

# Γλώσσες Προγραμματισμού I



Pieter Bruegel, *The Tower of Babel*, 1563

Κωστής Σαγώνας <[kostis@cs.ntua.gr](mailto:kostis@cs.ntua.gr)>  
Νίκος Παπασπύρου <[nickie@softlab.ntua.gr](mailto:nickie@softlab.ntua.gr)>

# Σχετικά με το μάθημα

---

Τι: μάθημα 6ου εξαμήνου ΣΗΜΜΥ, υποχρεωτικό στη Ροή Λ

Ιστοσελίδα: <https://courses.softlab.ntua.gr/pl1/>

Mailing list (moodle): πληροφορίες στην ιστοσελίδα

<https://coursespclab.ece.ntua.gr/course/view.php?id=59>

Πότε: κάθε Πέμπτη 15:15–17:00 και Παρασκευή 10:45-12:30

Πρόγραμμα:

- 18 διαλέξεις «θεωρίας»
- 6 «εργαστηριακά» μαθήματα στα περιβάλλοντα των γλωσσών
- 1 επαναληπτικό μάθημα στο τέλος (αν υπάρξει χρόνος)

# Σχετικά με το μάθημα

---

**Εργασίες:** Θα δοθούν συνολικά 3 σειρές ασκήσεων

- Θα αφορούν 4-5 προβλήματα
- καθένα από τα οποία θα πρέπει να λύσετε (συγκριτικά) σε περισσότερες από μία γλώσσες, μεταξύ των:
- C/C++, ML, Java, Python, Prolog

**Βαθμολογία:**

30% εργασίες

80% διαγώνισμα

# Συνεργασία μεταξύ φοιτητών

---

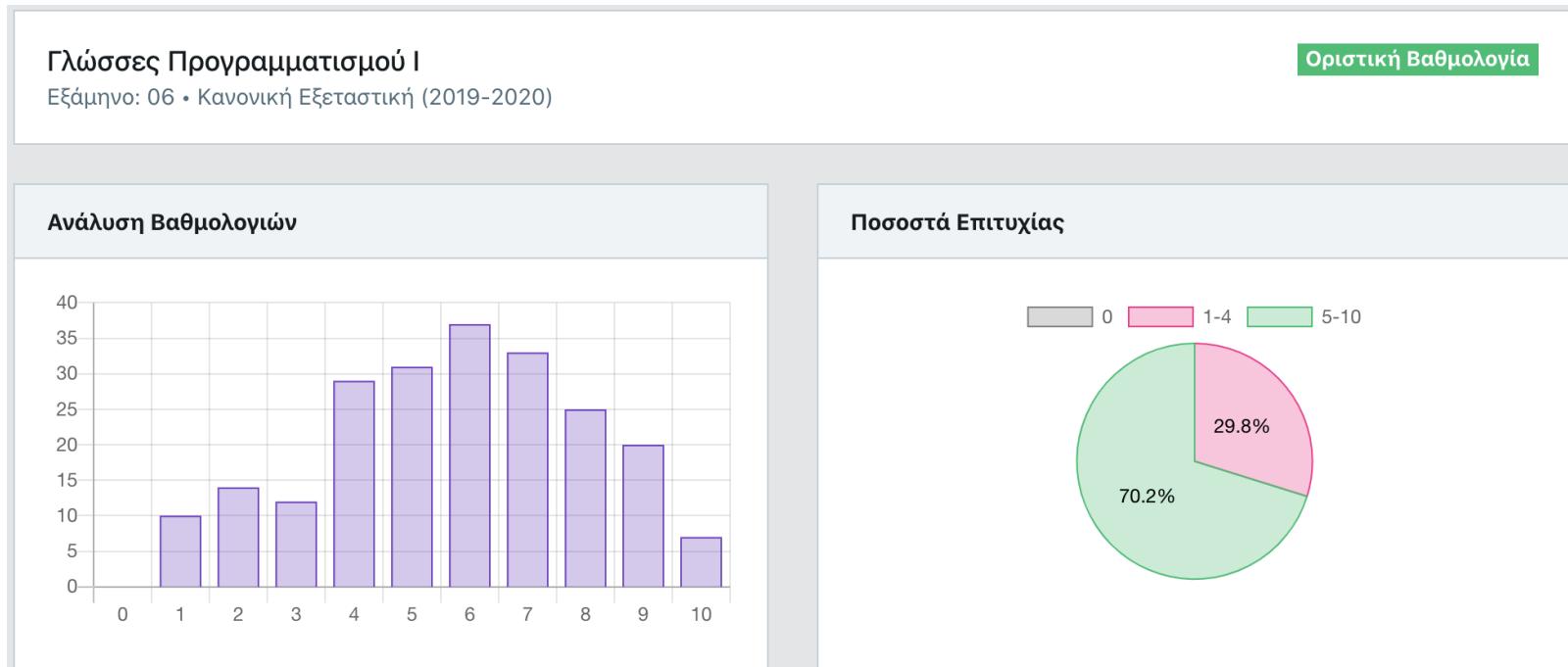
- **Οι προθεσμίες των εργασιών τηρούνται αυστηρά**
  - Μέσω του συστήματος ηλεκτρονικής υποβολής των εργασιών
- Οι εργασίες γίνονται σε ομάδες το πολύ δύο ατόμων
- Επιτρέπεται να συζητάτε ασκήσεις με τους συμφοιτητές σας, αλλά **οι εργασίες πρέπει να είναι δική σας δουλειά**
- Δεν επιτρέπεται να δίνετε την εργασία σας σε άλλους ή να τις βάλετε σε μέρος στο οποίο άλλοι έχουν πρόσβαση
- Σε περίπτωση που διαπιστωθούν φαινόμενα αντιγραφής, οι εργασίες αυτομάτως βαθμολογούνται με μηδέν και θα κινηθούν όλες οι προβλεπόμενες διαδικασίες

# Εξετάσεις

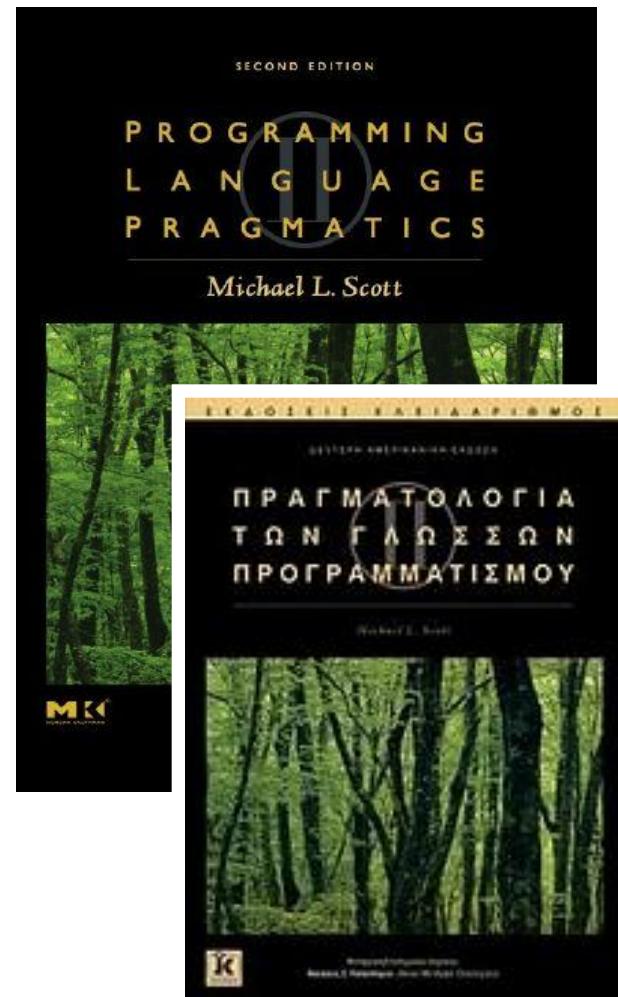
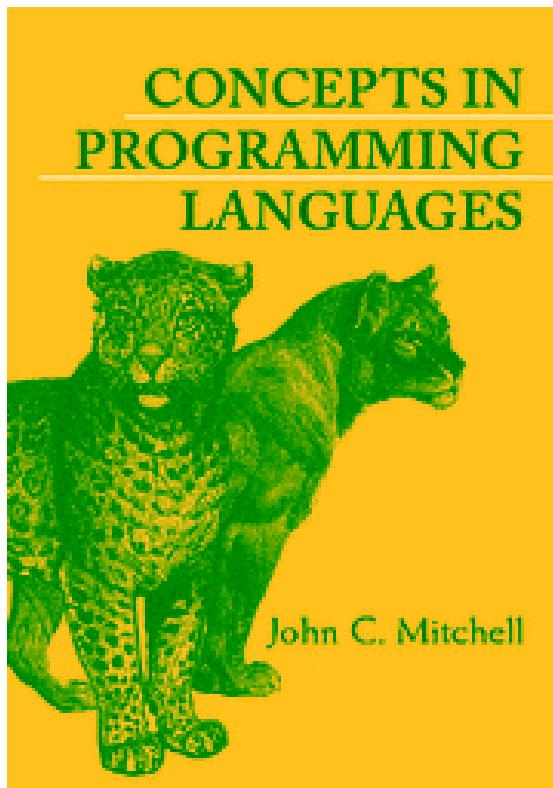
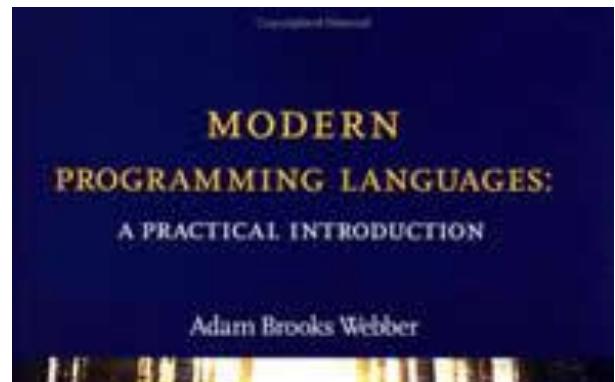
## Εξετάσεις: τον Ιούνιο και το Σεπτέμβριο

Μάλλον εξ αποστάσεως, αλλά με βαθμολογία 0-10

**Προηγούμενα θέματα:** στην ιστοσελίδα του μαθήματος και στο wiki / forum

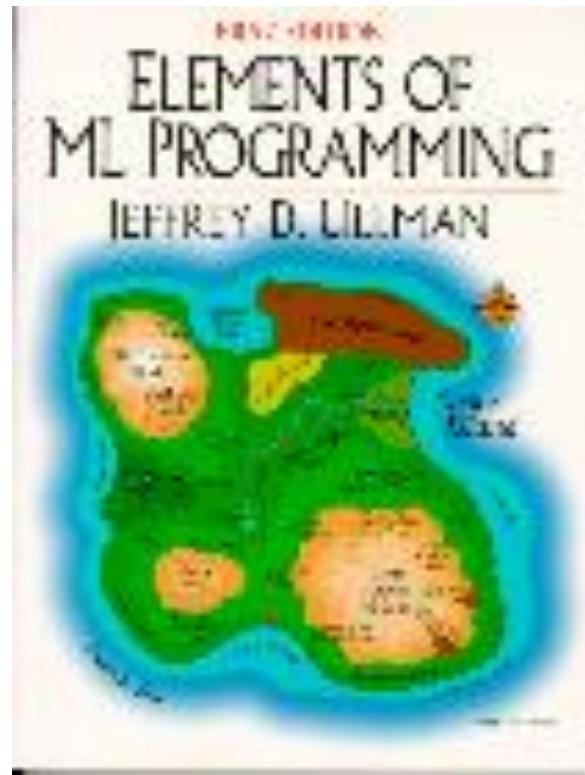
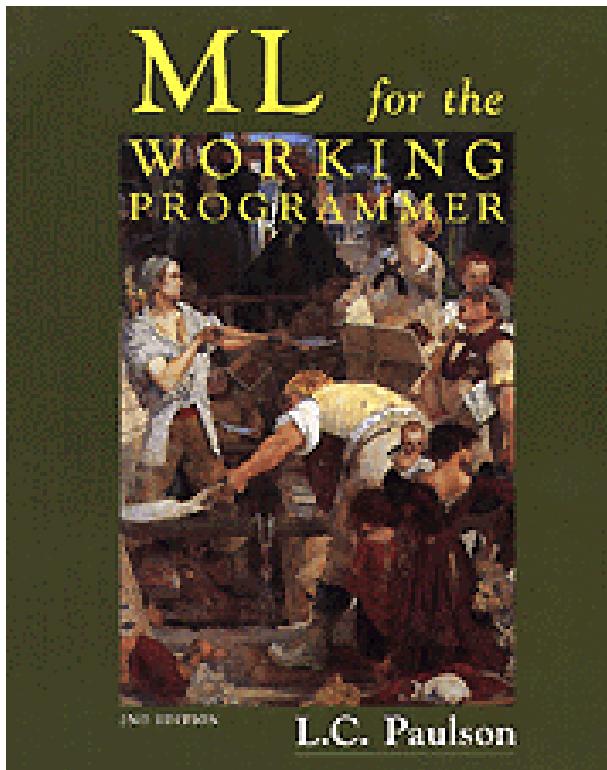


# Προτεινόμενα βιβλία

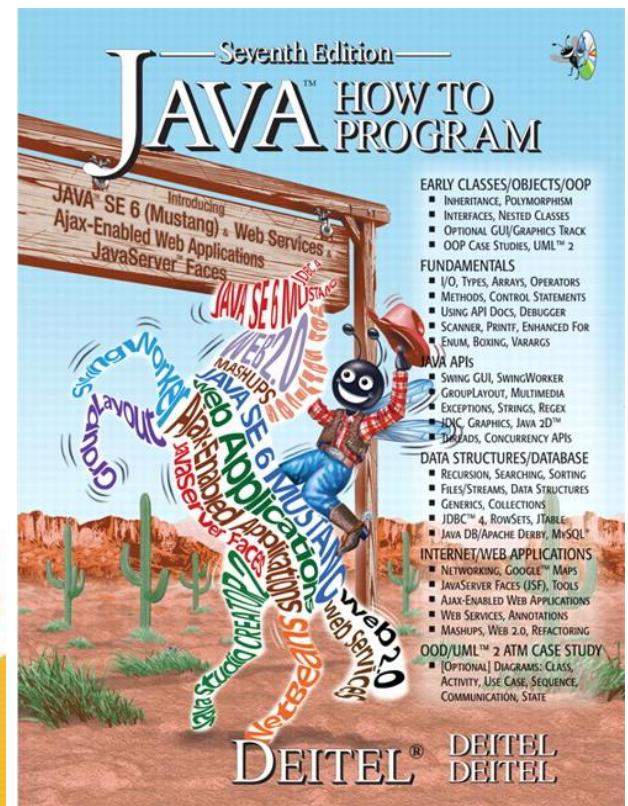
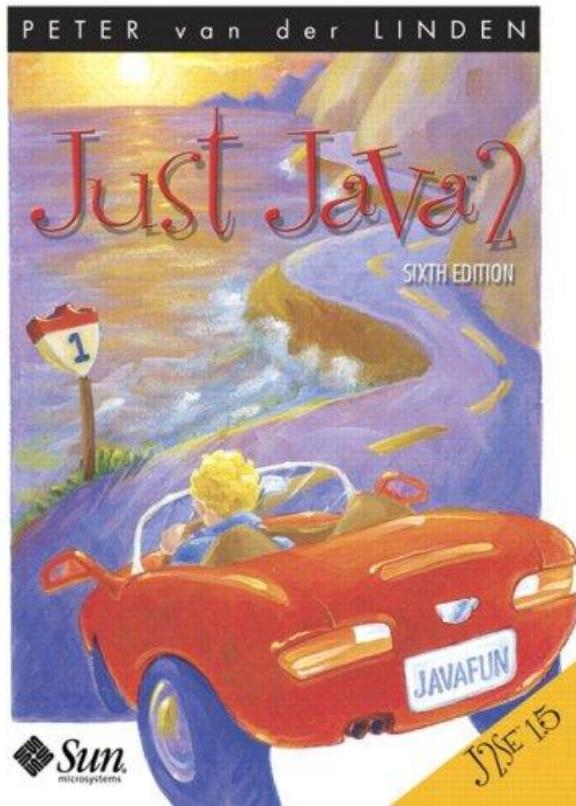
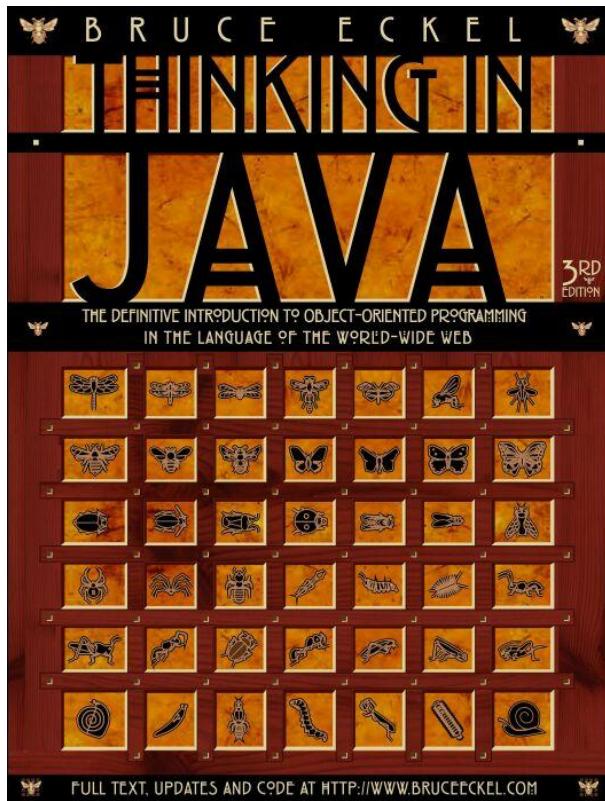


# Προτεινόμενα βιβλία για ML

---

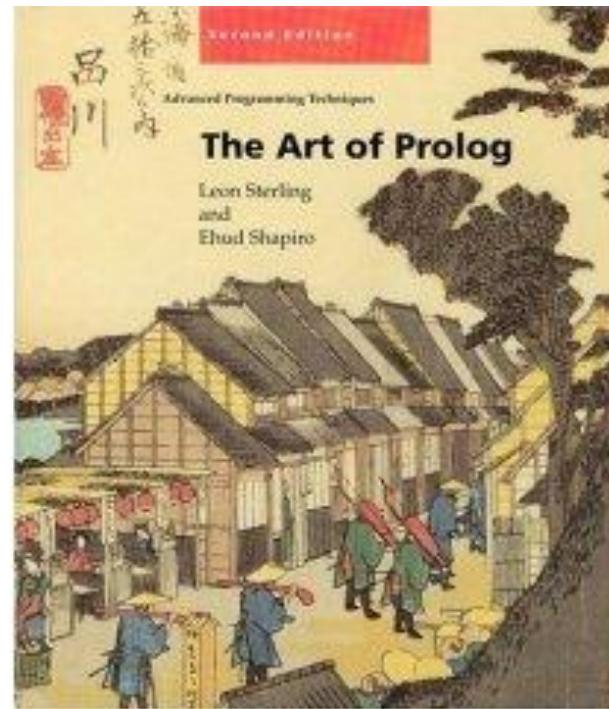
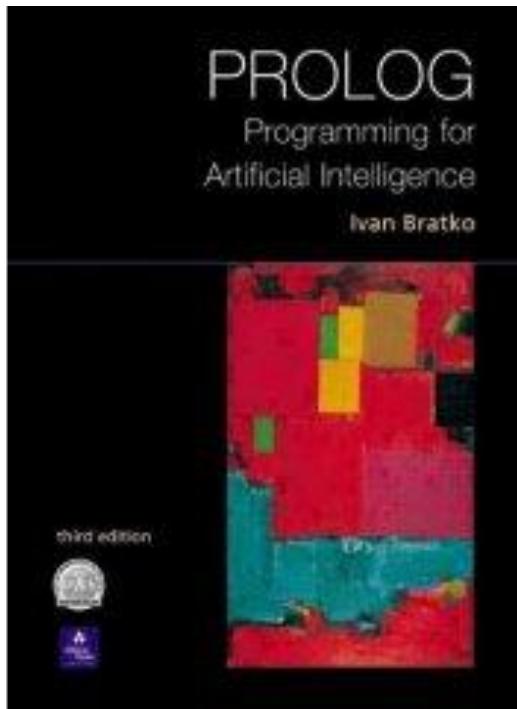


# Προτεινόμενα βιβλία για Java



# Προτεινόμενα βιβλία για Prolog

---



# Γιατί είναι ενδιαφέρουσες οι γλώσσες;

---

- Λόγω της ποικιλίας τους και των χαρακτηριστικών τους
- Λόγω των αμφιλεγόμενων στοιχείων τους
- Λόγω της ενδιαφέρουσας εξέλιξής τους
- Λόγω της στενής τους σχέσης με τον προγραμματισμό και την ανάπτυξη λογισμικού
- Λόγω του θεωρητικού τους υπόβαθρου και της στενής τους σχέσης με την επιστήμη των υπολογιστών

# Φοβερή ποικιλία γλωσσών προγραμματισμού

---

- Υπάρχουν πάρα πολλές και αρκετά διαφορετικές μεταξύ τους γλώσσες
- Το 1995, μια συλλογή που εμφανιζόταν συχνά στη λίστα comp.lang.misc περιλάμβανε πάνω από 2300 γλώσσες
- Οι γλώσσες συχνά κατατάσσονται στις εξής οικογένειες:
  - Προστακτικού προγραμματισμού (**Pascal, C, Ada**)
  - Συναρτησιακού προγραμματισμού (**Lisp, ML, Haskell, Erlang**)
  - Λογικού προγραμματισμού (**Prolog, Mercury**)
  - Αντικειμενοστρεφούς προγραμματισμού (**Smalltalk, C++, Java, C#**)
  - Γλώσσες σεναρίων (**Perl, Javascript, PHP, Python, Ruby**)

# Γλώσσες προστακτικού προγραμματισμού

---

Παράδειγμα: η συνάρτηση παραγοντικό στη C

```
int fact(int n) {  
    int f = 1;  
    while (n > 0) f *= n--;  
    return f;  
}
```

- Κύρια χαρακτηριστικά:
  - Ανάθεση μεταβλητών (πολλαπλή)
  - Επανάληψη
  - Η σειρά εκτέλεσης παίζει σημαντικό ρόλο

# Γλώσσες συναρτησιακού προγραμματισμού (1)

Παράδειγμα: η συνάρτηση παραγοντικό στην ML

```
fun fact x =  
  if x <= 0 then 1 else x * fact(x-1);
```

- Κύρια χαρακτηριστικά:
  - Μεταβλητές μιας τιμής
  - Η επανάληψη εκφράζεται με χρήση αναδρομής

# Γλώσσες συναρτησιακού προγραμματισμού (2)

Παράδειγμα: η συνάρτηση παραγοντικό στη Lisp

```
(defun fact (x)
  (if (<= x 0) 1 (* x (fact (- x 1)))))
```

- Συντακτικά, η συνάρτηση δείχνει αρκετά διαφορετική από ό,τι στην ML
- 'Όμως, η ML και η Lisp είναι συγγενείς γλώσσες

# Γλώσσες λογικού προγραμματισμού

Παράδειγμα: η συνάρτηση παραγοντικό στην Prolog

```
fact(X, F) :-  
    ( X =:= 1 -> F = 1  
    ; X > 1,  
      NewX is X - 1,  
      fact(NewX, NF),  
      F is X * NF  
    ).
```

- Κύρια χαρακτηριστικά:
  - Λογικές μεταβλητές και χρήση ενοποίησης
  - Το πρόγραμμα γράφεται με χρήση κανόνων λογικής
  - (Τα παραπάνω δε φαίνονται πολύ καθαρά στο συγκεκριμένο κώδικα)

# Γλώσσες αντικειμενοστρεφούς προγραμματισμού

Παράδειγμα: ορισμός στη Java ενός αντικειμένου που μπορεί να αποθηκεύσει έναν ακέραιο και να υπολογίσει το παραγοντικό του

```
public class MyInt {  
    private int value;  
    public MyInt(int value) {  
        this.value = value;  
    }  
    public int getValue() {  
        return value;  
    }  
    public MyInt getFact() {  
        return new MyInt(fact(value));  
    }  
    private int fact(int n) {  
        int f = 1;  
        while (n > 1) f *= n--;  
        return f;  
    }  
}
```

Κύρια χαρακτηριστικά:

- Ανάθεση
- Χρήση αντικειμένων: δεδομένων που έχουν κατάσταση και ξέρουν πώς
  - να τη μεταβάλλουν
  - να την γνωστοποιήσουν σε άλλα αντικείμενα

# Πλεονεκτήματα και μειονεκτήματα

---

- Συνήθως, διαφορετικές γλώσσες δείχνουν τα πλεονεκτήματά τους σε διαφορετικού είδους εφαρμογές
- Η έννοια της τέλειας γλώσσας προγραμματισμού δεν υφίσταται (αντικειμενικά)
- Αποφασίστε μόνοι σας στο τέλος του μαθήματος, με βάση:
  - την εμπειρία σας
  - τις προσωπικές σας προτιμήσεις
  - (Όχι με βάση τη συνάρτηση παραγοντικό!)

# Οικογένειες δε θίγουμε...

---

- Υπάρχουν πολλές οικογένειες γλωσσών  
(η λίστα είναι μη εξαντλητική και έχει επικαλύψεις)
  - Applicative, concurrent, constraint, declarative, definitional, procedural, scripting, single-assignment, ...
- Κάποιες γλώσσες ανήκουν σε πολλές οικογένειες
- Κάποιες άλλες είναι τόσο ιδιάζουσες που η κατάταξή τους σε κάποια οικογένεια δεν έχει μεγάλο νόημα

# Παράδειγμα: Παραγοντικό σε Forth

---

- Γλώσσα βασισμένη σε στοίβα (stack-oriented)

```
: FACTORIAL
    1 SWAP BEGIN ?DUP WHILE TUCK * SWAP 1- REPEAT ;
```

- Θα μπορούσε να χαρακτηριστεί προστακτική γλώσσα, αλλά έχει λίγα κοινά στοιχεία με τις περισσότερες προστακτικές γλώσσες

(Η γλώσσα Postscript είναι επίσης stack-oriented)

# Παράδειγμα: Παραγοντικό σε APL

---

$\times / \iota X$

- Μια έκφραση APL που υπολογίζει το παραγοντικό του  $X$
- Επεκτείνει το  $X$  σε ένα διάνυσμα (vector) από ακεραίους  $1..X$ , τους οποίους μετά πολλαπλασιάζει μεταξύ τους
- Θα μπορούσε να θεωρηθεί συναρτησιακή γλώσσα, αλλά έχει ελάχιστα κοινά στοιχεία με τις περισσότερες γλώσσες συναρτησιακού προγραμματισμού

(Για την ακρίβεια, δε θα το γράφαμε με αυτό τον τρόπο στην APL, γιατί η γλώσσα περιλαμβάνει το μοναδιαίο τελεστή παραγοντικό:  $!X$ )

## Αμφιλεγόμενα χαρακτηριστικά και “γλωσσοπόλεμοι”

---

- Οι γλώσσες πολλές φορές καταλήγουν το αντικείμενο έντονων διαξιφισμών για τα χαρακτηριστικά τους
- Κάθε γλώσσα έχει τόσο υποστηρικτές όσο και πολέμιους οι οποίοι συνήθως έχουν έντονες γνώμες και πιστεύω

Για προσωπική εμπειρία, παρακολουθήστε τα newsgroups:  
comp.lang.\*

ή τον ιστότοπο:

<http://lambda-the-ultimate.org/>

# Οι διακρίσεις και οι ορισμοί είναι λίγο ασαφείς

---

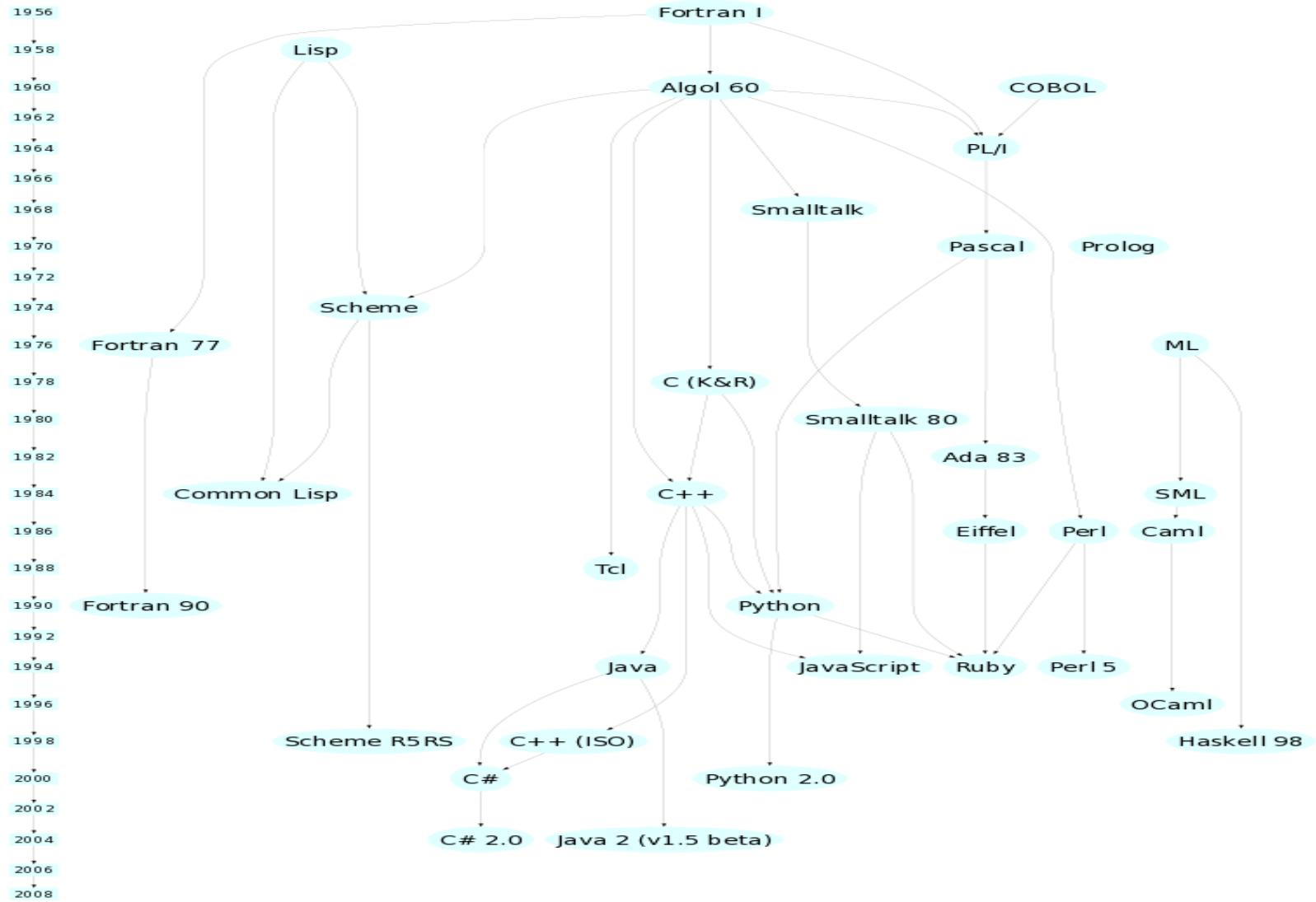
- Κάποιοι όροι αναφέρονται σε ασαφείς έννοιες
  - Για παράδειγμα, η κατηγοριοποίηση των γλωσσών σε οικογένειες
- Κανένα πρόβλημα, αν θυμάστε ότι κάποιοι όροι είναι σχετικά ασαφείς
  - Λάθος ερώτηση:
    - Είναι η γλώσσα X μια πραγματικά αντικειμενοστρεφής γλώσσα;
  - Σωστή ερώτηση:
    - Ποια χαρακτηριστικά της γλώσσας X υποστηρίζουν τον αντικειμενοστρεφή προγραμματισμό και πόσο καλά;

# Η φοβερή εξέλιξη των γλωσσών

---

- Οι γλώσσες προγραμματισμού εξελίσσονται με πολύ γρήγορο ρυθμό
  - Νέες γλώσσες δημιουργούνται
  - Παλιές γλώσσες αποκτούν διαλέκτους ή μεταλλάσσονται

# Εξέλιξη γλωσσών προγραμματισμού



# Assembly

---

Πριν: Αριθμοί

55  
89E5  
8B4508  
8B550C  
39D0  
740D  
39D0  
7E08  
29D0  
39D0  
75F6  
C9  
C3  
29C2  
EBF6

Μετά: Σύμβολα

gcd: **pushl %ebp**  
**movl %esp, %ebp**  
**movl 8(%ebp), %eax**  
**movl 12(%ebp), %edx**  
**cmpl %edx, %eax**  
**je .L9**  
.L7: **cmpl %edx, %eax**  
**jle .L5**  
**subl %edx, %eax**  
.L2: **cmpl %edx, %eax**  
**jne .L7**  
.L9: **leave**  
**ret**  
.L5: **subl %eax, %edx**  
**jmp .L2**

# FORTRAN (FORmula TRANslator)

---

Πριν: Σύμβολα

```
gcd: pushl %ebp  
      movl %esp, %ebp  
      movl 8(%ebp), %eax  
      movl 12(%ebp), %edx  
      cmpl %edx, %eax  
      je .L9  
.L7:  cmpl %edx, %eax  
      jle .L5  
      subl %edx, %eax  
.L2:  cmpl %edx, %eax  
      jne .L7  
.L9:  leave  
      ret  
.L5:  subl %eax, %edx  
      jmp .L2
```

Μετά: Εκφράσεις, έλεγχος ροής

```
10   IF (a .EQ. b) GOTO 20  
      IF (a .LT. b) THEN  
          a = a - b  
      ELSE  
          b = b - a  
      ENDIF  
      GOTO 10  
20   END
```

# COBOL

---

Δηλώσεις τύπων, εγγραφών, διαχείριση αρχείων

**data division.**

**file section.**

\* describe the input file

fd employee-file-in

label records standard

block contains 5 records

record contains 31 characters

data record is employee-record-in.

01 employee-record-in.

  02 employee-name-in   pic x(20) .

  02 employee-rate-in   pic 9(3)v99 .

  02 employee-hours-in   pic 9(3)v99 .

  02 line-feed-in        pic x(1) .



# LISP, Scheme, Common LISP

---

## Συναρτησιακές γλώσσες υψηλού επιπέδου

```
(defun gnome-doc-insert ()
  "Add a documentation header to the current function.
  Only C/C++ function types are properly supported currently."
(interactive)
(let (c-insert-here (point))
  (save-excursion
    (beginning-of-defun)
    (let (c-arglist
          c-funcname
          (c-point (point))
          c-comment-point
          c-isvoid
          c-doinsert)
        (search-backward "(")
        (forward-line -2)
        (while (or (looking-at "^$")
                   (looking-at "^\s+ * \}"))
               (looking-at "^\s+ \\*"))
               (looking-at "^\s+ #")))
      (forward-line 1)))
```



## Γλώσσα αλληλεπίδρασης (interactive) με ισχυρούς τελεστές

```

[0] Z←GAUSSRAND N;B;F;M;P;Q;R
[1] ⍝ Returns ω random numbers having a Gaussian normal distribution
[2] ⍝ (with mean 0 and variance 1) Uses the Box-Muller method.
[3] ⍝ See Numerical Recipes in C, pg. 289.
[4]
[5] Z←10
[6] M←-1+2★31          ⍝ largest integer
[7] L1:Q←N-PZ           ⍝ how many more we need
[8] →(Q≤0)/L2           ⍝ quit if none
[9] Q←⌈1.3×Q÷2          ⍝ approx num points needed
[10] P←-1+(2÷M-1)×-1+?(Q,2)PM ⍝ random points in -1 to 1 square
[11] R++/P×P              ⍝ distance from origin squared
[12] B←(R≠0)∧R<1
[13] R←B/R ⋄ P←B≠P      ⍝ points within unit circle
[14] F←(-2×(⊗R)÷R)★.5
[15] Z←Z,,P×F,[1.5]F
[16] →L1
[17] L2:Z←N+Z
[18] ⍝ ArchDate: 12/1

```



# Algol, Pascal, Clu, Modula, Ada

---

*Προστακτικές γλώσσες με τυπικά ορισμένο συντακτικό, χρήση μπλοκ,  
δομημένος προγραμματισμός*

**PROC insert = (INT e, REF TREE t)VOID:**

```
# NB inserts in t as a side effect #
IF TREE(t) IS NIL THEN t := HEAP NODE := (e, TREE(NIL), TREE(NIL))
ELIF e < e OF t THEN insert(e, l OF t)
ELIF e > e OF t THEN insert(e, r OF t)
FI;
```

**PROC trav = (INT switch, TREE t, SCANNER continue, alternative)VOID:**

```
# traverse the root node and right sub-tree of t only. #
IF t IS NIL THEN continue(switch, alternative)
```

```
ELIF e OF t <= switch THEN
    print(e OF t);
```

```
    traverse( switch, r OF t, continue, alternative)
```

```
ELSE # e OF t > switch #
    PROC defer = (INT sw, SCANNER alt)VOID:
```

```
        trav(sw, t, continue, alt);
```

```
        alternative(e OF t, defer)
```

```
    FI;
```

# SNOBOL, Icon

---

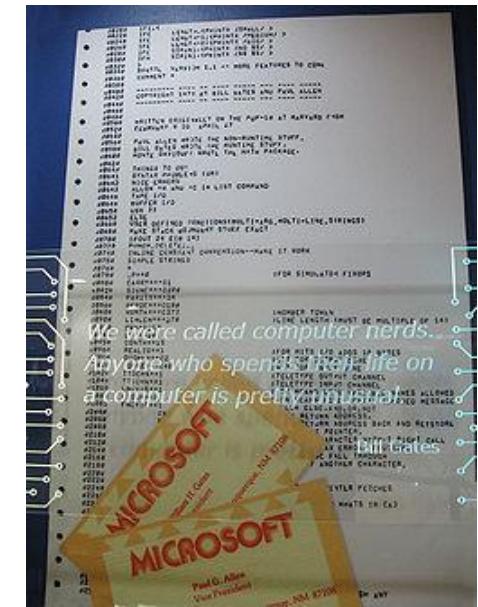
## Γλώσσες επεξεργασίας συμβολοσειρών

```
LETTER = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ$#@'  
SP.CH = "+-,=.*()'/& "  
SCOTA = SP.CH  
SCOTA '&' =  
Q = """  
QLIT = Q FENCE BREAK(Q) Q  
ELEM = QLIT | 'L' Q | ANY(SCOTA) | BREAK(SCOTA) | REM  
F3 = ARBNO(ELEM FENCE)  
B = (SPAN(' ') | RPOS(0)) FENCE  
F1 = BREAK(' ') | REM  
F2 = F1  
CAOP = ('LCL' | 'SET') ANY('ABC') |  
+ 'AIF' | 'AGO' | 'ACTR' | 'ANOP'  
ATTR = ANY('TLSIKN')  
ELEMC = ' (' FENCE *F3C ') ' | ATTR Q | ELEM  
F3C = ARBNO(ELEMC FENCE)  
ASM360 = F1 . NAME B  
+ ( CAOP . OPERATION B F3C . OPERAND |  
+ F2 . OPERATION B F3 . OPERAND)  
+ B REM . COMMENT
```

# BASIC

## Προγραμματισμός για τις “μάζες”

```
10 PRINT "GUESS A NUMBER BETWEEN ONE AND TEN"  
20 INPUT A$  
30 IF A$ = "5" THEN PRINT "GOOD JOB, YOU GUESSED IT"  
40 IF A$ = "5" GOTO 100  
50 PRINT "YOU ARE WRONG. TRY AGAIN"  
60 GOTO 10  
100 END
```



# Simula, Smalltalk, C++, Java, C#

---

Γλώσσες φιλοσοφίας αντικειμενοστρεφούς προγραμματισμού

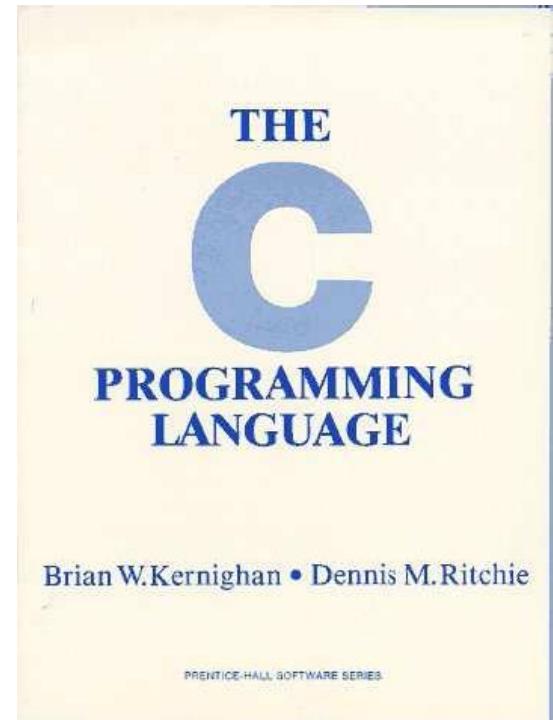
```
class Shape(x, y); integer x; integer y;
virtual: procedure draw;
begin
    comment -- get the x & y coordinates --;
    integer procedure getX;
        getX := x;
    integer procedure getY;
        getY := y;
    comment -- set the x & y coordinates --;
    integer procedure setX(newx); integer newx;
        x := newx;
    integer procedure setY(newy); integer newy;
        y := newy;
end Shape;
```

# C

---

## Ικανοποιητική επίδοση για προγραμματισμό συστήματος

```
int gcd(int a, int b)
{
    while (a != b) {
        if (a > b) a -= b;
        else b -= a;
    }
    return a;
}
```



# ML, Miranda, Haskell, Erlang



```
structure RevStack = struct
  type 'a stack = 'a list
  exception Empty
  val empty = []
  fun isEmpty (s:'a stack):bool =
    (case s
     of [] => true
      | _ => false)
  fun top (s:'a stack) : =
    (case s
     of [] => raise Empty
      | x::xs => x)
  fun pop (s:'a stack) :'a stack =
    (case s
     of [] => raise Empty
      | x::xs => xs)
  fun push (s:'a stack,x: 'a) :'a stack = x::s
  fun rev (s:'a stack) :'a stack = rev (s)
end
```

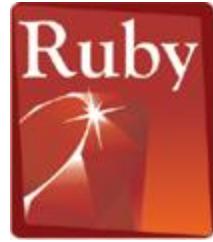


# sh, awk, perl, tcl, javascript, python, ruby



## Γλώσσες σεναρίων (Scripting languages)

```
class() {  
    classname='echo "$1" | sed -n '1 s/ *.*$//p' '  
    parent='echo "$1" | sed -n '1 s/^.*: */p' '  
    hppbody='echo "$1" | sed -n '2,$p' '  
    forwarddefs="$forwarddefs  
    class $classname;"  
    if (echo $hppbody | grep -q "$classname()"); then  
        defaultconstructor=  
    else  
        defaultconstructor="$classname() {}"  
    fi  
}
```



# VisiCalc, Lotus 1-2-3, Excel

---

Γλώσσες προγραμματισμού λογιστικών φύλλων

	A	B	
1	Hours	23	
2	Wage per hour	\$ 5.36	
3			
4	Total Pay	\$ 123.28	← B1 * B2
5			
6			

# SQL

---

## Γλώσσες βάσεων δεδομένων

```
CREATE TABLE shirt (
    id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
    style ENUM('t-shirt', 'polo', 'dress') NOT NULL,
    color ENUM('red', 'blue', 'white', 'black') NOT NULL,
    owner SMALLINT UNSIGNED NOT NULL
        REFERENCES person(id),
    PRIMARY KEY (id)
);

INSERT INTO shirt VALUES
    (NULL, 'polo', 'blue', LAST_INSERT_ID()),
    (NULL, 'dress', 'white', LAST_INSERT_ID()),
    (NULL, 't-shirt', 'blue', LAST_INSERT_ID());
```



```
> SELECT * FROM users WHERE clue > 0
0 rows returned
```

# Prolog, Mercury

---

## Γλώσσες λογικού προγραμματισμού

```
/* palindrome(Xs) is true if Xs is a palindrome. */
/* e.g. palindrome([m,a,d,a,m, i,m, a,d,a,m]). */

palindrome([]).
palindrome([_]).
palindrome([X|Xs]) :-
    append(Xs1, [X], Xs), palindrome(Xs1).

append([], Ys, Ys).
append([X|Xs], Ys, [X|Zs]) :- append(Xs, Ys, Zs).
```

# Νέες γλώσσες προγραμματισμού

---

- “Καθαρότητα” σχεδίασης: δεν υπάρχει η ανάγκη να διατηρηθεί η συμβατότητα με υπάρχοντα προγράμματα
- Όμως πλέον οι νέες γλώσσες δεν είναι προϊόντα παρθενογέννησης: συνήθως χρησιμοποιούν ιδέες από ήδη υπάρχουσες γλώσσες
- Κάποιες από αυτές (λίγες) χρησιμοποιούνται ευρέως, άλλες όχι
- Ανεξάρτητα της χρήσης τους, αποτελούν πηγή ιδεών για τις επόμενες γενεές των γλωσσών προγραμματισμού

# Ευρέως χρησιμοποιούμενη: Java

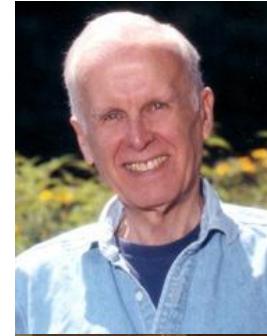


- Αρκετά δημοφιλής από το 1995 και έκτοτε
- Η Java χρησιμοποιεί πολλές ιδέες από τη C++, κάποιες άλλες ιδέες από τη Mesa και τη Modula, την ιδέα της αυτόματης διαχείρισης μνήμης από τη Lisp, και άλλες ιδέες από άλλες γλώσσες
- Η C++ περιλαμβάνει το μεγαλύτερο κομμάτι της C και την επέκτεινε με ιδέες από τις γλώσσες Simula 67, Ada, Clu, ML και Algol 68
- Η C προέκυψε από τη B, που προέκυψε από τη BCPL, που προέκυψε από τη CPL, που προέκυψε από την Algol 60, που προέκυψε από την Algol 58

# Μη ευρέως χρησιμοποιούμενη: Algol

---

- Μια από τις πρώτες γλώσσες: ALGOrithmic Language
- Εκδόσεις: Algol 58, Algol 60, Algol 68
- Ποτέ δε χρησιμοποιήθηκε ευρέως
- 'Όμως εισήγαγε πολλές ιδέες που στη συνέχεια χρησιμοποιήθηκαν από άλλες γλώσσες, όπως για παράδειγμα:
  - Δομή ανά μπλοκ και εμβέλεια μεταβλητών
  - Αναδρομικές συναρτήσεις
  - Πέρασμα παραμέτρων κατά τιμή (parameter passing by value)



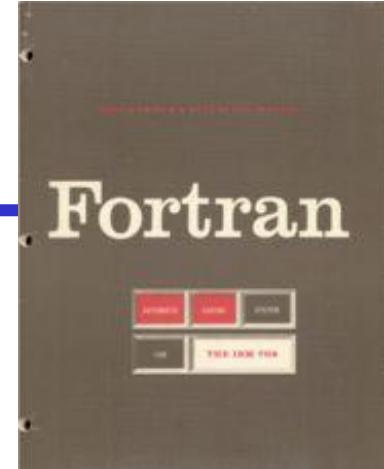
## Διάλεκτοι

---

- Η εμπειρία από τη χρήση γλωσσών αναδεικνύει πιθανές ατέλειες του σχεδιασμού τους και συχνά οδηγεί σε νέες διαλέκτους
- Νέες ιδέες πολλές φορές ενσωματώνονται σε νέες διαλέκτους παλαιών γλωσσών

# Κάποιες διάλεκτοι της Fortran

- Αρχική Fortran, IBM, 1954
- Βασικά standards:
  - Fortran II
  - Fortran III
  - Fortran IV
  - Fortran 66
  - Fortran 77
  - Fortran 90
  - Fortran 95
  - Fortran 2K
- Αποκλίσεις σε κάθε υλοποίηση
- Παράλληλη επεξεργασία
  - HPF
  - Fortran M
  - Vienna Fortran
  - και πολλές άλλες



# Η σχέση των γλωσσών με τον προγραμματισμό

---

- Οι γλώσσες επηρεάζουν τον προγραμματισμό
  - Η κάθε γλώσσα ενθαρρύνει ένα συγκεκριμένο τρόπο προγραμματισμού / αλγορίθμικής επίλυσης προβλημάτων
- Οι εμπειρίες από τον προγραμματισμό εφαρμογών επηρεάζουν το σχεδιασμό (στοιχείων) νέων γλωσσών
- Διαφορετικές γλώσσες ενθαρρύνουν διαφορετικά στυλ προγραμματισμού
  - Αντικειμενοστρεφείς: αντικείμενα και χρήση get/set μεθόδων
  - Συναρτησιακές: πολλές μικρές συναρτήσεις χωρίς παρενέργειες
  - Λογικές: διαδικασία της αναζήτησης σ'ένα λογικά ορισμένο χώρο

# Αντίσταση κατά των γλωσσών;

---

- Γλώσσες που ενθαρρύνουν συγκεκριμένους τρόπους προγραμματισμού συνήθως δεν τους επιβάλλουν πλήρως
- Κατά συνέπεια, είναι δυνατό να παρακάμψουμε ή και να αγνοήσουμε πλήρως τη “φιλοσοφία” κάποιας γλώσσας
- Συνήθως όμως αυτό δεν είναι καλή ιδέα...

# Προστακτική ML

---

Η ML αποθαρρύνει τη χρήση αναθέσεων και παρενεργειών.

Παρόλα αυτά:

```
fun fact n =
  let
    val i = ref 1;
    val xn = ref n
  in
    while !xn > 1 do (
      i := !i * !xn;
      xn := !xn - 1
    );
    !i
  end;
```

# Μη αντικειμενοστρεφής Java

---

Η Java, σε μεγαλύτερο βαθμό από τη C++, ενθαρρύνει τον αντικειμενοστρεφή προγραμματισμό. Παρόλα αυτά:

```
class Fubar {  
    public static void main (String[] args) {  
        // όλο το πρόγραμμα εδώ!  
    }  
}
```

# Συναρτησιακή Pascal

- Κάθε προστακτική γλώσσα που υποστηρίζει αναδρομή, μπορεί να χρησιμοποιηθεί ως συναρτησιακή γλώσσα

```
function ForLoop(Low, High: Integer): Boolean;
begin
  if Low <= High then
    begin
      {όλο το σώμα του for loop εδώ}
      ForLoop := ForLoop(Low+1, High)
    end
  else
    ForLoop := True
end;
```

# Γλώσσες και θεωρία τυπικών γλωσσών

---

Θεωρία των τυπικών γλωσσών: μία από τις θεμελιώδεις μαθηματικές περιοχές της επιστήμης των υπολογιστών

- Κανονικές γραμματικές, αυτόματα πεπερασμένων καταστάσεων
  - Αποτελούν τη βάση για το λεκτικό των γλωσσών προγραμματισμού και του λεκτικού αναλυτή (scanner) ενός compiler
- Γραμματικές ελεύθερες συμφραζόμενων, αυτόματα στοίβας
  - Αποτελούν τη βάση για το συντακτικό των γλωσσών προγραμματισμού και του συντακτικού αναλυτή (parser) ενός compiler
- Μηχανές Turing
  - Προσφέρουν το θεωρητικό υπόβαθρο για να μελετήσουμε την υπολογιστική ισχύ των γλωσσών προγραμματισμού

# Ισοδυναμία κατά Turing (Turing equivalence)

---

- Οι (περισσότερες) γλώσσες προγραμματισμού έχουν διαφορετικά χαρακτηριστικά και πλεονεκτήματα χρήσης, αλλά όλες έχουν την ίδια ισχύ επίλυσης προβλημάτων
  - {προβλήματα επιλύσιμα στη Java}
  - = {προβλήματα επιλύσιμα στη Fortran}
  - = {προβλήματα επιλύσιμα στη C}
  - = ...
- Και όλες έχουν την ίδια ισχύ με διάφορα υπολογιστικά μοντέλα
  - {προβλήματα επιλύσιμα σε μηχανές Turing}
  - = {προβλήματα επιλύσιμα σε λάμβδα λογισμό}
  - = ...
- Το παραπάνω είναι γνωστό ως η θέση Church-Turing



# Συμπερασματικά

---

- Γιατί είναι ενδιαφέρουσες οι γλώσσες προγραμματισμού (και αυτό το μάθημα):
  - Λόγω της ποικιλίας τους και των χαρακτηριστικών τους
  - Λόγω των αμφιλεγόμενων στοιχείων τους
  - Λόγω της ενδιαφέρουσας εξέλιξής τους
  - Λόγω της στενής τους σχέσης με τον προγραμματισμό και την ανάπτυξη λογισμικού
  - Λόγω του θεωρητικού τους υπόβαθρου και της στενής τους σχέσης με την επιστήμη των υπολογιστών
- Επίσης, λόγω του ότι θα μάθετε αρκετά καλά τρεις (+) επιπλέον γλώσσες!