the lab
- Rule for justified absence: present document to the teacher the week after; otherwise, only as specified by Faculty regulation
- Cheating is not accepted: others work or genAI work (unless specified in the lab requirements)

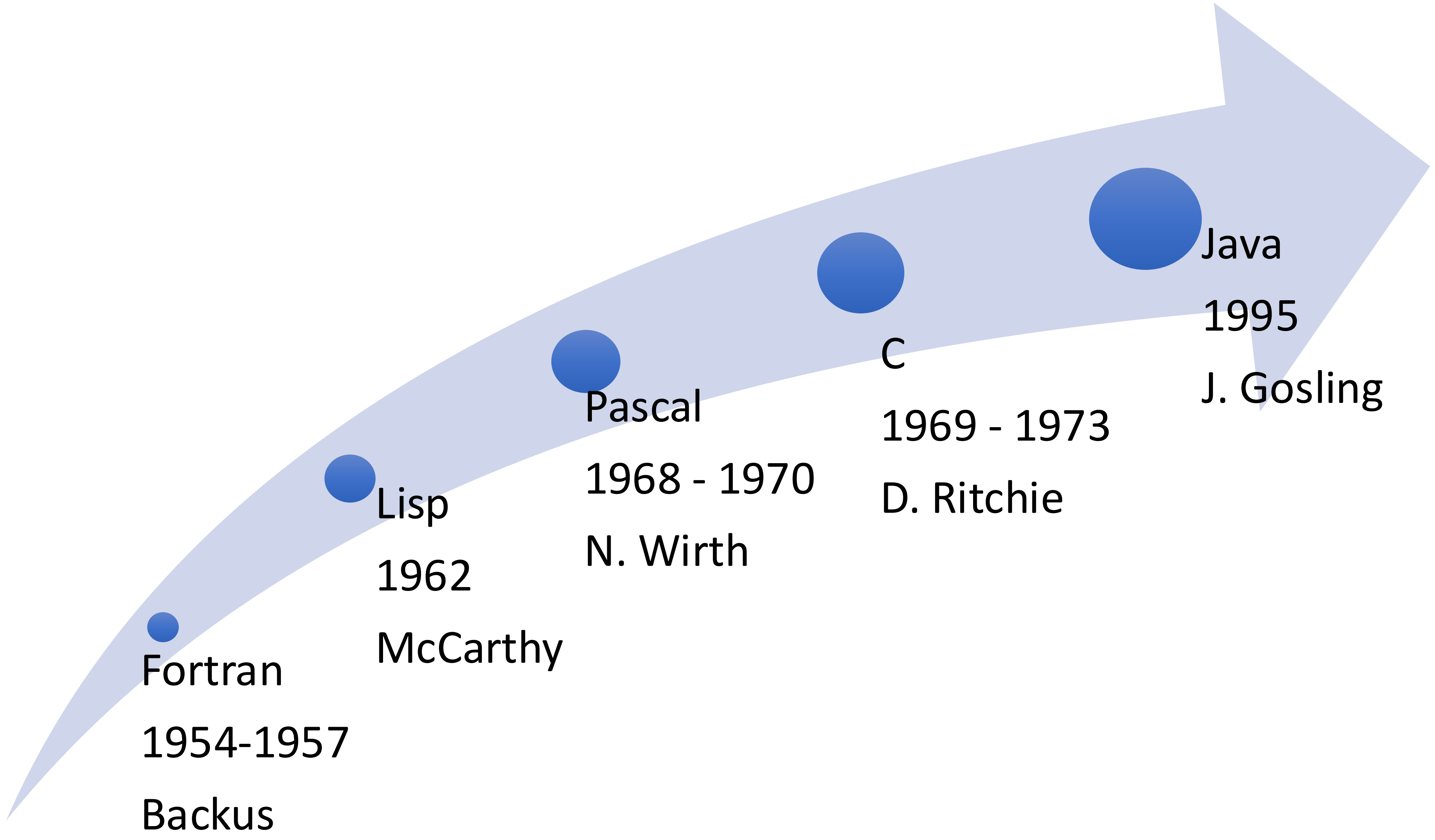
References

- See fișa disciplinei

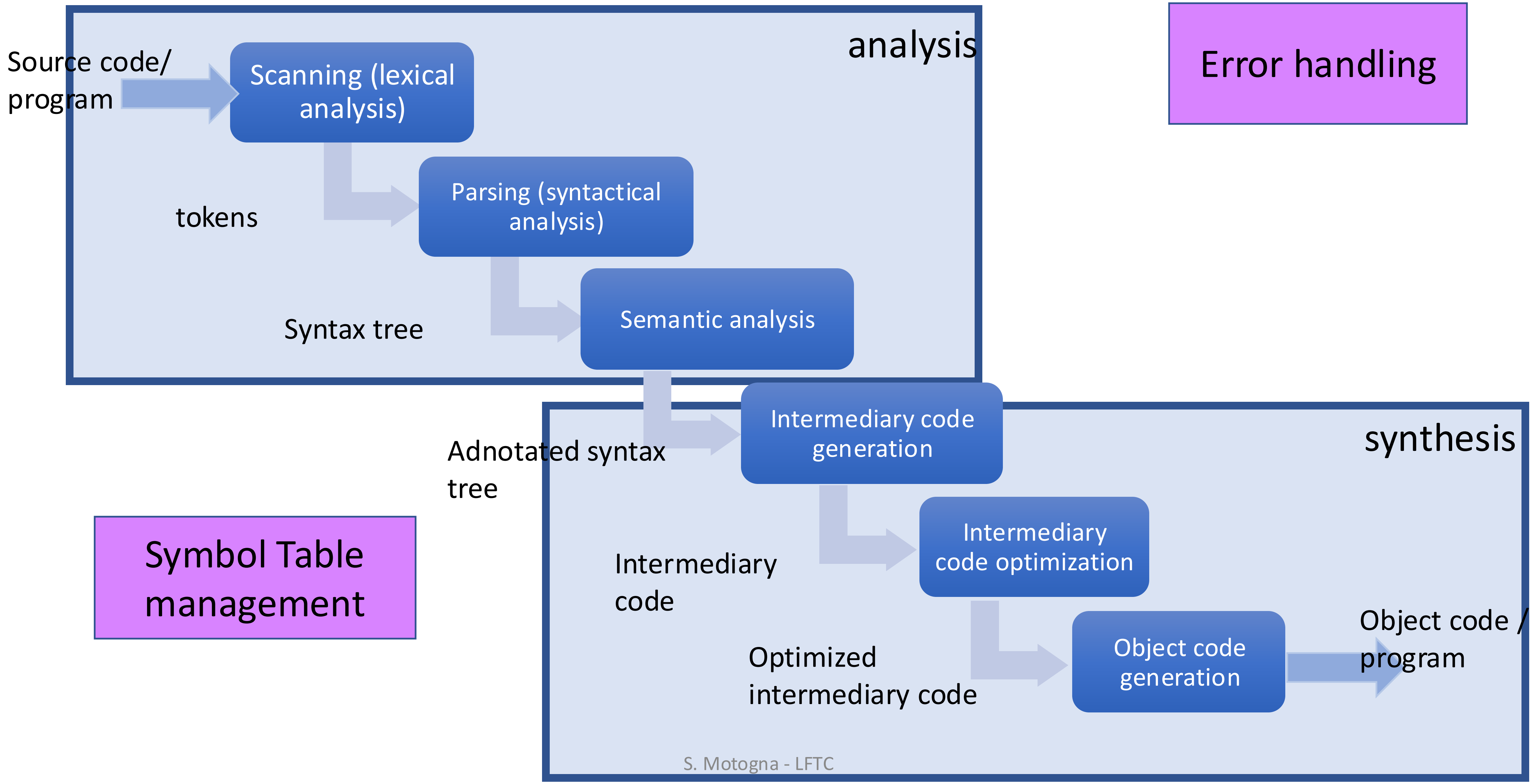
What is a compiler?



A little bit of history …



Structure of a compiler



Chapter 1. Scanning

Definition = treats the source program as a sequence of characters, detect lexical tokens, classify and codify them

INPUT: source program
OUTPUT: PIF + ST

Algorithm Scanning v1

```
While (not(eof)) do
    detect(token);
    classify(token);
    codify(token);
End_while
```

Classify

- Classes of tokens:
  - Identifiers
  - Constants
  - Reserved words (keywords)
  - Separators
  - Operators

- If a token can NOT be classified => LEXICAL ERROR

Symbol Table

Definition *= contains all information collected during compiling regarding the symbolic names from the source program*

↑

identifiers, constants, etc.

Variants:

- Unique symbol table – contains all symbolic names
- distinct symbol tables: IT (identifiers table) + CT (constants table)

ST organization

Remark: search and insert

1.	Unsorted table – in order of detection in source code	O(n)
2.	Sorted table: alphabetic (numeric)	O(lg n)
3.	Binary search tree (balanced)	O(lg n)
4.	Hash table	O(1)

Hash table

- K = set of keys (symbolic names)
- A = set of positions (|A| = m; m –prime number)

$h : K \rightarrow A$

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

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

Visibility domain (scope)

- Each scope – separate ST
- Structure -> inclusion tree