# Pascal

## Μια Μεθοδική Προσέγγιση

Τάσος Λαδιάς


Τεχνικές Δομημένου Προγραμματισμού
Μέσα από Λυμένα
Παραδείγματα

# Περιεχόμενα

# Περιεχόμενα

# Περιεχόμενα

# Περιεχόμενα

## Η αναγκαιότητα των εντολων επανάληψης.

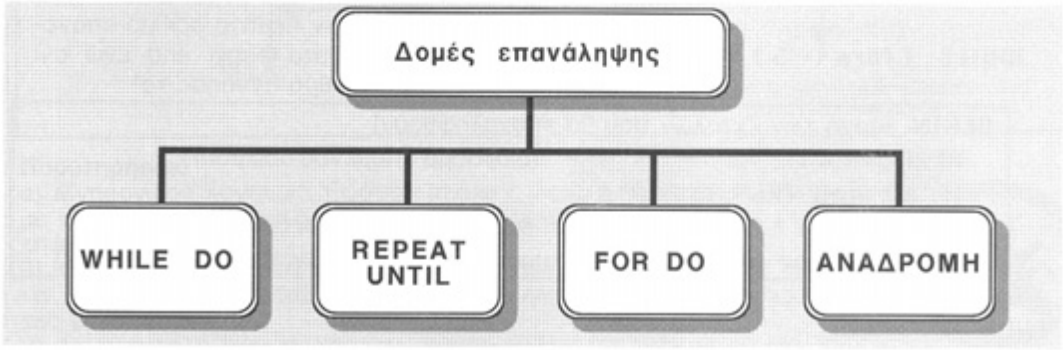
**Το πρόβλημα**
Να βρεθεί ο μεγαλύτερος από πέντε ακέραιους αριθμούς που δίνονται από το πληκτρολόγιο.
**Ο αλγόριθμος**
Το πρόγραμμα EuresnMaximum_3 που έχουμε δεί προηγουμένως βρίσκει το μέγιστο μεταξύ δύο αριθμών. Το πρόγραμμα EuresnMaxTriwv βρίσκει το μέγιστο μεταξύ τριών αριθμών χρησιμοποιώντας δύο φορές τον αλγόριθμο του προγράμματος EuresnMaximum_3 (μια φορά γιά να βρούμε το μεγαλύτερο από τους δύο πρώτους αριθμούς και την επόμενη για τον προηγηθέντα μεγαλύτερο και τον τρίτο αριθμό). Ας στηριχθούμε σ' αυτή την σκέψη και ας την εφαρμόσουμε γιά να βρούμε το μέγιστο πέντε αριθμών.
**Το πρόγραμμα**

```
PROGRAM EuresnMaximumPevte_1 ;
    VAR a, max : INTEGER ;
    BEGIN
      READLN ( a ) ;  {διάβασμα πρώτου αριθμού}
      max := a ; { καταχωρείται ο μοναδικός αριθμός ως ο μέγιστος }
      READLN ( a ) ;                    {διάβασμα δεύτερου αριθμού}
      IF a > max THEN max := a ;
      READLN ( a ) ;                      {διάβασμα τρίτου αριθμού}
      IF a > max THEN max := a ;
      READLN ( a ) ;                    {διάβασμα τέταρτου αριθμού}
      IF a > max THEN max := a ;
      READLN ( a ) ;                     {διάβασμα πέμπτου αριθμού}
      IF a > max THEN max := a ;
      WRITELN ( max ) ;
    END .
```

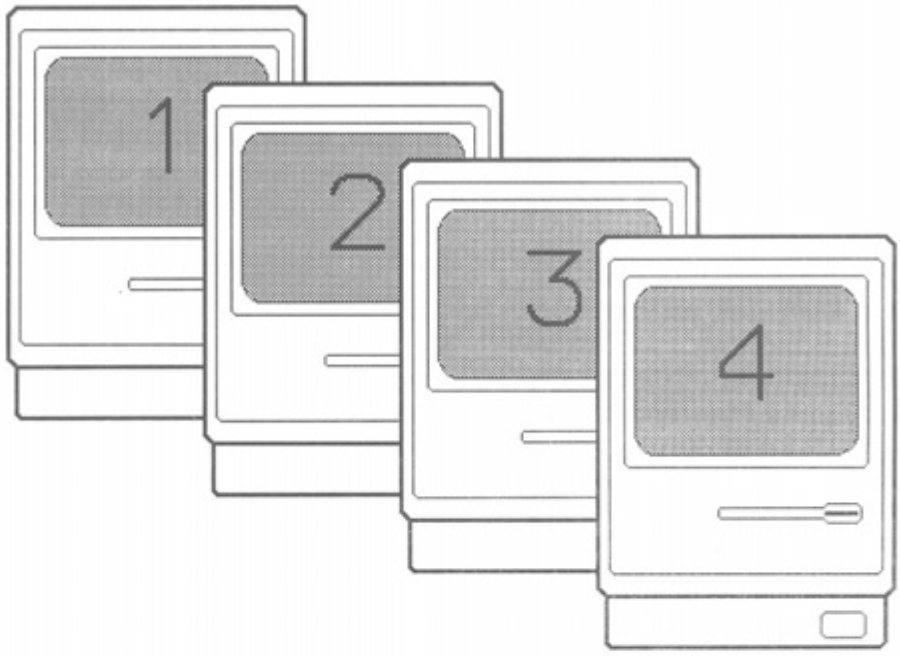
Αν το πρόβλημα μας ζήταγε να βρούμε το μέγιστο από 2000 αριθμούς θα γράφαμε ένα πρόγραμμα με 4000 εντολές περίπου. Παρατηρούμε ότι οι εντολές

```
READLN ( a ) ;
IF a > max THEN max := a ;
```

επαναλαμβάνονται αρκετές φορές. Βλέπουμε λοιπόν να αναδύεται η ανάγκη να μπορούμε να επαναλαμβάνουμε ένα σύνολο εντολών πολλές φορές. Η pascal διαθέτει τρείς εντολές το while, το repeat και το for και μια δυνατότητα τεχνικής προγραμματισμού, την αναδρομή.



## Η αναγκαιότητα των εντολών επανάληψης.




Ας δούμε λοιπόν πως μπορεί να τροποποιηθεί το προηγούμενο πρόγραμμα χρησιμοποιώντας μιά από τις εντολές επανάληψης π.χ. την while.

Το πρόγραμμα

```
PROGRAM EuresnMaximumPeute_2 ;
 VAR a, max, fora : INTEGER ;
 BEGIN
   READLN ( a ) ;  {διάβασμα πρώτου αριθμού}
   max := a ; { καταχωρείται ο μοναδικός αριθμός ως ο μέγιστος }
   fora := 1 ; { αρίθμηση του διαβάσματος αριθμού από πληκτρολόγιο }
                          {εφόσον δεν διάβασες και τον πέμπτο αριθμό επανέ-
   WHILE  ( fora <= 5 )  DO λαβε ό,τι περιέχεται μέσα στα begin end ενώ αν
                          διάβασες και τον πέμπτο αριθμό αγνόησε τα}
     BEGIN {αρχή των εντολών που θα επαναληφθούν}
       READLN ( a ) ;              {διάβασμα επόμενου αριθμού}
       IF a > max THEN max := a ;
       fora := fora + 1 ;  {σημείωσε ότι διάβασες ένα ακόμα αριθμό}
     END ; {τέλος των εντολών που θα επαναληφθούν και ξαναπήγαινε στο while}
   WRITELN ( max ) ;
 END .
```

Ας δούμε στη συνέχεια πως συντάσσεται η εντολή while.

# Η εντολή επανάληψης WHILE.

Ο ψευδοκώδικας, η σύνταξη και το λογικό διάγραμμα της εντολής while.

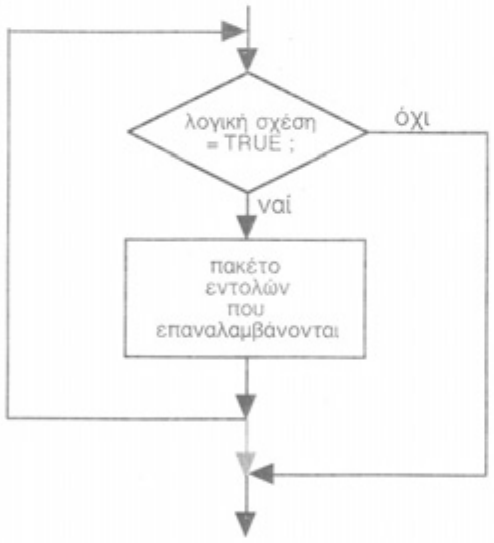
**Ψευδοκώδικας**

```
. . . . . .
Εφόσον η λογική σχέση είναι αληθής
εκτέλεσε το παρακάτω πακέτο εντολών
ή αλλιώς παρέκαμψέ το.

   Αρχή πακέτου εντολών.
     Εντολή 1.
     Εντολή 2.
     . . . . . .
     Εντολή ν.
   Τέλος πακέτου εντολών.

. . . . . .
```

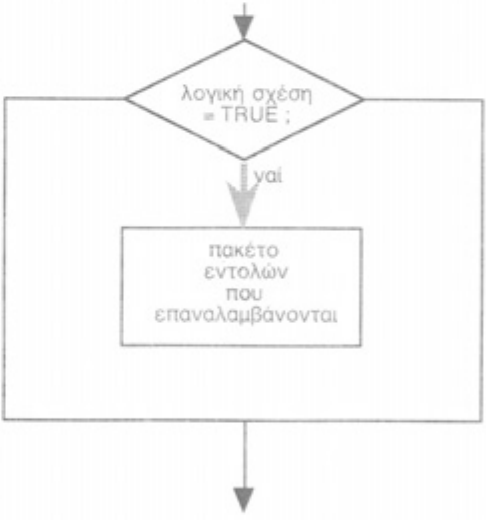
**Pascal**

```
. . . . . .
WHILE λογική σχέση{=TRUE} DO

  BEGIN
    Εντολή_1 ;
    Εντολή_2 ;
    . . . . . .
    Εντολή_N ;
  END ;
. . . . . .
```

**Λογικό διάγραμμα**
χρησιμοποιώντας τίς εντολές
IF THEN ELSE και GO TO




**Λογικό διάγραμμα**
της εντολής WHILE DO




* Ο συμβολισμός του λογικού διαγράμματος της εντολής while προτείνεται από τον συγγραφέα.

**Παρατηρήσεις**

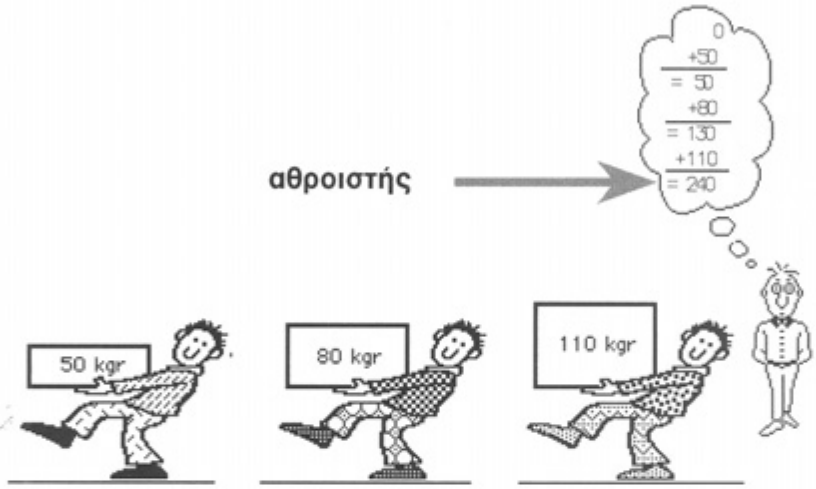
α) Μνημονικός κανόνας: Εφόσον ισχύει η λογική σχέση τότε εκτελούμε την επόμενη εντολή (που εδώ είναι η πρώτη εντολή της επανάληψης).

β) Κάποια από τις εντολές του πακέτου που επαναλαμβάνεται πρέπει να επηρεάζει τις μεταβλητές που υπεισέρχονται στη λογική σχέση της εντολής WHILE.

γ) Αν η εντολή που πρέπει να επαναλαμβάνεται είναι μόνο μία τότε δε χρειάζεται το begin end.

## Η εντολή επανάληψης WHILE.
## Η έννοια του αθροιστή.

ΑΘΡΟΙΣΤΕΣ
Συνήθως στα προβλήματα που αντιμετωπίζονται στον προγραμματισμό χρειάζεται να υπολογισθεί η ποσότητα κάποιου μεγέθους.
Οι μεταβλητές που χρησιμοποιούμε και στις οποίες συσσωρεύεται η ολική ποσότητα του μεγέθους λέγονται αθροιστές.
Οι αθροιστές στην αρχή του προγράμματος παίρνουν μιά αρχική τιμή (κατά κανόνα




1) Δίνονται διαδοχικά από το πληκτρολόγιο τα βάρη μερικών κιβωτίων (απροσδιόριστο το πλήθος τους) με το τελευταίο κιβώτιο να είναι άδειο (βάρος μηδέν). Ζητείται να υπολογισθεί το συνολικό βάρος των κιβωτίων.

```
PROGRAM Loop1 ;
  VAR  baros, synolikoBaros : INTEGER ;
  BEGIN
     synolikoBaros := 0 ;     { αθροιστής }
     READLN ( baros ) ;
     WHILE   baros > 0   DO
         BEGIN
           synolikoBaros := synolikoBaros + baros ;
           READLN ( baros ) ;
         END ;
      WRITELN ( synolikoBaros ) ;
  END .
```

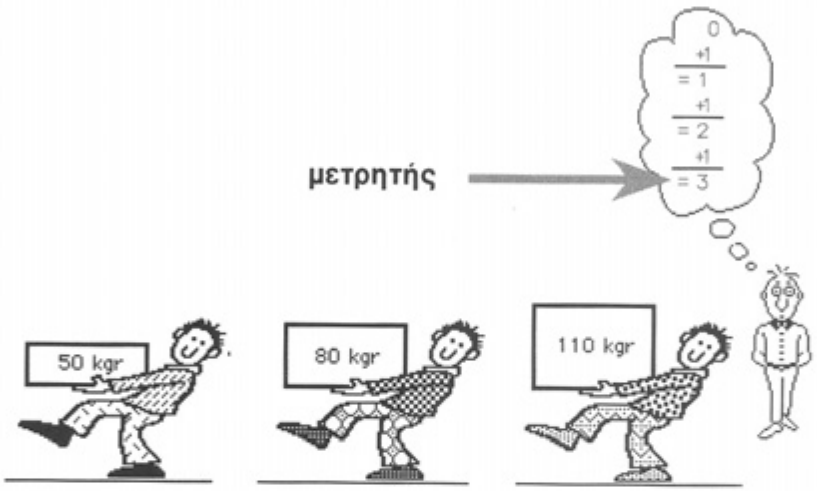
## Η εντολή επανάληψης WHILE.
## Η έννοια του μετρητή.

ΜΕΤΡΗΤΕΣ

Συνήθως στα προβλήματα που αντιμετωπίζονται στον προγραμματισμό χρειάζεται να καταμετρηθεί το πλήθος κάποιων συμβάντων ή η ποσότητα μεγεθών που μετριούνται με το κομμάτι ...

Οι μεταβλητές που χρησιμοποιούμε και στις οποίες υπολογίζεται το πλήθος των συμβάντων/κομματιών λέγονται μετρητές.

Οι μετρητές στην αρχή του προγράμματος παίρνουν μιά αρχική τιμή (κατά κανόνα μηδέν).




2) Δίνονται διαδοχικά από το πληκτρολόγιο τα βάρη μερικών κιβωτίων (απροσδιόριστο το πλήθος τους) με το τελευταίο κιβώτιο να είναι άδειο (βάρος μηδέν). Ζητείται να μετρηθεί το πλήθος των κιβωτίων.

```
PROGRAM Loop2 ;
  VAR  baros, metrntnsKibwtiwn : INTEGER ;
  BEGIN
    metrntnsKibwtiwn := 0 ;
    READLN ( baros ) ;
    WHILE   baros > 0   DO
        BEGIN
          metrntnsKibwtiwn := metrntnsKibwtiwn + 1 ;
          READLN ( baros ) ;
        END ;
     WRITELN ( metrntnsKibwtiwn ) ;
  END .
```

3) Δίνονται διαδοχικά από το πληκτρολόγιο τα βάρη μερικών κιβωτίων (απροσδιόριστο το πλήθος τους) με το τελευταίο κιβώτιο είναι άδειο (βάρος μηδέν). Ζητείται να υπολογισθεί το μέσο βάρος των κιβωτίων.

```
PROGRAM Loop3 ;
  VAR  baros, synolikoBaros, metrntnsKibwtiwn : INTEGER ;
       MesoBarosKibwtiwn : REAL ;
  BEGIN
    synolikoBaros := 0 ;
    metrntnsKibwtiwn := 0 ;
    READLN ( baros ) ;
    WHILE   baros > 0   DO
        BEGIN
          synolikoBaros := synolikoBaros + baros ;          {αθροιστής}
          metrntnsKibwtiwn := metrntnsKibwtiwn + 1 ;  {μετρητής}
          READLN ( baros ) ;
        END ;
     MesoBarosKibwtiwn := ( synolikoBaros / metrntnsKibwtiwn ) ;
     WRITELN ( MesoBarosKibwtiwn ) ;
  END .
```

4) Δίνονται διαδοχικά από το πληκτρολόγιο τα βάρη μερικών κιβωτίων (απροσδιόριστο το πλήθος τους) με το τελευταίο κιβώτιο να είναι άδειο (βάρος μηδέν). Ζητείται να μετρηθεί το πλήθος των κιβωτίων που έχουν βάρος μεγαλύτερο των 100Kgr.

```
PROGRAM Loop4 ;
  VAR  baros, metrntnsMegalwn : INTEGER ;
  BEGIN
    metrntnsMegalwn := 0 ;
    READLN ( baros ) ;
    WHILE   baros > 0   DO
        BEGIN
          IF baros > 100
            THEN metrntnsMegalwn := metrntnsMegalwn + 1 ;
          READLN ( baros ) ;
        END ;
     WRITELN ( metrntnsMegalwn ) ;
  END .
```

## Η εντολή επανάληψης WHILE.

5) Δίνονται διαδοχικά από το πληκτρολόγιο τα βάρη μερικών κιβωτίων (απροσδιόριστο το πλήθος τους) με το τελευταίο κιβώτιο να άδειο (βάρος μηδέν). Ζητείται να μετρηθεί α) το πλήθος των βαριών κιβωτίων (έχουν βάρος μεγαλύτερο των 100Kg) β) το πλήθος των μεσαίων κιβωτίων (έχουν βάρος μικρότερο ή ίσο των 100Kgr και μεγαλύτερο των 50Kgr) και γ) το πλήθος των ελαφρών κιβωτίων (έχουν βάρος μικρότερο ή ίσο των 50Kgr).

```
PROGRAM Loop5 ;
 VAR  baros, metrMeg, metrMes, metrMik : INTEGER ;
 BEGIN
   metrMeg := 0 ;
   metrMes := 0 ;
   metrMik := 0 ;
   READLN ( baros ) ;
   WHILE   baros > 0   DO
       BEGIN
         IF baros > 100
           THEN metrMeg := metrMeg + 1
           ELSE  IF baros > 50
                    THEN metrMes := metrMes + 1
                    ELSE  metrMik := metrMik + 1 ;
         READLN ( baros ) ;
       END ;
    WRITELN ( metrMeg, metrMes, metrMik ) ;
 END .
```

## Η εντολή επανάληψης WHILE.

6) Δίνονται διαδοχικά από το πληκτρολόγιο τα βάρη μερικών κιβωτίων (απροσδιόριστο το πλήθος τους) με το τελευταίο κιβώτιο να είναι άδειο (βάρος μηδέν). Ζητείται να υπολογισθεί α) το μέσο βάρος των βαριών κιβωτίων (έχουν βάρος μεγαλύτερο των 100Kg) β) το μέσο βάρος των μεσαίων κιβωτίων (έχουν βάρος μικρότερο ή ίσο των 100Kgr και μεγαλύτερο των 50Kgr) και γ) το μέσο βάρος των ελαφρών κιβωτίων (έχουν βάρος μικρότερο ή ίσο των 50Kgr).

```
PROGRAM Loop6 ;
  VAR  baros : INTEGER ;
       metrMeg, metrMes, metrMik : INTEGER ;
       syuMeg, syuMes, syuMik : INTEGER ;
       MesBarMeg, MesBarMes, MesBarMik : REAL ;
  BEGIN
    syuMeg := 0 ;
    syuMes := 0 ;
    syuMik := 0 ;
    metrMeg := 0 ;
    metrMes := 0 ;
    metrMik := 0 ;
    READLN ( baros ) ;
    WHILE   baros > 0   DO
        BEGIN
          IF baros > 100
             THEN
                BEGIN
                  syuMeg := syuMeg + baros ;
                  metrMeg := metrMeg + 1 ;
                END
             ELSE  IF baros > 50
                      THEN
                         BEGIN
                           syuMes := syuMes + baros ;
                           metrMes := metrMes + 1 ;
                         END
                      ELSE
                         BEGIN
                           syuMik := syuMik + baros ;
                           metrMik := metrMik + 1 ;
                         END ;
          READLN ( baros ) ;
        END ;
```

```
    IF metrMeg <> 0
       THEN MesBarMeg := ( syuMeg / metrMeg )
       ELSE  MesBarMeg := 0 ;

     IF metrMes <> 0
       THEN MesBarMes := ( syuMes / metrMes )
       ELSE  MesBarMes := 0 ;

    IF metrMik <> 0
       THEN MesBarMik := ( syuMik / metrMik )
       ELSE  MesBarMik := 0 ;

    WRITELN ( 'Μέσος όρος μεγάλων κιβωτίων:', MesBarMeg ) ;
    WRITELN ( 'Μέσος όρος μεσαίων κιβωτίων:', MesBarMes ) ;
    WRITELN ( 'Μέσος όρος μικρών κιβωτίων:', MesBarMik ) ;
END .
```

**Παρατηρήσεις**

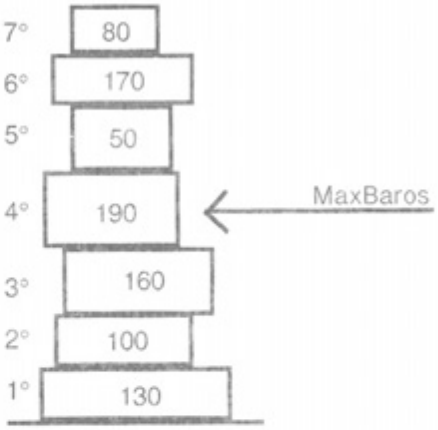
1) Οι τελευταίες εντολές IF χρησιμοποιήθηκαν για την περίπτωση εκείνη που π.χ. δεν υπήρχε κανένα ελαφρύ κιβώτιο. Τότε ο metrMeg θα ισούτο με μηδέν και η διαίρεση syvMeg διά metrMeg θα δημιουργούσε πρόβλημα.
Κανόνας: Ένα πρόγραμμα πρέπει να καλύπτει όλες τις περιπτώσεις του προβλήματος που υποστηρίζει.

2) Όπως έχει ήδη διαφανεί τα ονόματα των μεταβλητών, των σταθερών και των προγραμμάτων δεν είναι τυχαία αλλά υπονοούν την χρήση τους. Έτσι στην προηγούμενη άσκηση καταλαβαίνουμε εύκολα ότι η μεταβλητή metrMeg είναι ο μετρητής των μεγάλων κιβωτίων, η μεταβλητή syvMik περιέχει το συνολικό βάρος των μικρών κιβωτίων ενώ η μεταβλητή MesBarMes είναι ο μέσος όρος των βαρών των κιβωτίων μεσαίου μεγέθους.
3) Τα αποτελέσματα πρέπει να συνοδεύονται από μηνύματα σαφή και περιεκτικά όπως γίνεται με τα τρία τελευταία writeln.

## Η εντολή επανάληψης WHILE.

7) Δίνονται διαδοχικά από το πληκτρολόγιο τα βάρη μερικών κιβωτίων (απροσδιόριστο το πλήθος τους) με το τελευταίο κιβώτιο να είναι άδειο (βάρος μηδέν). Ζητείται να βρεθεί το μεγαλύτερο βάρος.

```
PROGRAM Loop7 ;
  VAR baros, MaxBaros : INTEGER ;
  BEGIN
    READLN ( baros ) ;
    MaxBaros := baros ;
    WHILE   baros > 0   DO
        BEGIN
          IF   baros > MaxBaros
             THEN  MaxBaros := baros ;
          READLN ( baros ) ;
        END ;
    WRITELN ( MaxBaros ) ;
  END .
```



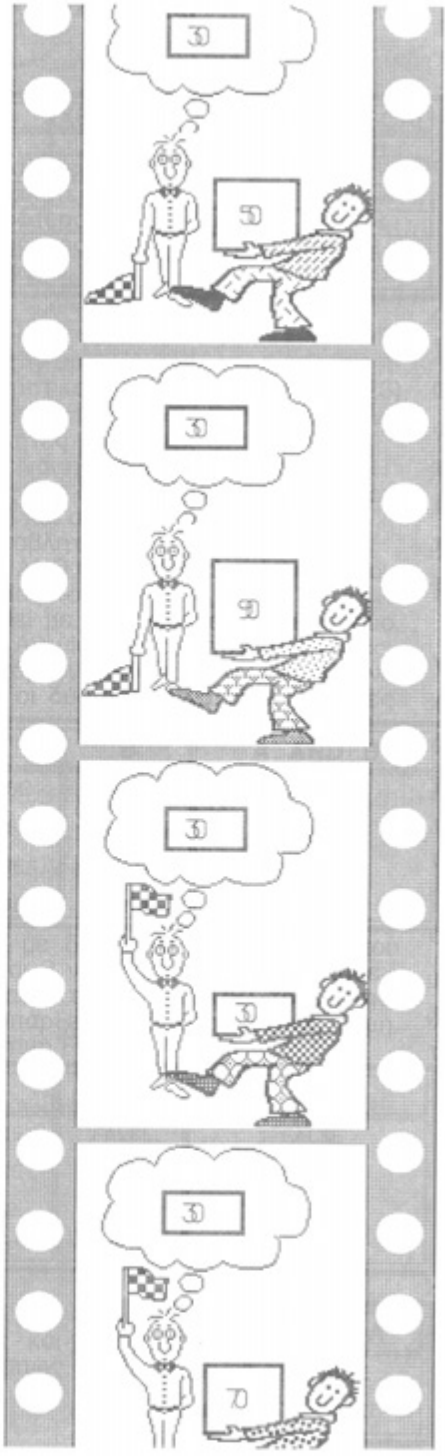

8) Δίνονται διαδοχικά από το πληκτρολόγιο τα βάρη μερικών κιβωτίων (απροσδιόριστο το πλήθος τους) με το τελευταίο κιβώτιο να είναι άδειο (βάρος μηδέν). Ζητείται να βρεθεί η σειρά του κιβωτίου με το μεγαλύτερο βάρος.

```
PROGRAM Loop8 ;
  VAR baros, MaxBaros, seira, ThesnMaxBarous : INTEGER ;
  BEGIN
    seira := 1 ;
    READLN ( baros ) ;
    MaxBaros := baros ;
    ThesnMaxBarous := seira ;
    WHILE   baros > 0   DO
        BEGIN
          READLN ( baros ) ;
          seira := seira + 1 ;
          IF   baros > MaxBaros
             THEN
              BEGIN
                MaxBaros := baros ;
                ThesnMaxBarous := seira ;
              END ;
        END ;
    WRITELN ( ThesnMaxBarous ) ;
  END .
```

## Η εντολή επανάληψης WHILE.

9) Δίνεται από το πληκτρολόγιο ένας συγκεκριμένος αριθμός και στη συνέχεια δίνονται διαδοχικά από το πληκτρολόγιο τα βάρη μερικών κιβωτίων (απροσδιόριστο το πλήθος τους) με το τελευταίο κιβώτιο να είναι άδειο (βάρος μηδέν). Ζητείται να ενημερώνεται μια ένδειξη για το αν υπάρχει κιβώτιο με βάρος ίδιο με τον αριθμό που δόθηκε στην αρχή.

Θα χρησιμοποιήσουμε την έννοια της σημαίας που μάθαμε σε προηγούμενο κεφάλαιο. Εκεί αναφέραμε ότι "συνήθως στα προβλήματα που αντιμετωπίζονται στον προγραμματισμό χρειάζεται να δηλωθεί αν συνέβη ή όχι κάποιο περιστατικό. Αυτό γίνεται με τη χρήση μιας μεταβλητής τύπου boolean η οποία στη συγκεκριμένη περίπτωση λέγεται σημαία (flag). Η χρησιμότητά της είναι ανάλογη με τον επόπτη γραμμών στο ποδόσφαιρο."
Στα διπλανά στιγμιότυπα παρουσιάζεται ο τρόπος που χρησιμοποιείται η σημαία γιά να δηλώνεται στο τέλος αν βρέθηκε κιβώτιο με το ζητούμενο βάρος.




```
PROGRAM Loop9 ;
  VAR
     arithmos, baros : INTEGER ;
     yparxei : BOOLEAN ;
  BEGIN
     READLN ( arithmos ) ;
     yparxei := FALSE ; {κατεβάζει την σημαία}
     READLN ( baros ) ;
     WHILE   baros > 0   DO
          BEGIN
            IF baros=arithmos
               THEN       {ανέβασε την σημαία}
                  yparxei := TRUE ;
            READLN ( baros ) ;
          END ;
      WRITELN ( yparxei ) ;
  END .
```

## Η εντολή επαναληψης WHILE.

10) Υπάρχει ένα προκαθορισμένο άνω όριο (πχ τα 100Kgr). Δίνονται διαδοχικά από το πληκτρολόγιο τα βάρη μερικών κιβωτίων (απροσδιόριστο το πλήθος τους) με το τελευταίο κιβώτιο να είναι άδειο (βάρος μηδέν). Ζητείται να ενημερώνεται μία ένδειξη γιά το αν υπήρξε έστω και ένα κιβώτιο που το βάρος του να υπερέβηκε το προκαθορισμένο άνω όριο.

```
PROGRAM Loop10 ;
  VAR     baros : INTEGER ;
          yperbasn : BOOLEAN ;
  BEGIN
    yperbasn := FALSE ;
    READLN ( baros ) ;
    WHILE   baros > 0   DO
        BEGIN
          IF  baros > 100
             THEN  yperbasn := TRUE ;
          READLN ( baros ) ;
        END ;
     WRITELN ( yperbasn ) ;
  END .
```

11) Υπάρχει ένα προκαθορισμένο άνω όριο (πχ τα 100Kgr). Δίνονται διαδοχικά από το πληκτρολόγιο τα βάρη μερικών κιβωτίων (απροσδιόριστο το πλήθος τους) με το τελευταίο κιβώτιο να είναι άδειο (βάρος μηδέν). Ζητείται να υπολογισθεί το συνολικό βάρος των κιβωτίων αν η φόρτωση σταματήσει είτε με το άδειο κιβώτιο είτε αν βρεθεί κιβώτιο με βάρος μεγαλύτερο του ανω ορίου.

```
PROGRAM Loop11 ;
  VAR     baros, synolikoBaros : INTEGER ;
          yperbasn : BOOLEAN ;
  BEGIN
    synolikoBaros := 0 ;
    READLN ( baros ) ;
    IF  baros >=100
        THEN  yperbasn := TRUE
        ELSE  yperbasn := FALSE ;
    WHILE  ( ( baros > 0 )  AND  ( NOT yperbasn ) )   DO
        BEGIN
          IF  baros < 100
             THEN  BEGIN
                      synolikoBaros := synolikoBaros + baros ;
                      yperbasn := TRUE ;
                   END ;
          READLN ( baros ) ;
        END ;
     WRITELN ( synolikoBaros ) ;
  END .
```

# Pascal

## Μια Μεθοδική Προσέγγιση

Το βιβλίο αυτό προσπαθεί να δώσει μια άλλη προσέγγιση στον τρόπο διδασκαλίας γλωσσών προγραμματισμού.

Ακολουθώντας μια μέθοδο προσέγγισης μέσα από λυμένα προβλήματα, και αφού δώσει στον αναγνώστη, όσο πιο νωρίς γίνεται, τα απαραίτητα εργαλεία (εντολές και διαδικασίες), τον φέρνει σε επαφή με το πρόγραμμα.

Οι ασκήσεις, που δίνονται για να εξοικειωθεί ο αναγνώστης με τις εντολές, είναι πρότυπα για επόμενες. Βλέπουμε λοιπόν ένα πρόγραμμα που χρησιμοποιείται για να κατανοήσουμε μια συνάρτηση, να παρουσιάζεται στη συνέχεια εμπλουτισμένο με μια νέα εντολή, να εξελίσσεται χρησιμοποιώντας μια νέα τεχνική και να καταλήγει μετά σε διαδικασία που θα χρησιμοποιηθεί από ένα άλλο πρόγραμμα.

Η καλύτερη λύση για κάποιο πρόβλημα δε δίνεται εύκολα. Λύσεις εξ ουρανού ενθουσιάζουν το κοινό μιας παράστασης, αλλά δεν είναι τρόπος διδασκαλίας. Το βιβλίο προσπαθεί να οδηγήσει τη σκέψη του αναγνώστη σε κάποια λύση, που είναι εφικτή με τις γνώσεις που έχει μέχρι εκείνο το σημείο. Αφού βρεθεί μια λύση, τότε αναπτύσσει κάποιο προβληματισμό για βελτίωση, δημιουργώντας έτσι την ανάγκη για μια νέα εντολή ή μέθοδο. Στη συνέχεια παρουσιάζει αυτή την εντολή, και αφού ο αναγνώστης εξοικειωθεί με τη χρήση της, τότε μόνο δίνεται η βελτιωμένη έκδοση του προβλήματος που τον απασχολεί.

Είναι ένα βιβλίο που διαβάζεται από την πρώτη μέχρι την τελευταία σελίδα σαν μυθιστόρημα, και ελπίζουμε να είναι το ίδιο συναρπαστικό...

Κεντρική Διάθεση: ΚΛΕΙΔΑΡΙΘΜΟΣ
Σπ. Τρικούπη 16 ΑΘΗΝΑ τηλ. 3608408, 3604492

ISBN 960-209-118-5