Formal Languages and Compiler Design

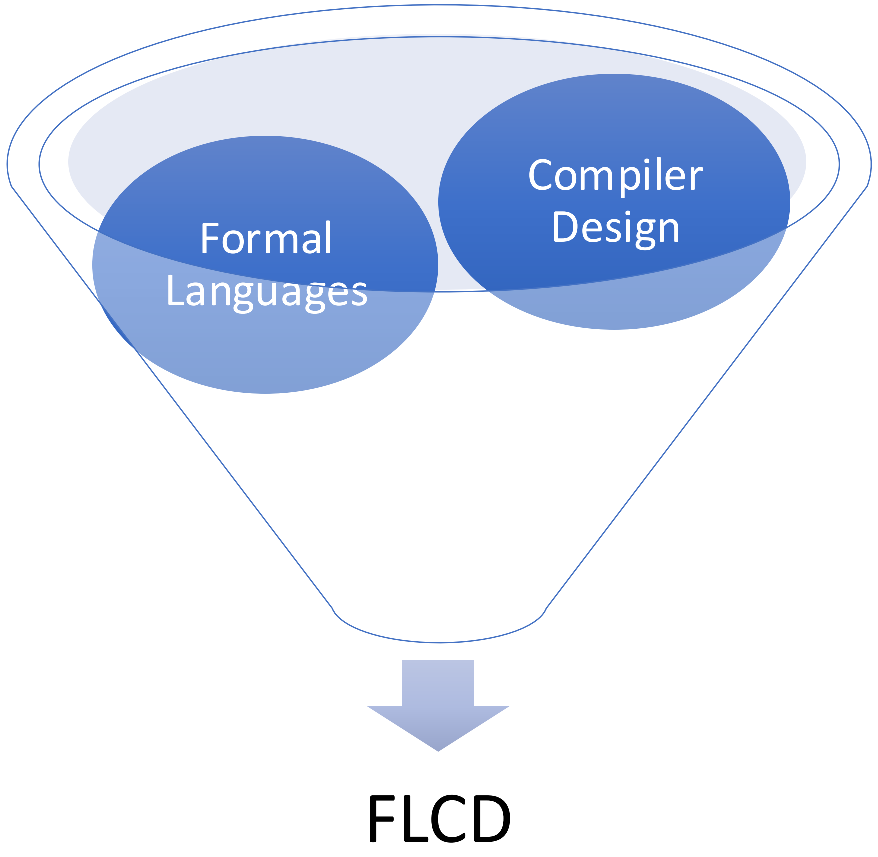
Simona Motogna

Why?

Historical reasons

Be a better programmer

Performant algorithms



Organization Issues

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

10 presences – seminar (75%)
12 presences – lab (80%)

PRESENCE IS MANDATORY

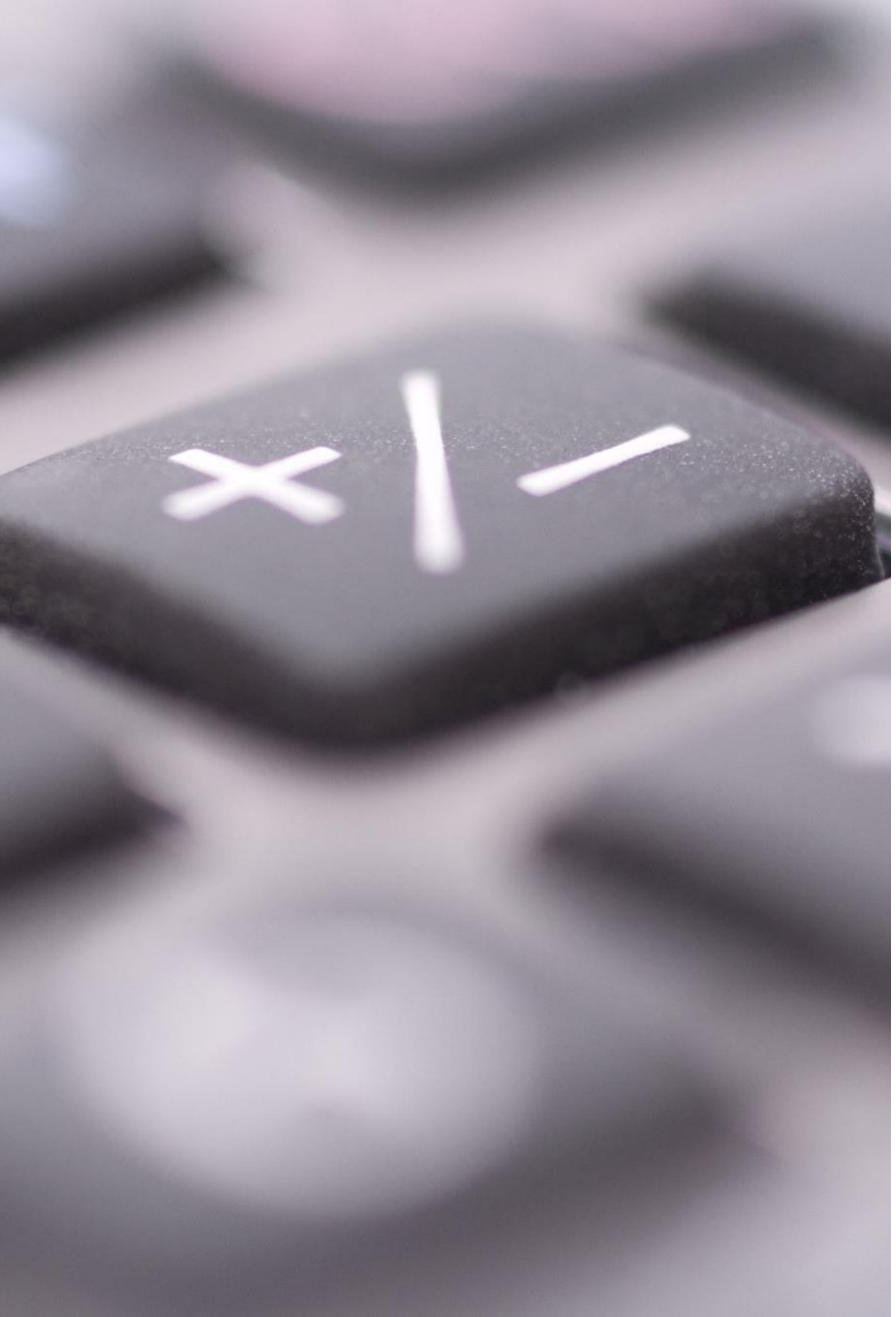
Most interesting stuff for students

- **MsTeams** – communication chanel, code: **s0bgl6w**
  - All course resources
  - Lab assignments
  - communication
- **Github classroom**: [formal-langages-and-compiler-design]….

Minimal Conditions to Pass

- *Minimum 10 presences at seminar*
- *Minimum 12 presences at laboratory*

- *Minimum grade **5** at lab*
- *Minimum grade 5 at final exam*



Final grade

60% final exam

+

30% lab

+

10% seminar

Bonus

Lab work

- 10 laboratory tasks

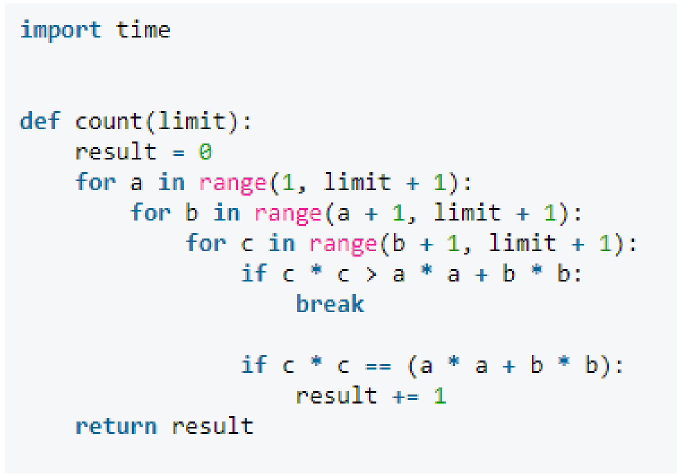
- !!! Must be completed and loaded during lab hours

Bonus points:

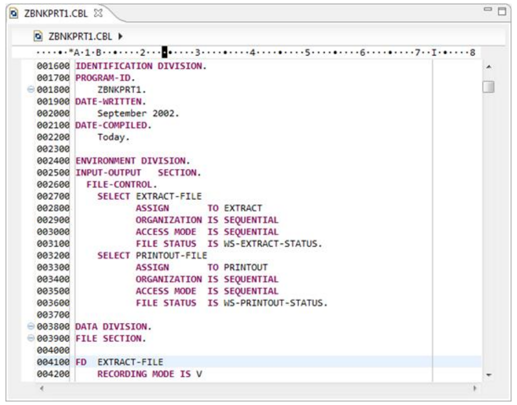
- “awesome” solutions
- Extra work

References

- See fișa disciplinei

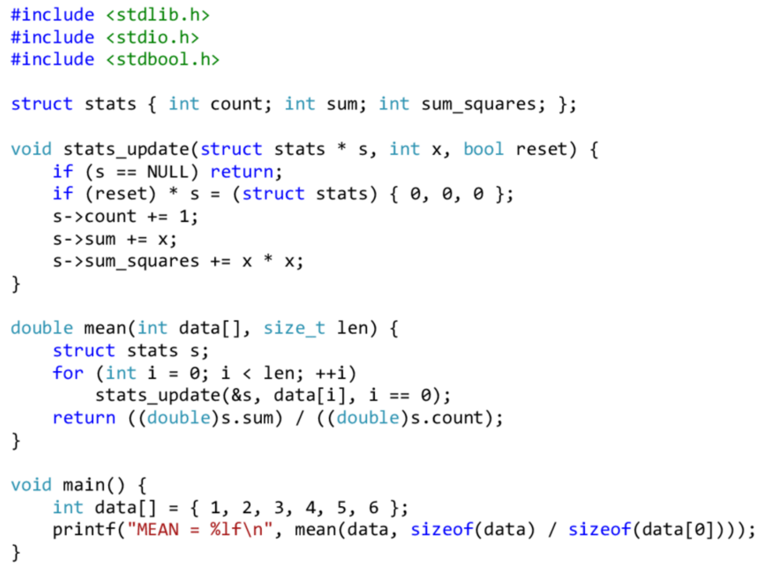


```python
import time


def count(limit):
    result = 0
    for a in range(1, limit + 1):
        for b in range(a + 1, limit + 1):
            for c in range(b + 1, limit + 1):
                if c * c > a * a + b * b:
                    break

                if c * c == (a * a + b * b):
                    result += 1
    return result
```

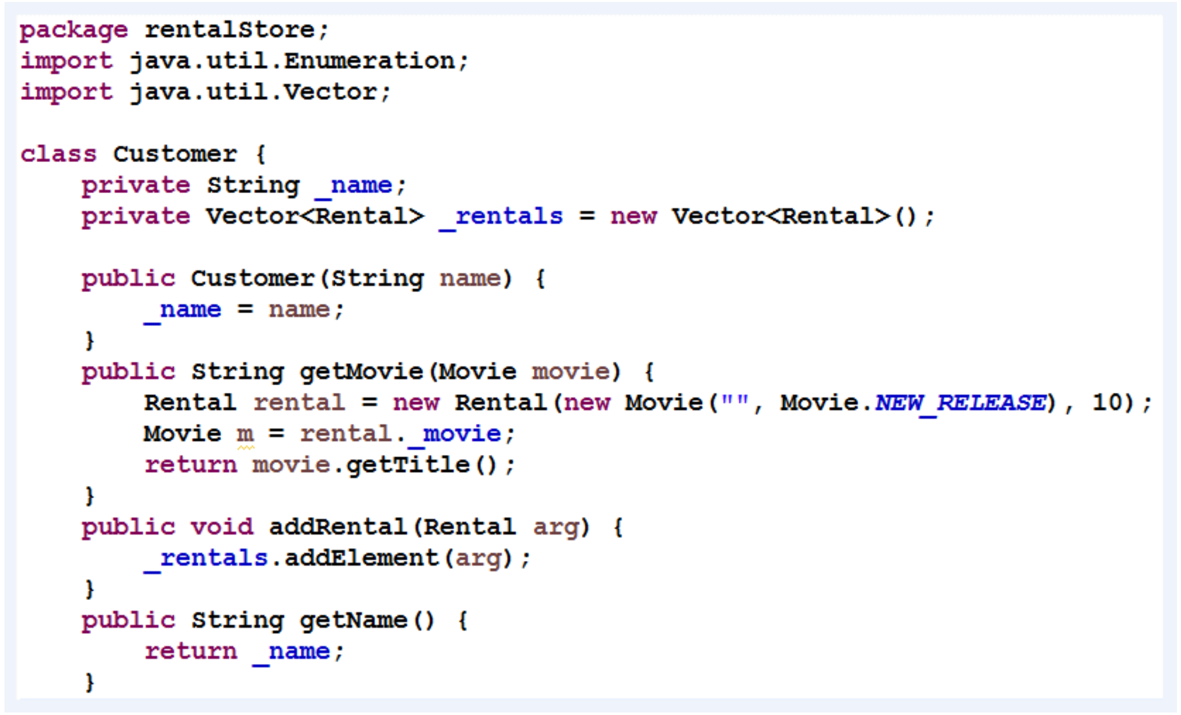


ZBNKPRT1.CBL
ZBNKPRT1.CBL
·····•·*A·1·B··•····2···•····3····•····4····•····5····•····6····•····7··I·•····8

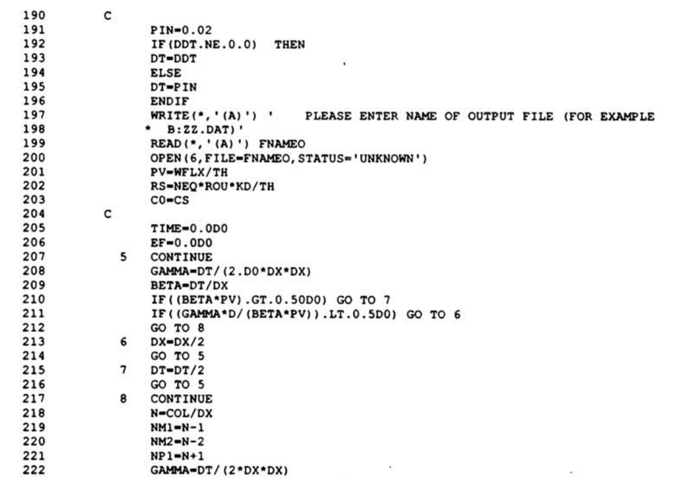
```cobol
001600 IDENTIFICATION DIVISION.
001700 PROGRAM-ID.
001800     ZBNKPRT1.
001900 DATE-WRITTEN.
002000     September 2002.
002100 DATE-COMPILED.
002200     Today.
002300
002400 ENVIRONMENT DIVISION.
002500 INPUT-OUTPUT   SECTION.
002600   FILE-CONTROL.
002700     SELECT EXTRACT-FILE
002800            ASSIGN       TO EXTRACT
002900            ORGANIZATION IS SEQUENTIAL
003000            ACCESS MODE  IS SEQUENTIAL
003100            FILE STATUS  IS WS-EXTRACT-STATUS.
003200     SELECT PRINTOUT-FILE
003300            ASSIGN       TO PRINTOUT
003400            ORGANIZATION IS SEQUENTIAL
003500            ACCESS MODE  IS SEQUENTIAL
003600            FILE STATUS  IS WS-PRINTOUT-STATUS.
003700
003800 DATA DIVISION.
003900 FILE SECTION.
004000
004100 FD  EXTRACT-FILE
004200     RECORDING MODE IS V
```



```c
#include <stdlib.h>
#include <stdio.h>
#include <stdbool.h>

struct stats { int count; int sum; int sum_squares; };

void stats_update(struct stats * s, int x, bool reset) {
    if (s == NULL) return;
    if (reset) * s = (struct stats) { 0, 0, 0 };
    s->count += 1;
    s->sum += x;
    s->sum_squares += x * x;
}

double mean(int data[], size_t len) {
    struct stats s;
    for (int i = 0; i < len; ++i)
        stats_update(&s, data[i], i == 0);
    return ((double)s.sum) / ((double)s.count);
}

void main() {
    int data[] = { 1, 2, 3, 4, 5, 6 };
    printf("MEAN = %lf\n", mean(data, sizeof(data) / sizeof(data[0])));
}
```

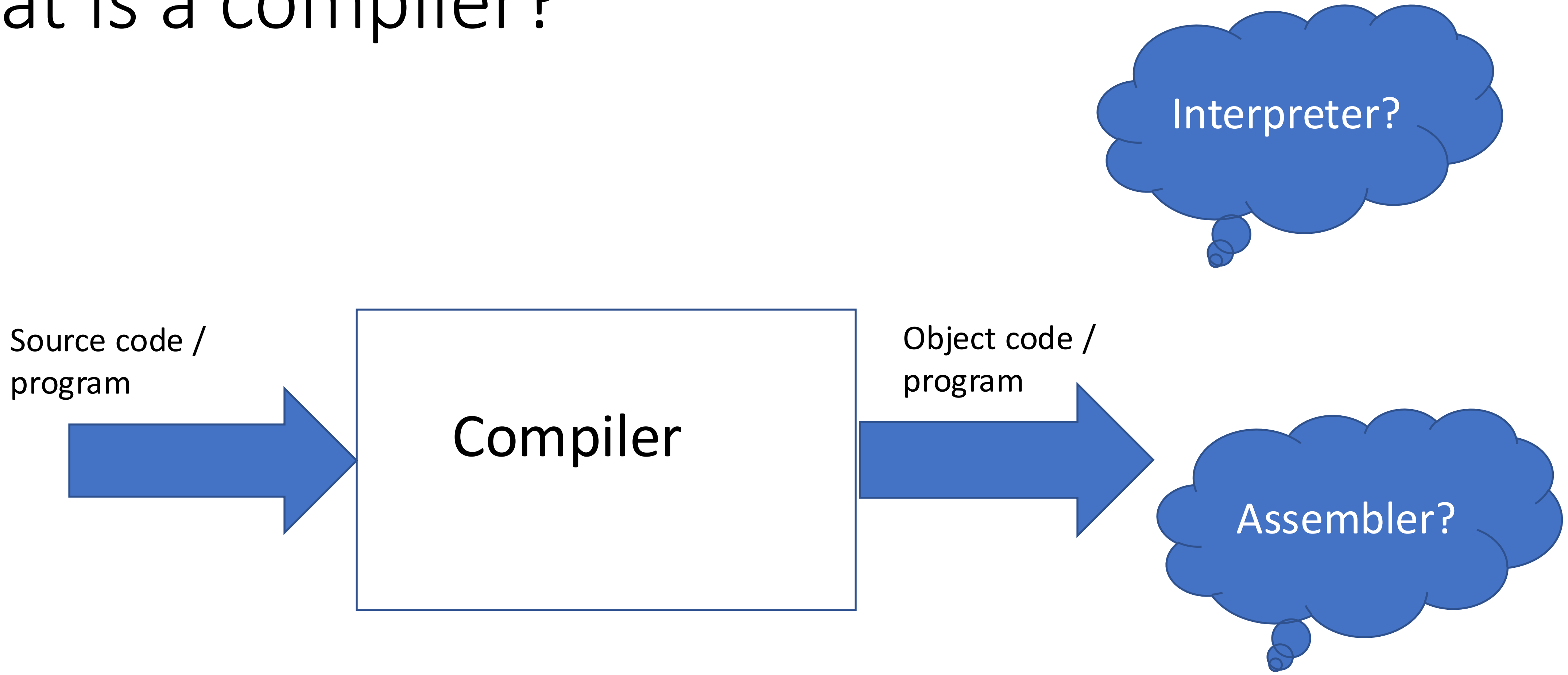


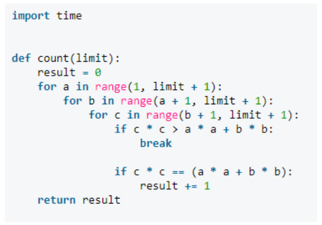
```java
package rentalStore;
import java.util.Enumeration;
import java.util.Vector;

class Customer {
    private String _name;
    private Vector<Rental> _rentals = new Vector<Rental>();

    public Customer(String name) {
        _name = name;
    }
    public String getMovie(Movie movie) {
        Rental rental = new Rental(new Movie("", Movie.NEW_RELEASE), 10);
        Movie m = rental._movie;
        return movie.getTitle();
    }
    public void addRental(Rental arg) {
        _rentals.addElement(arg);
    }
    public String getName() {
        return _name;
    }
```



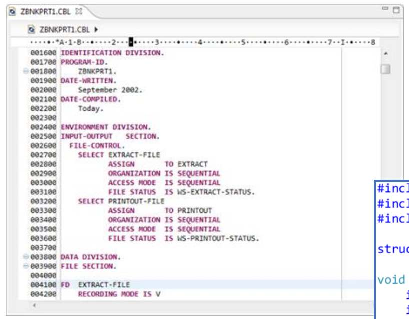
```fortran
C
      PIN=0.02
      IF(DDT.NE.0.0)  THEN
      DT=DDT
      ELSE
      DT=PIN
      ENDIF
      WRITE(*,'(A)') '    PLEASE ENTER NAME OF OUTPUT FILE (FOR EXAMPLE
     *  B:ZZ.DAT)'
      READ(*,'(A)') FNAMEO
      OPEN(6,FILE=FNAMEO,STATUS='UNKNOWN')
      PV=WFLX/TH
      RS=NEQ*ROU*KD/TH
      C0=CS
C
      TIME=0.0D0
      EF=0.0D0
    5 CONTINUE
      GAMMA=DT/(2.D0*DX*DX)
      BETA=DT/DX
      IF((BETA*PV).GT.0.50D0) GO TO 7
      IF((GAMMA*D/(BETA*PV)).LT.0.5D0) GO TO 6
      GO TO 8
    6 DX=DX/2
      GO TO 5
    7 DT=DT/2
      GO TO 5
    8 CONTINUE
      N=COL/DX
      NM1=N-1
      NM2=N-2
      NP1=N+1
      GAMMA=DT/(2*DX*DX)
```

What is a compiler?



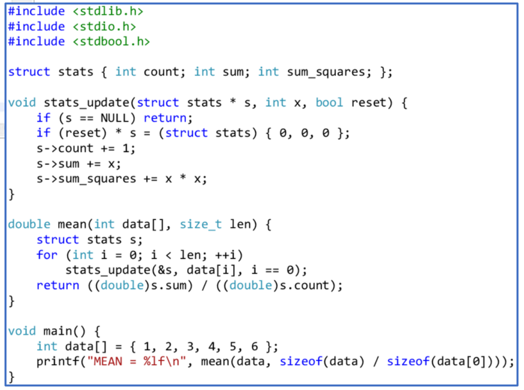


```python
import time

def count(limit):
    result = 0
    for a in range(1, limit + 1):
        for b in range(a + 1, limit + 1):
            for c in range(b + 1, limit + 1):
                if c * c > a * a + b * b:
                    break

                if c * c == (a * a + b * b):
                    result += 1
    return result
```

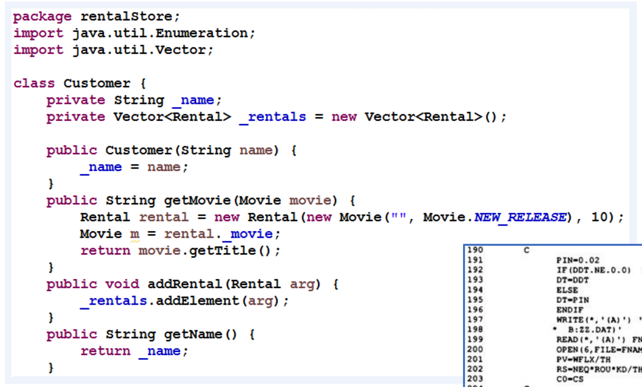


ZBNKPRT1.CBL

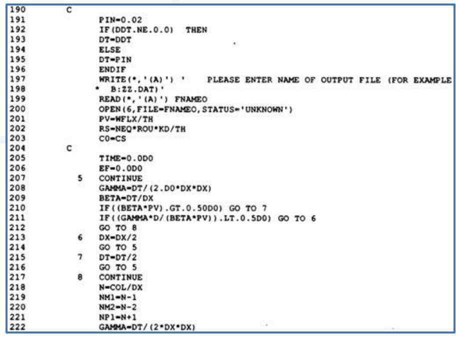
```cobol
001600 IDENTIFICATION DIVISION.
001700 PROGRAM-ID.
001800     ZBNKPRT1.
001900 DATE-WRITTEN.
002000     September 2002.
002100 DATE-COMPILED.
002200     Today.
002300
002400 ENVIRONMENT DIVISION.
002500 INPUT-OUTPUT   SECTION.
002600   FILE-CONTROL.
002700     SELECT EXTRACT-FILE
002800            ASSIGN       TO EXTRACT
002900            ORGANIZATION IS SEQUENTIAL
003000            ACCESS MODE  IS SEQUENTIAL
003100            FILE STATUS  IS WS-EXTRACT-STATUS.
003200     SELECT PRINTOUT-FILE
003300            ASSIGN       TO PRINTOUT
003400            ORGANIZATION IS SEQUENTIAL
003500            ACCESS MODE  IS SEQUENTIAL
003600            FILE STATUS  IS WS-PRINTOUT-STATUS.
003700
003800 DATA DIVISION.
003900 FILE SECTION.
004000
004100 FD  EXTRACT-FILE
004200     RECORDING MODE IS V
```



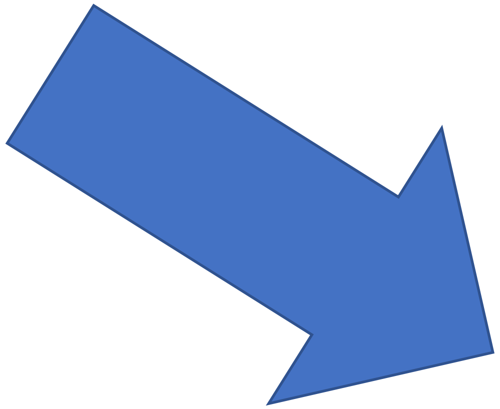
```c
#include <stdlib.h>
#include <stdio.h>
#include <stdbool.h>

struct stats { int count; int sum; int sum_squares; };

void stats_update(struct stats * s, int x, bool reset) {
    if (s == NULL) return;
    if (reset) * s = (struct stats) { 0, 0, 0 };
    s->count += 1;
    s->sum += x;
    s->sum_squares += x * x;
}

double mean(int data[], size_t len) {
    struct stats s;
    for (int i = 0; i < len; ++i)
        stats_update(&s, data[i], i == 0);
    return ((double)s.sum) / ((double)s.count);
}

void main() {
    int data[] = { 1, 2, 3, 4, 5, 6 };
    printf("MEAN = %lf\n", mean(data, sizeof(data) / sizeof(data[0])));
}
```

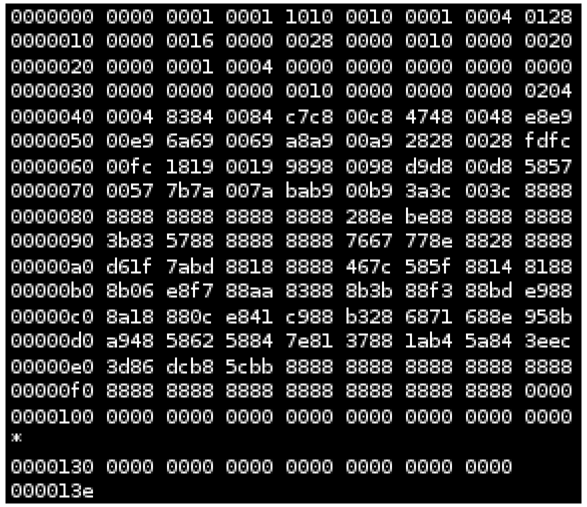


```java
package rentalStore;
import java.util.Enumeration;
import java.util.Vector;

class Customer {
    private String _name;
    private Vector<Rental> _rentals = new Vector<Rental>();

    public Customer(String name) {
        _name = name;
    }
    public String getMovie(Movie movie) {
        Rental rental = new Rental(new Movie("", Movie.NEW_RELEASE), 10);
        Movie m = rental._movie;
        return movie.getTitle();
    }
    public void addRental(Rental arg) {
        _rentals.addElement(arg);
    }
    public String getName() {
        return _name;
    }
```



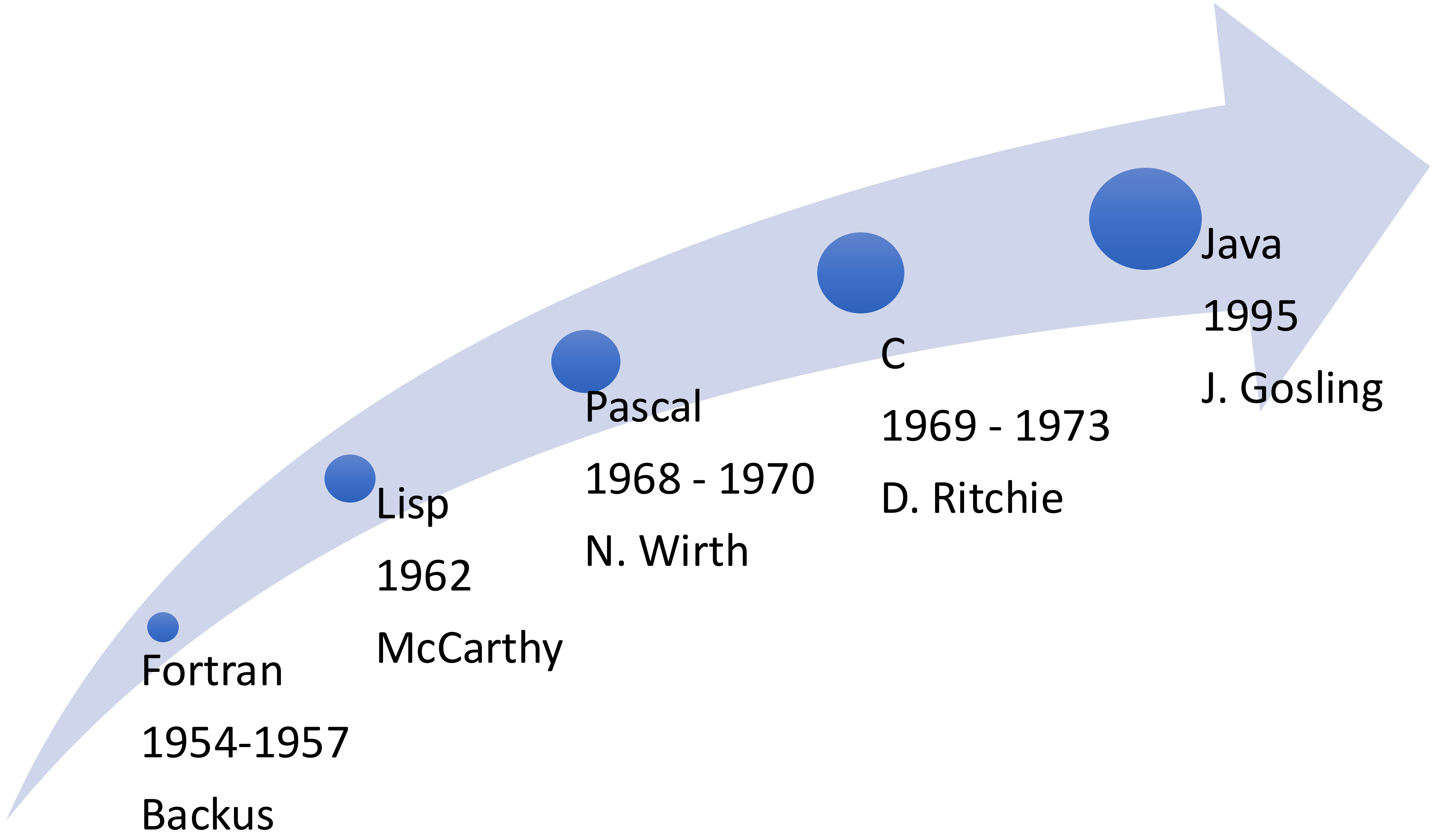
```fortran
C
      PIN=0.02
      IF(DDT.NE.0.0)  THEN
      DT=DDT
      ELSE
      DT=PIN
      ENDIF
      WRITE(*,'(A)') '    PLEASE ENTER NAME OF OUTPUT FILE (FOR EXAMPLE
     *  B:ZZ.DAT)'
      READ(*,'(A)') FNAMEO
      OPEN(6,FILE=FNAMEO,STATUS='UNKNOWN')
      PV=WFLX/TH
      RS=NEQ*ROU*KD/TH
      C0=CS
C
      TIME=0.0D0
      EF=0.0D0
    5 CONTINUE
      GAMMA=DT/(2.D0*DX*DX)
      BETA=DT/DX
      IF((BETA*PV).GT.0.50D0) GO TO 7
      IF((GAMMA*D/(BETA*PV)).LT.0.5D0) GO TO 6
      GO TO 8
    6 DX=DX/2
      GO TO 5
    7 DT=DT/2
      GO TO 5
    8 CONTINUE
      N=COL/DX
      NM1=N-1
      NM2=N-2
      NP1=N+1
      GAMMA=DT/(2*DX*DX)
```



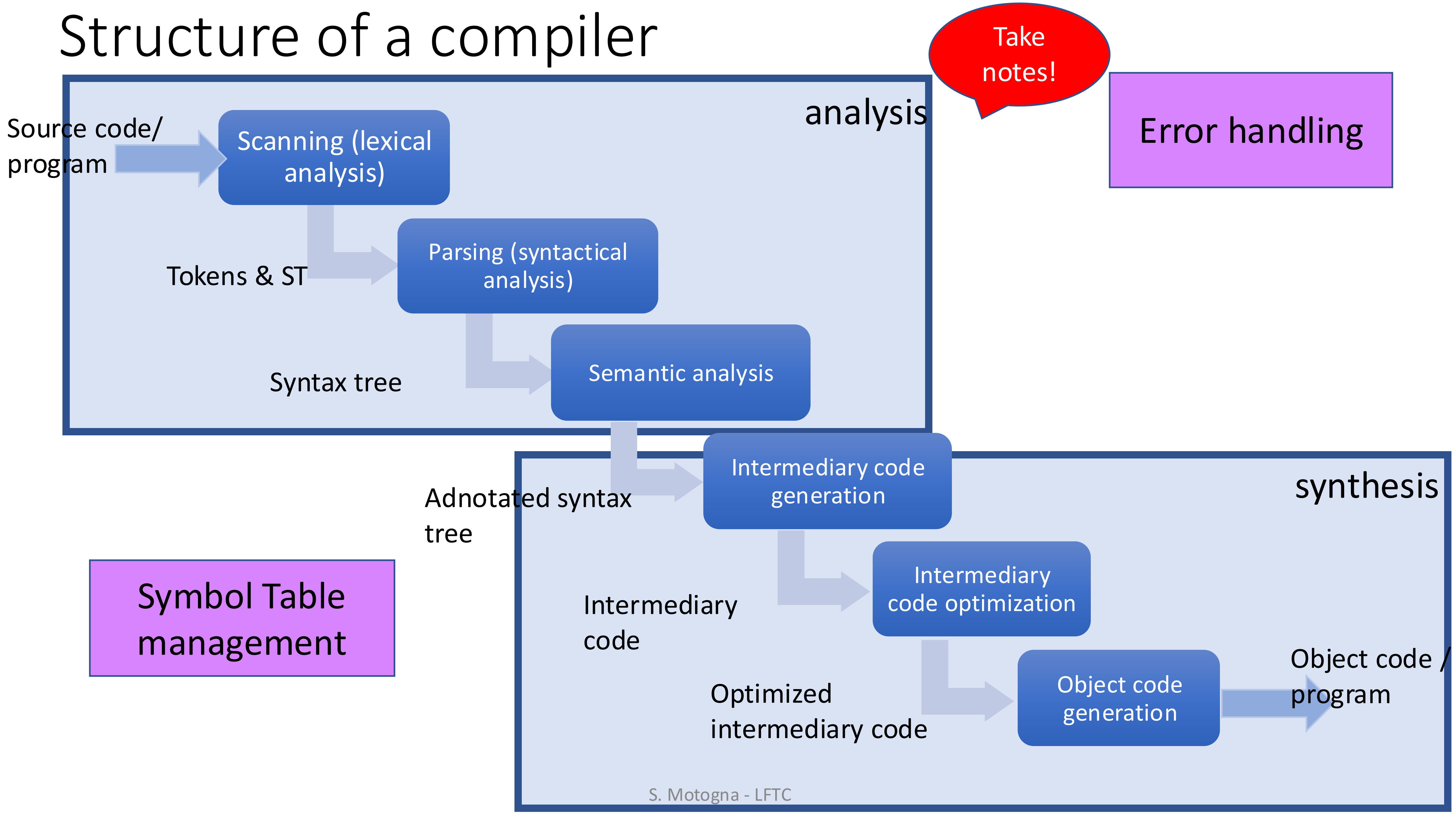


```
0000000 0000 0001 0001 1010 0010 0001 0004 0128
0000010 0000 0016 0000 0028 0000 0010 0000 0020
0000020 0000 0001 0004 0000 0000 0000 0000 0000
0000030 0000 0000 0000 0010 0000 0000 0000 0204
0000040 0004 8384 0084 c7c8 00c8 4748 0048 e8e9
0000050 00e9 6a69 0069 a8a9 00a9 2828 0028 fdfc
0000060 00fc 1819 0019 9898 0098 d9d8 00d8 5857
0000070 0057 7b7a 007a bab9 00b9 3a3c 003c 8888
0000080 8888 8888 8888 8888 288e be88 8888 8888
0000090 3b83 5788 8888 8888 7667 778e 8828 8888
00000a0 d61f 7abd 8818 8888 467c 585f 8814 8188
00000b0 8b06 e8f7 88aa 8388 8b3b 88f3 88bd e988
00000c0 8a18 880c e841 c988 b328 6871 688e 958b
00000d0 a948 5862 5884 7e81 3788 1ab4 5a84 3eec
00000e0 3d86 dcb8 5cbb 8888 8888 8888 8888 8888
00000f0 8888 8888 8888 8888 8888 8888 8888 0000
0000100 0000 0000 0000 0000 0000 0000 0000 0000
*
0000130 0000 0000 0000 0000 0000 0000 0000
000013e
```

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
```

Detect

Take notes!

I am a student.I   am Simona

- Separators => ***Remark 1)***

```
if (x==y) {x=y+2}
```

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

Classify

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

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

Codify

- May be codification table

OR

code for identifiers and constants

- Identifier, constant => Symbol Table (ST)

- PIF = Program Internal Form = array of pairs

- pairs (token, position in ST)

$\uparrow$

identifier, constant

Algorithm Scanning v2

```
While (not(eof)) do
      detect(token);
      if token is reserved word OR operator OR separator
             then genPIF(token, 0)
             else
            if token is identifier OR constant
                  then index = pos(token, ST);
                         genPIF(token, index)
                  else  message "Lexical error"
            endif
       endif
endwhile
```

a=a+b

FIP

(id,1)
(=,0)
(id,1)
(+,0)
(id,2)

ST

1	a
2	b

Remarks:

- genPIF = adds a pair (token, position) to PIF

- Pos(`token`,ST) – searches *`token`* in symbol table S*T*; if found then return position; if not found insert in SR and return position
- Order of classification (reserved word, then identifier)
- If-then-else imbricate => detect error if a token cannot be classified

Remarks:

- Also comments are eliminated
- Most important operations: SEARCH and INSERT

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

Toy hash function to use at lab:
Sum of ASCII codes of chars