Κεφάλαιο 6 - Εισαγωγή στον Προγραμματισμό

6.1 Η έννοια του προγράμματος

Η **επίλυση** ενός προβλήματος με τον υπολογιστή περιλαμβάνει, **τρία σημαντικά στάδια**:

1. Τον ακριβή προσδιορισμό του προβλήματος.
2. Την ανάπτυξη του αντίστοιχου αλγορίθμου.
3. Τη διατύπωση του αλγορίθμου σε κατανοητή μορφή από τον υπολογιστή.

Το τρίτο ισοδυναμεί με τον προγραμματισμό και το πρόγραμμα: σύνολο των εντολών που πρέπει να δοθούν στον υπολογιστή, ώστε να υλοποιηθεί ο αλγόριθμος για την επίλυση του προβλήματος, γραμμένες σε κάποια γλώσσα προγραμματισμού με βασικά στοιχεία, τα δεδομένα και τις δομές δεδομένων επί των οποίων ενεργεί.

Οι γλώσσες προγραμματισμού αναπτύχθηκαν με σκοπό την επικοινωνία ανθρώπου - υπολογιστή

Στοιχειώδεις ενέργειες του υπολογιστή:

απόθηκευση, ανάκτηση, αριθμητικές πράξεις, σύγκριση δυαδικών ψηφίων

6.3 Φυσικές και τεχνητές γλώσσες

Μια γλώσσα γενικά προσδιορίζεται από:

1. Το αλφάβητο: το σύνολο των στοιχείων που χρησιμοποιεί (γράμματα πεζά και κεφαλαία, ψηφία, σημεία στίξης)

2. Το λεξιλόγιο: τις λέξεις που είναι δεκτές από τη γλώσσα.

3. Τη γραμματική: η οποία αποτελείται από:

α) Το **τυπικό ή τυπολογικό (*accidence*)**: το σύνολο των κανόνων που ορίζουν τις μορφές με τις οποίες είναι αποδεκτή μία λέξη (π.χ. κλίσεις, βαθμοί)

β) Το **συντακτικό (*syntax*)**: το σύνολο των κανόνων για τη δημιουργία προτάσεων.

4. Τη σημασιολογία (*Semantics*): το σύνολο των κανόνων που καθορίζει το νόημα των λέξεων, των εκφράσεων και των προτάσεων.

Διαφορές μεταξύ φυσικών και τεχνητών γλωσσών

Σχετικά με τη δυνατότητα εξέλιξής τους:

- Οι φυσικές γλώσσες εξελίσσονται συνεχώς γιατί χρησιμοποιούνται καθημερινά για την ανθρώπινη επικοινωνία.

- Οι τεχνητές γλώσσες χαρακτηρίζονται από στασιμότητα, αφού κατασκευάζονται για ένα συγκεκριμένο σκοπό. Ωστόσο βελτιώνονται για να διορθωθούν οι αδυναμίες τους ή για να ακολουθήσουν τις εξελίξεις.

6.4 Τεχνικές σχεδίασης προγραμμάτων

6.4.1 Ιεραρχική σχεδίαση προγράμματος (*hierarchical programming*)

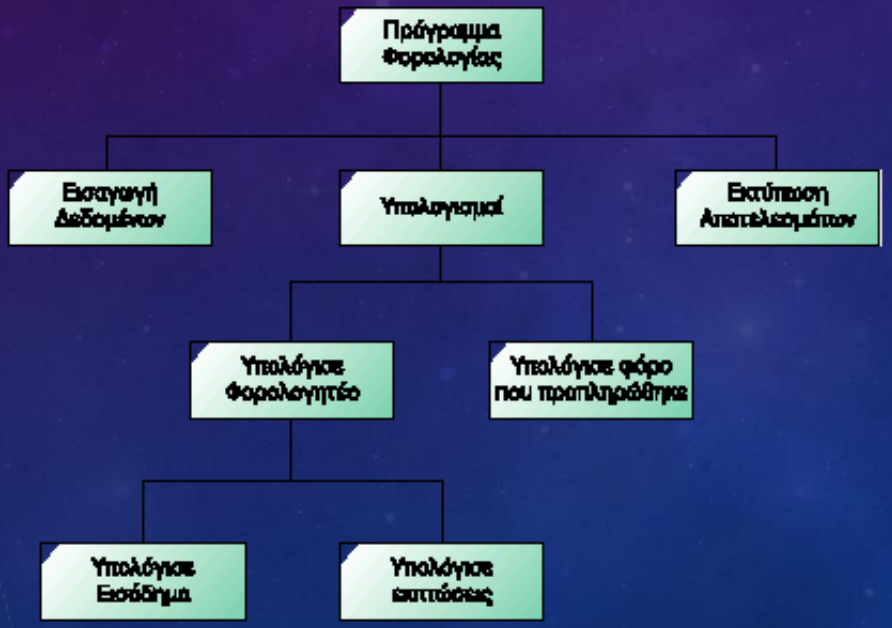
Ιεραρχική σχεδίαση : Είναι διαδικασία σχεδίασης "από επάνω προς τα κάτω" ή (*top-down program design*) που περιλαμβάνει:

α) τον καθορισμό των βασικών λειτουργιών ενός προγράμματος, σε ανώτερο επίπεδο και

β) τη διάσπαση του προβλήματος σε μια σειρά από απλούστερα υποπροβλήματα, τα οποία είναι εύκολο να επιλυθούν οδηγώντας στην επίλυση του αρχικού προβλήματος.

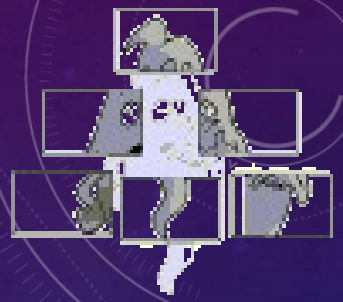
Η ιεραρχική σχεδίαση χρησιμοποιεί τις *διαγραμματικές τεχνικές*.

Παράδειγμα : *Ιεραρχική σχεδίαση υπολογισμού του φόρου εισοδήματος*



6.4.2 Τμηματικός προγραμματισμός (*modular programming*)

Υλοποίηση : μετά την ανάλυση του προβλήματος σε υποπροβλήματα, κάθε υποπρόβλημα αποτελεί ανεξάρτητη ενότητα (*module*), που γράφεται ξεχωριστά.



Πλεονεκτήματα: διευκολύνει τη δημιουργία του προγράμματος, μειώνει τα λάθη και επιτρέπει την ευκολότερη παρακολούθηση, κατανόηση και συντήρηση του προγράμματος από τρίτους.

6.4.3 Δομημένος προγραμματισμός (*structural programming*)

Παρουσιάστηκε στα μέσα του 1960 με σκοπό τη μείωση των εντολών GOTO. Η εντολή GOTO έχει ως αποτέλεσμα την αλλαγή της ροής του προγράμματος, της διακλάδωσης σε μία άλλη εντολή του προγράμματος εκτός από την επόμενη. Π.χ:

```
... ! μη δομημένα
ΑΝ Αριθμός>0 ΤΟΤΕ
  GOTO 1
ΤΕΛΟΣ_ΑΝ
ΑΝ Αριθμός=0 ΤΟΤΕ
  GOTO 2
ΤΕΛΟΣ_ΑΝ
ΓΡΑΨΕ ‘Αρνητικός’
GOTO 4
1:ΓΡΑΨΕ ‘Θετικός’
GOTO 4
2: ΓΡΑΨΕ ‘Μηδέν’
4: ΓΡΑΨΕ ‘Τέλος’
```

ή

```
... ! δομημένα
ΑΝ Αριθμός>0 ΤΟΤΕ
    ΓΡΑΨΕ ‘Θετικός’
ΑΛΛΙΩΣ_ΑΝ Αριθμός=0 ΤΟΤΕ
    ΓΡΑΨΕ ‘Μηδέν’
ΑΛΛΙΩΣ
    ΓΡΑΨΕ ‘Αρνητικός’
ΤΕΛΟΣ_ΑΝ
ΓΡΑΨΕ ‘Τέλος’
! (πιο εύκολο στην κατανόηση)
```

6.4.3 Δομημένος προγραμματισμός (*structural programming*)

Είναι μία μεθοδολογία σύνταξης προγραμμάτων που χρησιμοποιεί τη δομή της ακολουθίας, τη δομή της επιλογής και τη δομή της επανάληψης μόνες και σε συνδυασμό τους. Συνδυάζει την ιεραρχική σχεδίαση και τον τμηματικό προγραμματισμό. Κάθε πρόγραμμα όπως και κάθε ενότητα προγράμματος έχει μόνο μία είσοδο και μόνο μία έξοδο.

Πλεονεκτήματα του δομημένου προγραμματισμού

1. Δημιουργία απλούστερων προγραμμάτων.
2. Άμεση μεταφορά των αλγορίθμων σε προγράμματα.
3. Διευκόλυνση ανάλυσης του προγράμματος σε τμήματα.
4. Περιορισμός των λαθών κατά την ανάπτυξη του προγράμματος.
5. Διευκόλυνση στην ανάγνωση και κατανόηση του προγράμματος από τρίτους.
6. Ευκολότερη διόρθωση και συντήρηση.

6.5 Αντικειμενοστραφής προγραμματισμός (*object-oriented*)

Βασική ιδέα: ένα πρόγραμμα περιγράφει «ενέργειες» (επεξεργασία) που εφαρμόζονται πάνω σε δεδομένα. Η αντικειμενοστραφής σχεδίαση εκλαμβάνει ως πρωτεύοντα δομικά στοιχεία ενός προγράμματος τα δεδομένα, από τα οποία δημιουργούνται με κατάλληλη μορφοποίηση τα **αντικείμενα** (*objects*) που εμπεριέχουν και τις εφαρμόσιμες επάνω στα δεδομένα ενέργειες.

Πλεονέκτημα: προγράμματα περισσότερο ευέλικτα και επαναχρησιμοποιήσιμα.

Χρησιμοποιεί την ιεραρχική σχεδίαση, τον τμηματικό προγραμματισμό και ακολουθεί τις αρχές του δομημένου προγραμματισμού.

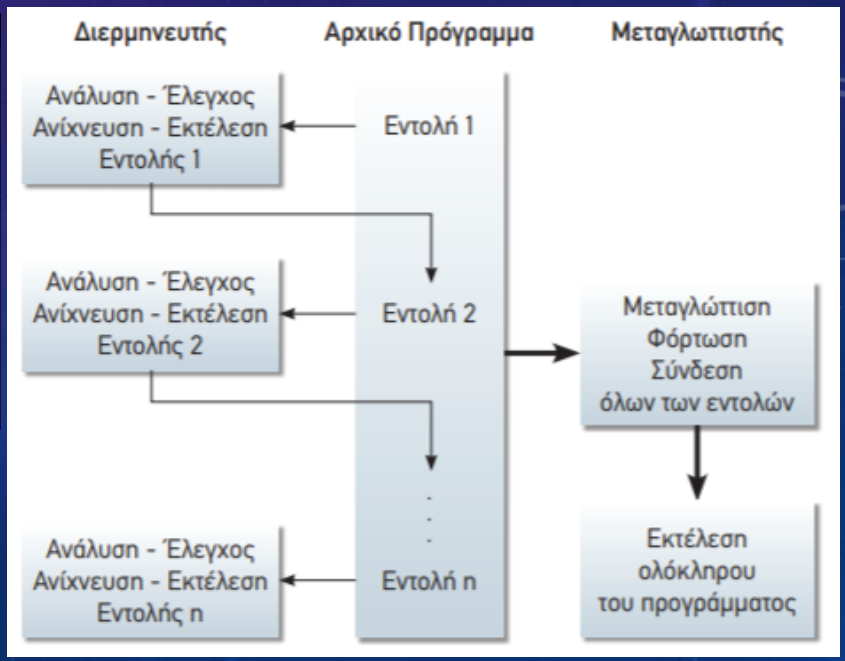
6.7 Προγραμματιστικά περιβάλλοντα

Στόχος: κάθε πρόγραμμα να μετατραπεί σε μορφή αναγνώσιμη και εκτελέσιμη από τον υπολογιστή, δηλαδή σε εντολές γλώσσας μηχανής.

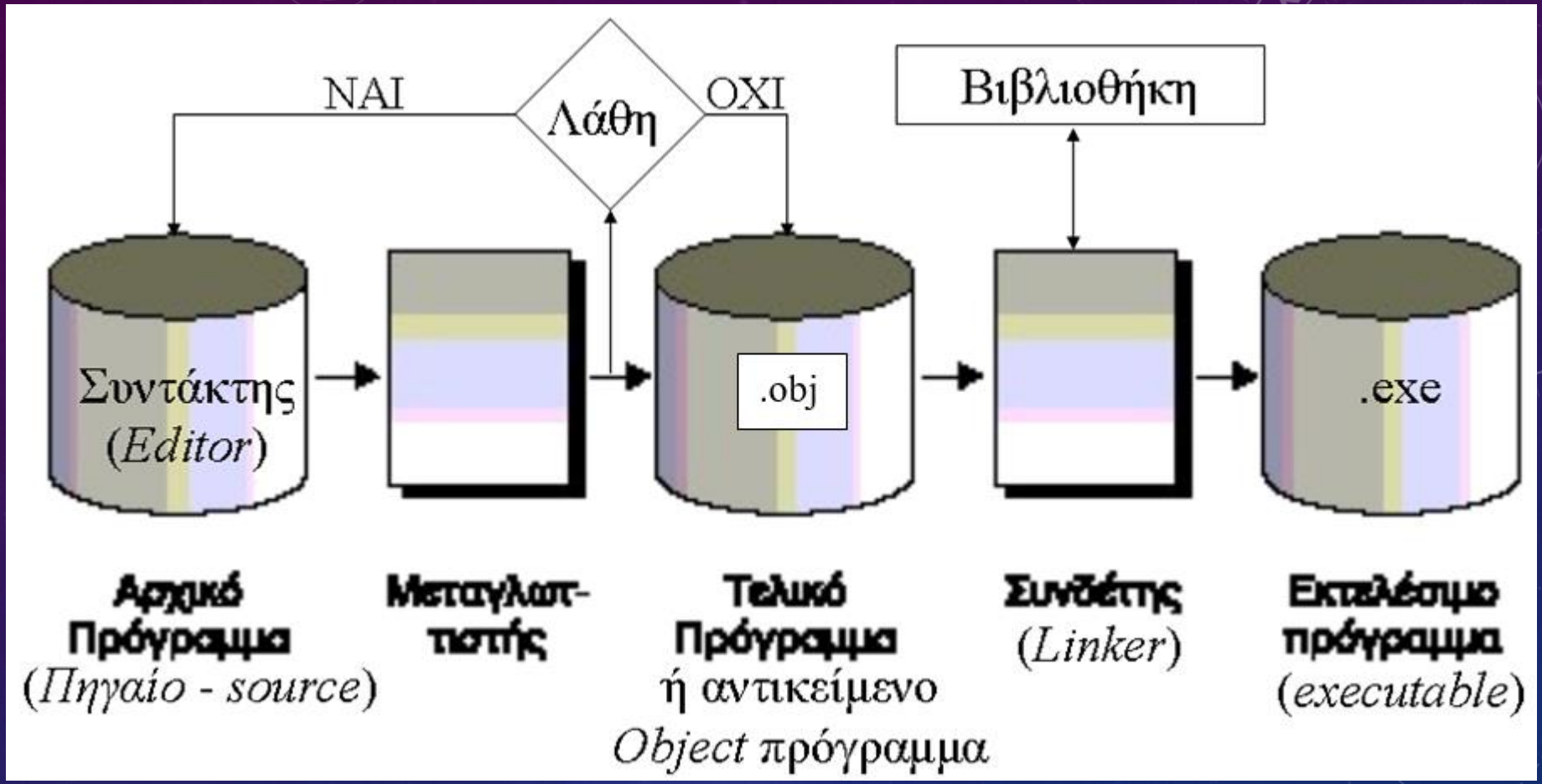
Μεταφραστικά προγράμματα:

1. **Μεταγλωττιστές (compilers):** δέχονται στην είσοδο ένα πρόγραμμα γραμμένο σε μία γλώσσα υψηλού επιπέδου και παράγει ένα ισοδύναμο πρόγραμμα σε γλώσσα μηχανής. (-): υποχρεωτική μεταγλώττιση και σύνδεση πριν την εκτέλεση (+): ταχύτερη εκτέλεση

2. **Διερμηνευτές (interpreters):** διαβάζουν μία προς μία τις εντολές του αρχικού προγράμματος και για κάθε μία εκτελεί αμέσως μία ισοδύναμη ακολουθία εντολών μηχανής. (+): άμεση εκτέλεση και διόρθωση (-) : πιο αργή εκτέλεση



Μεταγλώττιση και σύνδεση προγράμματος:



Κάθε προγραμματιστικό περιβάλλον παρέχει με ενιαίο τρόπο: έναν συντάκτη, έναν μεταγλωττιστή και έναν συνδέτη.

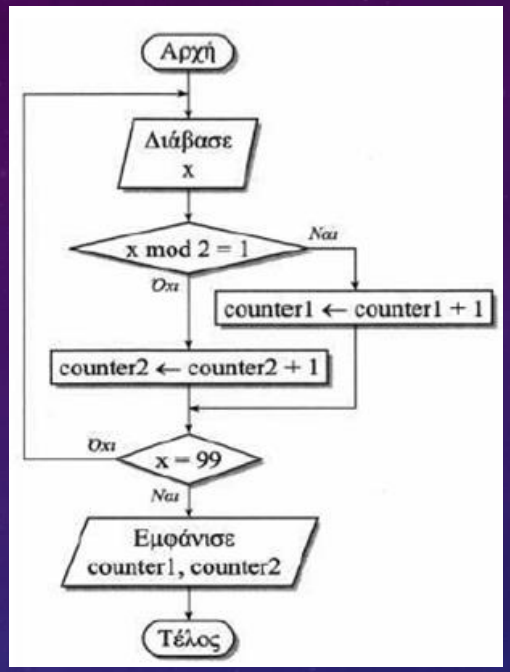
Επιπλέον, ένα περιβάλλον οπτικού (visual) προγραμματισμού διευκολύνει τη δημιουργία γραφικών αντικειμένων (π.χ. φόρμες, λίστες, παράθυρα διαλόγου, μενού, κουμπιά κλπ.)

Μεταγλώττιση και σύνδεση προγράμματος:

ΑρχήΕπανάληψης
  ΑρχήΕπανάληψης
    ΑρχήΕπανάληψης
      Συντάκτης: πηγαίος κώδικας
      Μεταγλωττιστής
    ΜέχριςΌτου (συντακτικά λάθη = 0)
    Αντικείμενο πρόγραμμα
    Συνδέτης
    Εκτελέσιμο - έλεγχος
  ΜέχριςΌτου (λογικά ή κατά την εκτέλεση λάθη = 0)
  Εκτελέσιμο - χρήση
ΜέχριςΌτου (λογικά ή κατά την εκτέλεση λάθη = 0)

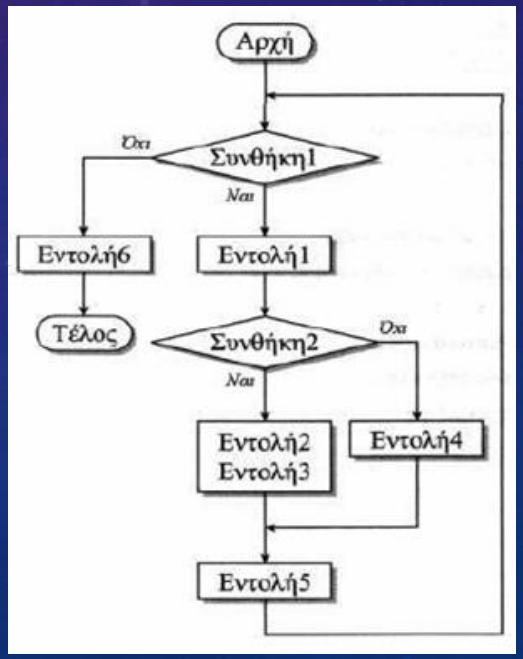
Μετατροπή μη δομημένου κώδικα (με goto) σε δομημένο (χωρίς goto) μέσω ΔΡ

```
Αλγόριθμος ΜηΔομημένο1
    ΣημείοΑ: Διάβασε x
    Αν χ mod 2 = 1 τότε
        counter1 ← counter1 + 1
    Αλλιώς
        counter2 ← counter2 + 1
    Τέλος_αν
    Αν χ = 99 τότε
        Πήγαινε στο ΣημείοΒ
    Αλλιώς
        Πήγαινε στο ΣημείοΑ
    Τέλος_αν
    ΣημείοΒ: Εμφάνισε counter1, counter2
Τέλος ΜηΔομημένοΙ
```




```
Αλγόριθμος Δομημένο1
    Αρχή_Επανάληψης
        Διάβασε χ
        Αν χ mod 2 = 1 τότε
            counter1 ← counter1 + 1
        αλλιώς
            counter2 ← counter2 + 1
        Τέλος_αν
    Μέχρις_ότου x = 99
    Εμφάνισε counter1, counter2
Τέλος Δομημένο 1
```

```
Αλγόριθμος ΜηΔομημένο2
    ΣημείοΑ: Αν Συνθήκη1 τότε
        Εντολή 1
        Αν Συνθήκη2 τότε
            Εντολή2
            Εντολή3
            Πήγαινε στο ΣημείοΒ
        Τέλος_αν
        Εντολή4
        ΣημείοΒ: Εντολή5
        Πήγαινε στο ΣημείοΑ
    Τέλος_αν
    Εντολή6
Τέλος ΜηΔομημένο2
```



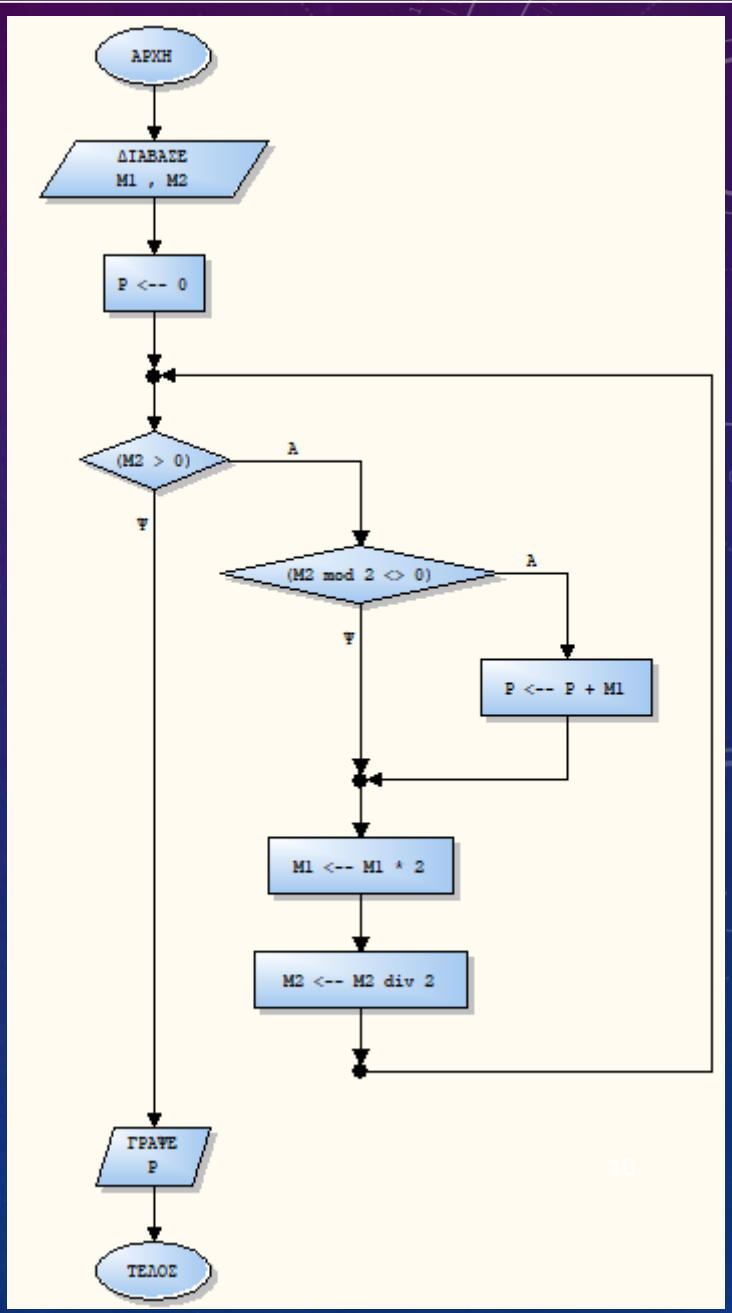

```
Αλγόριθμος Δομημένο2
    Όσο Συνθήκη1 επανάλαβε
        Εντολή1
        Αν Συνθήκη2 τότε
            Εντολή2
            Εντολή3
        Αλλιώς
            Εντολή4
        Τέλος_αν
        Εντολή5
    Τέλος_επανάληψης
    Εντολή6
Τέλος Δομημένο2
```

Καραμαούνας Πολύκαρπος

Μετατροπή μη δομημένου κώδικα (με goto) σε δομημένο (χωρίς goto) μέσω ΔΡ

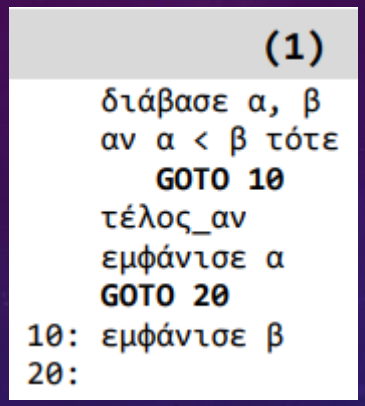
Αλγόριθμος: Πολλαπλασιασμός δύο θετικών ακεραίων (αλά ρωσικά)	
Είσοδος: Έξοδος:	Δύο ακέραιοι M1 και M2, όπου M1, M2 ≥ 1 Το γινόμενο P=M1*M2
Βήμα 1	Θέσε P=0
Βήμα 2	Αν M2>0, τότε πήγαινε στο Βήμα 3, αλλιώς πήγαινε στο Βήμα 7
Βήμα 3	Αν ο M2 είναι περιττός, τότε θέσε P=P+M1
Βήμα 4	Θέσε M1=M1*2
Βήμα 5	Θέσε M2=M2/2 (θεώρησε μόνο το ακέραιο μέρος)
Βήμα 6	Πήγαινε στο Βήμα 2
Βήμα 7	Τύπωσε τον P.

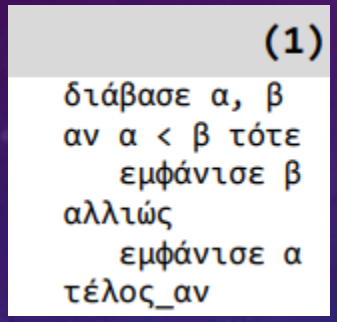
```
ΠΡΟΓΡΑΜΜΑ  Πολλαπλασιασμός_αλά_ρωσικά
ΜΕΤΑΒΛΗΤΕΣ
  ΑΚΕΡΑΙΕΣ: M1, M2, P
ΑΡΧΗ
  ΔΙΑΒΑΣΕ M1, M2
  P <-- 0
  Όσο (M2 > 0) επανάλαβε
    Αν (M2 mod 2 <> 0)  ΤΟΤΕ
      P <-- P + M1
    ΤΕΛΟΣ_ΑΝ
    M1 <--  M1 * 2
    M2 <--  M2 div 2
  ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
  Γράψε P
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

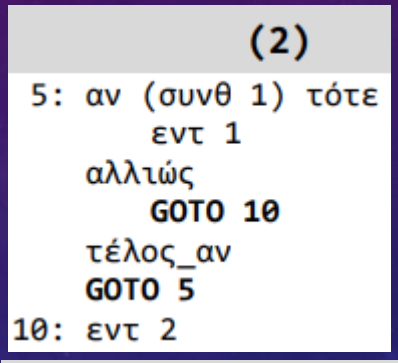


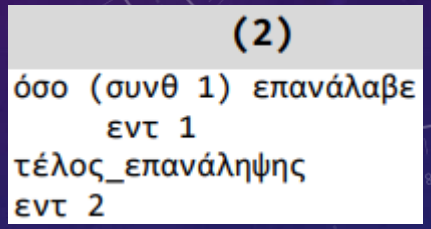
Μετατροπή μη δομημένου κώδικα (με goto) σε δομημένο (χωρίς goto) μέσω ΔΡ

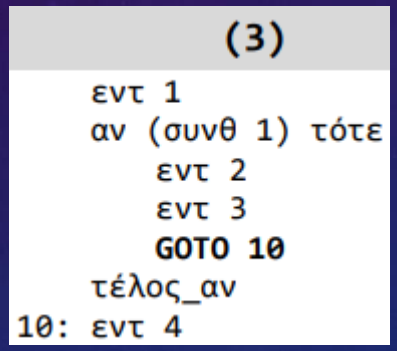
http://www.zioulas.gr

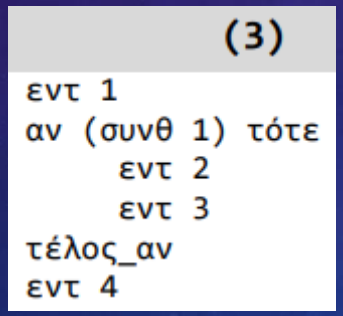


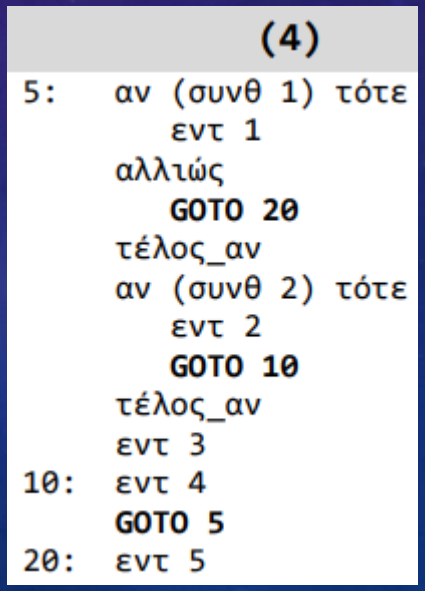


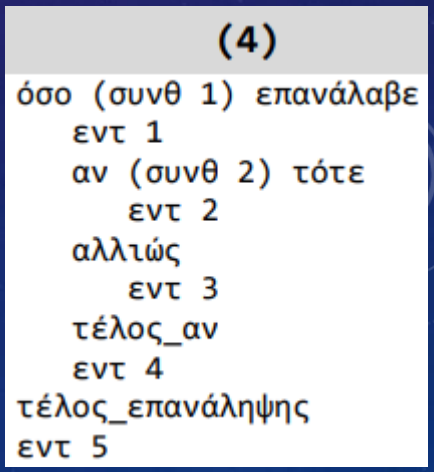












Μετατροπή μη δομημένου κώδικα (με goto) σε δομημένο (χωρίς goto) μέσω ΔΡ

http://www.zioulas.gr

(5)

```
5:   αν (συνθ 1) τότε
        εντ 1
10:     εντ 2
        αν (συνθ 2) τότε
           εντ 3
           GOTO 10
        τέλος_αν
        GOTO 5
     αλλιώς
        εντ 4
     τέλος_αν
```

⇔

(5)

```
όσο (συνθ 1) επανάλαβε
   εντ 1
   εντ 2
   όσο (συνθ 2) επανάλαβε
      εντ 3
      εντ 2
   τέλος_επανάληψης
τέλος_επανάληψης
εντ 4
```

(6)

```
5: αν (συνθ 1) τότε
      εντ 1
      αν (συνθ 2) τότε
         εντ 2
         εντ 3
         GOTO 10
      τέλος_αν
      εντ 4
10:   εντ 5
      GOTO 5
   τέλος_αν
   εντ 6
```

⇔

```
Όσο (συνθ1) επανάλαβε
  εντ 1
  αν (συνθ2) τότε
    εντ 2
    εντ 3
  αλλιώς
    εντ 4
  τέλος_αν
  εντ 5
τέλος_επανάληψης
εντ 6
```

(7)

```
   Διάβασε F
   C ← 5/9*(F-32)
   Αν C < 10 τότε
      GOTO 10
   Αν C > 25 τότε
      GOTO 20
   Αν C ≤ 25 τότε
      GOTO 30
10: Εκτύπωσε 'Χαμηλή'
   GOTO 40
20: Εκτύπωσε 'Υψηλή'
   GOTO 40
30: Εκτύπωσε 'Μέτρια'
40: !συνέχεια
```

⇔

(7)

```
Διάβασε F
C ← 5/9 * (F-32)
Αν C < 10 τότε
   Εκτύπωσε 'Χαμηλή'
αλλιώς_αν C > 25 τότε
   Εκτύπωσε 'Υψηλή'
αλλιώς
   Εκτύπωσε 'Μέτρια'
τέλος_αν
```

(8)

```
   ι ← 1
10: Διάβασε αποδ
   κρατ ← 0,2 * αποδ
   πληρ ← αποδ – κρατ
   Εκτύπωσε ι, αποδ, πληρ
   Αν ι = 5 τότε
      GOTO 20
   αλλιώς
      ι ← ι + 1
      GOTO 10
   τέλος_αν
20: !συνέχεια
```

⇔

(8)

```
Για ι από 1 μέχρι 5
   Διάβασε αποδ
   κρατ ← 0,2 * αποδ
   πληρ ← αποδ – κρατ
   Εκτύπωσε ι, αποδ, πληρ
Τέλος_επανάληψης
```

(με τη διαφορά ότι εδώ το ι καταλήγει με την τιμή 6)

(9)

```
5: Διάβασε α
  Αν (α mod 2 = 0) τότε
     Εμφάνισε  α
     GOTO 5
  Τέλος_Αν
```

⇔

(9)

```
Διάβασε α
Όσο α mod 2 = 0 επανάλαβε
    Εμφάνισε  α
    Διάβασε α
Τέλος_Επανάληψης
```

```
Όσο (Σ) επανάλαβε
        εντ
        goto 1
ΤέλοςΕπανάληψης
1:
```

⇔

```
Αν (Σ) τότε
        εντ
ΤέλοςΑν
```

(11)

```
Όσο συνθ1 επανάλαβε
   Εντ1
   Αν συνθ2 τότε
      Εντ3
   Αλλιώς
      Εντ2
      GOTO 10
   Τέλος_Αν
Τέλος_Επανάληψης
10:
```

⇕

(10)

```
      counter1 ← 0
      counter2 ← 0
10:   Διάβασε x
      Αν (x mod 2 = 1) τότε
         counter1 ← counter1 + 1
      αλλιώς
         counter2 ← counter2 + 1
      Τέλος_αν
      Αν (x = 99) τότε
         GOTO 20
      Αλλιώς
         GOTO 10
      Τέλος_αν
20:
```

⇔

(10)

```
counter1 ← 0
counter2 ← 0
Αρχή_Επανάληψης
   Διάβασε x
   Αν (x mod 2 = 1) τότε
      counter1 ← counter1 + 1
   αλλιώς
      counter2 ← counter2 + 1
   Τέλος_αν
Μέχρις_Ότου (x = 99)
```

(11)

```
flag ← ψευδής
Όσο συνθ1 και flag = ψευδής επανάλαβε
      Εντ1
      Αν συνθ2 τότε
         Εντ3
      Αλλιώς
         Εντ2
         flag ← αληθής
      Τέλος_Αν
Τέλος_Επανάληψης
```

(12)

```
      Sum ← 0
      ι ← 1
10:   Sum ← Sum + A[ι]
      ι ← ι + 1
      Αν ι > 100 τότε
         GOTO 30
      Τέλος_αν
      GOTO 10
30:   Εμφάνισε Sum
```

⇔

(12)

```
Sum ← 0
Για ι από 1 μέχρι 100
      Sum ← Sum + A[ι]
Τέλος_επανάληψης
Εμφάνισε Sum
```

(13)

```
      K ← 1
      S ← 0
20:   Αν K > 100 TOTE
         GOTO 40
      Τέλος_αν
30:   Διάβασε X
      Αν X <= 0 τότε
         GOTO 30
      Τέλος_αν
      S <- S + X
      K ← K + 1
      GOTO 20
40:   Εμφάνισε S
```

⇔

(13)

```
S ← 0
Για K από 1 μέχρι 100
   Αρχή_επανάληψης
      Διάβασε X
   Μέχρις_ότου X > 0
   S <- S + X
Τέλος_επανάληψης
Εμφάνισε S
```

Τμήμα 1

Βήμα 1	Θέσε Sum = 0
Βήμα 2	Θέσε i = 1
Βήμα 3	Αν το i <= 10 τότε **πήγαινε στο Βήμα 4**, αλλιώς **πήγαινε στο Βήμα 8**
Βήμα 4	Διάβασε X
Βήμα 5	Θέσε Sum = Sum + X
Βήμα 6	Θέσε i = i + 1
Βήμα 7	**Πήγαινε στο Βήμα 3**
Βήμα 8	Τύπωσε το Sum

⇔

```
Sum ← 0
Για ι από 1 μέχρι 10
    Διάβασε X
    Sum ← Sum + X
Τέλος_επανάληψης
Εκτύπωσε Sum
```

Τμήμα 2

Βήμα 1	Διάβασε το κ
Βήμα 2	Θέσε το S = 0
Βήμα 3	Αν κ > 0 τότε **πήγαινε στο Βήμα 4**, αλλιώς **πήγαινε στο Βήμα 8**
Βήμα 4	Μείωσε το κ κατά 1
Βήμα 5	Εμφάνισε το τετράγωνο του κ
Βήμα 6	Αύξησε το S κατά 1
Βήμα 7	**Πήγαινε στο Βήμα 3**
Βήμα 8	Εμφάνισε το S

⇔

```
Διάβασε κ
S ← 0
Όσο κ > 0 επανάλαβε
    κ ← κ - 1
    Εμφάνισε κ^2
    S ← S + 1
Τέλος_Επανάληψης
Εμφάνισε S
```

Τμήμα 3

Βήμα 1	Θέσε το κ = 0
Βήμα 2	Διάβασε το X
Βήμα 3	Αύξησε το κ κατά X
Βήμα 4	Αν το X <= 0 ή το κ < 50, τότε **πήγαινε το Βήμα 2**
Βήμα 5	Εμφάνισε το κ και το X

⇔

```
κ ← 0
Αρχή_επανάληψης
    Διάβασε X
    κ ← κ + X
Μέχρις_ότου X > 0 και κ >= 50
Εμφάνισε κ,X
```

Τμήμα 4

Βήμα 1	Θέσε το X = 0
Βήμα 2	Θέσε το Λ = 0
Βήμα 3	Αύξησε το Λ κατά X
Βήμα 4	Διάβασε το X
Βήμα 5	Αν Λ > 100 τότε **πήγαινε στο Βήμα 7**
Βήμα 6	**Πήγαινε στο Βήμα 3**
Βήμα 7	Εμφάνισε το Λ

⇔

```
X ← 0
Λ ← 0
Αρχή_επανάληψης
    Λ ← Λ + X
    Διάβασε X
Μέχρις_ότου Λ > 100
Εμφάνισε Λ
```