

# Pascal

## Μια Μεθοδική Προσέγγιση

Τάσος Λαδιάς



Τεχνικές Δομημένου Προγραμματισμού  
Μέσα από Λυμένα  
Παραδείγματα

# Περιεχόμενα

Πρόλογος

9

## 1. ΒΑΣΙΚΕΣ ΕΝΤΟΛΕΣ ΤΗΣ ΓΛΩΣΣΑΣ PASCAL

### 1.1. ΕΙΣΑΓΩΓΗ

|                                                                           |    |
|---------------------------------------------------------------------------|----|
| 1.1.1. Ο προγραμματιστής, η δακτυλογράφος, ο μεταφραστής και το μηχάνημα. | 11 |
| 1.1.2. Η δομή ενός προγράμματος pascal.                                   | 12 |
| 1.1.3. Τα είδη των δεδομένων της pascal.                                  | 13 |
| 1.1.4. Δηλώσεις μεταβλητών.                                               | 14 |
| 1.1.5. Εντολή εισόδου δεδομένων.                                          | 16 |
| 1.1.6. Εντολή εκχώρησης.                                                  | 17 |
| 1.1.7. Το πρόβλημα, ο αλγόριθμος, ο ψευδοκώδικας και το πρόγραμμα.        | 18 |
| 1.1.8. Εντολή εμφάνισης στην οθόνη.                                       | 19 |
| 1.1.9. Σχόλια στο κείμενο του προγράμματος.                               | 21 |
| 1.1.10. Φιλικότητα και τεκμηρίωση του προγράμματος.                       | 22 |
| 1.1.11. Αριθμητικές πράξεις και οι συναρτήσεις DIV, MOD.                  | 23 |
| 1.1.12. Προκαθορισμένες συναρτήσεις της Pascal.                           | 27 |
| 1.1.13. Βαθμωτοί, μη-βαθμωτοί τύποι και οι συναρτήσεις PRED και SUCC.     | 28 |
| 1.1.14. Λογικοί τελεστές και οι συναρτήσεις AND, OR και NOT.              | 29 |

### 1.2. ΕΝΤΟΛΕΣ ΑΠΟΦΑΣΕΩΝ

|                                                                 |    |
|-----------------------------------------------------------------|----|
| 1.2.1. Εντολές λήψης αποφάσεων.                                 | 32 |
| 1.2.2. Η εντολή IF...THEN...ELSE.                               | 33 |
| 1.2.3. Ο μέγιστος μεταξύ δύο ακεραίων αριθμών.                  | 34 |
| 1.2.4. Η έννοια της σημαίας (flag).                             | 38 |
| 1.2.5. IF μέσα σε IF (φωλιασμένο ή nested).                     | 39 |
| 1.2.6. Αξιολόγηση μαθητή με φωλιασμένο IF.                      | 41 |
| 1.2.7. Η εντολή case.                                           | 44 |
| 1.2.8. Δηλώσεις LABEL και η εντολή GOTO ... (ένα πρώτο σχόλιο). | 49 |

### 1.3. ΕΝΤΟΛΕΣ ΕΠΑΝΑΛΗΨΗΣ

|                                                                                 |    |
|---------------------------------------------------------------------------------|----|
| 1.3.1. Η αναγκαιότητα των εντολών επανάληψης.                                   | 51 |
| 1.3.2. Η εντολή επανάληψης WHILE.                                               | 53 |
| 1.3.3. Η έννοια του αθροιστή.                                                   | 54 |
| 1.3.4. Η έννοια του μετρητή.                                                    | 55 |
| 1.3.5. Περιοχή του προγράμματος που δηλώνονται οι σταθερές (CONST).             | 63 |
| 1.3.6. Ο 8-ψήφιος κώδικας χαρακτήρων (ELOT 928) και οι συναρτήσεις CHR και ORD. | 66 |
| 1.3.7. Η αναγκαιότητα της εντολής επανάληψης REPEAT...UNTIL.                    | 67 |
| 1.3.8. Η εντολή επανάληψης REPEAT...UNTIL.                                      | 68 |
| 1.3.9. Οι συναρτήσεις ROUND και TRUNC.                                          | 74 |
| 1.3.10. Η εντολή επανάληψης FOR.                                                | 75 |
| 1.3.11. Η συνάρτηση ODD.                                                        | 77 |
| 1.3.12. Φωλιασμένες επαναλήψεις.                                                | 80 |
| 1.3.13. Συσχετισμοί μεταξύ των εντολών επανάληψης.                              | 84 |
| 1.3.14. GOTO... No thank you (σχόλιο 2).                                        | 85 |
| 1.3.15. Ευανάγνωστα προγράμματα                                                 | 86 |

## 2. ΤΕΧΝΙΚΕΣ ΚΑΙ ΕΡΓΑΛΕΙΑ ΔΟΜΗΜΕΝΟΥ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

### 2.1. ΙΕΡΑΡΧΙΚΟΣ ΣΧΕΔΙΑΣΜΟΣ ΠΡΟΓΡΑΜΜΑΤΟΣ

|                                                       |    |
|-------------------------------------------------------|----|
| 2.1.1. Ιεραρχικός σχεδιασμός προγράμματος.            |    |
| 2.1.1.1. Οπτικός Πίνακας Περιεχομένων (V.T.O.C.)      | 89 |
| 2.1.1.2. Η σειρά εκτέλεσης των ενεργειών του V.T.O.C. | 90 |
| 2.1.1.3. Ο ψευδοκώδικας και το πρόγραμμα              | 91 |
| 2.1.1.4. Η σύνδεση των επιπέδων                       | 94 |
| 2.1.2. Τα modules                                     | 96 |

## Περιεχόμενα

### 2.2. ΔΙΑΔΙΚΑΣΙΕΣ

|                                                                                 |     |
|---------------------------------------------------------------------------------|-----|
| 2.2.1. Οι διαδικασίες                                                           | 98  |
| 2.2.1.1. Περιγραφή της διαδικασίας και μηχανισμός κλήσης αυτής.                 | 99  |
| 2.2.1.2. Διαδικασίες χωρίς παραμέτρους (καθολικές ή global μεταβλητές).         | 100 |
| 2.2.1.3. Διαδικασίες με παραμέτρους. Τοπικές ή local μεταβλητές σε διαδικασίες. | 102 |
| 2.2.1.4. Παράμετροι τιμής ή value parameters                                    | 105 |
| 2.2.1.5. Παράμετροι μεταβλητής ή variable parameters                            | 108 |
| 2.2.1.6. Πολλαπλή κλήση διαδικασίας από πρόγραμμα.                              | 111 |
| 2.2.1.7. Παράμετροι ορισμού (formal) & κλήσης (actual)                          | 111 |
| 2.2.1.8. Διαδικασίες ReverseTheNumber και CountTheDigits                        | 113 |
| 2.2.1.9. Διαδικασίες EuresnTaxnsMegethus και ApogymwnsApoAkraiaPsnfia           | 114 |
| 2.2.1.10. Μηχανισμός πέρασματος παραμέτρων.                                     | 115 |
| 2.2.2. Διαχείριση διαδικασιών                                                   |     |
| 2.2.2.1. Ανεξάρτητες μεταξύ τους διαδικασίες χωρίς ιεραρχία.                    | 116 |
| 2.2.2.2. Καθολικές μεταβλητές και παράμετροι μεταβλητών.                        | 118 |
| 2.2.2.3. Η οδηγία προς τον μεταφραστή για ενσωμάτωση αρχείου στο πρόγραμμα.     | 120 |
| 2.2.2.4. Κλήσεις διαδικασιών από ένα πρόγραμμα. Ένα πρώτο μενού.                | 121 |
| 2.2.2.5. Εξάρτημένες μεταξύ τους διαδικασίες χωρίς ιεραρχία.                    | 123 |
| 2.2.2.6. Εξάρτημένες μεταξύ τους διαδικασίες με ιεραρχία.                       | 125 |
| 2.2.2.7. Διαδικασία ασφαλούς εισαγωγής ακεραίων αριθμών από το πληκτρολόγιο.    | 127 |
| 2.2.2.8. Κλήσεις διαδικασιών με ιεραρχία από ένα πρόγραμμα.                     | 128 |
| 2.2.3. Συναρτήσεις                                                              | 130 |
| 2.2.3.1. Σύγκριση στο πέρασμα παραμέτρων μεταξύ διαδικασίας και συνάρτησης.     | 131 |
| 2.2.3.2. Κλήση συνάρτησης και διαδικασίας από το πρόγραμμα (χωρίς ιεραρχία).    | 132 |
| 2.2.3.3. Κλήση συνάρτησης από διαδικασία (χωρίς ιεραρχία)                       | 134 |
| 2.2.3.4. Κλήση συνάρτησης από διαδικασία (με ιεραρχία)                          | 136 |
| 2.2.3.5. Συνάρτηση ως παραμέτρος κλήσης.                                        | 138 |
| 2.2.3.6. Συνάρτηση ως παραμέτρος ορισμού.                                       | 140 |
| 2.2.4. Αναδρομικές διαδικασίες.                                                 | 141 |
| 2.2.4.1. Άμεσα αναδρομικές διαδικασίες.                                         | 142 |
| 2.2.4.2. Αναδρομή και επανάληψη.                                                | 148 |
| 2.2.4.3. Το παραγοντικό με αναδρομή και επανάληψη.                              | 149 |
| 2.2.4.4. Η αλλαγή βάσης αριθμού με αναδρομή.                                    | 150 |
| 2.2.4.5. Έμμεσα αναδρομικές διαδικασίες.                                        | 151 |
| 2.2.4.6. Έμμεσα αναδρομικές διαδικασίες και η προαναγγελία δήλωσης διαδικασίας. | 152 |
| 2.2.5. Παραγωγή παλινδρομικών αριθμών                                           |     |
| 2.2.5.1. Ο αλγόριθμος                                                           | 153 |
| 2.2.5.2. Οι Οπτικοί Πίνακες Περιεχομένων                                        | 154 |
| 2.2.5.3. Ο κώδικας του προγράμματος σε Pascal                                   | 156 |
| 2.2.5.4. Ο κώδικας των υποπρογραμμάτων                                          | 157 |

### 3. ΣΤΑΤΙΚΕΣ ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ ΤΗΣ PASCAL

#### 3.1. ΤΥΠΟΙ

|                                                              |     |
|--------------------------------------------------------------|-----|
| 3.1.1. Τύποι που ορίζονται από το χρήστη.                    | 159 |
| 3.1.1.1. Τύποι που ορίζονται από το χρήστη & διαδικασίες.    | 162 |
| 3.1.1.2. Τύποι υποπεριοχής.                                  | 164 |
| 3.1.1.3. Ερωτήσεις στους τύπους που ορίζονται από το χρήστη. | 165 |

#### 3.2. ΠΙΝΑΚΕΣ

|                                              |     |
|----------------------------------------------|-----|
| 3.2.1. Η αναγκαιότητα των πινάκων.           | 168 |
| 3.2.2. Πίνακες                               | 169 |
| 3.2.3. Παραδείγματα περιγραφής πινάκων       | 172 |
| 3.2.4. Νομίσματα                             | 174 |
| 3.2.5. Πίνακας - μετρητής                    | 175 |
| 3.2.6. Προτεινόμενες ασκήσεις στους πίνακες. | 176 |
| 3.2.7. Ερωτήσεις στους πίνακες.              | 177 |
| 3.2.8. Συμπυκνωμένοι πίνακες.                | 178 |
| 3.2.9. Αρμαθίες χαρακτήρων (strings).        | 179 |
| 3.2.10. Μενού με πίνακες                     | 180 |

## Περιεχόμενα

|                                                                                               |     |
|-----------------------------------------------------------------------------------------------|-----|
| 3.2.10.1. Οι προδιαγραφές                                                                     | 181 |
| 3.2.10.2. Οι διαδικασίες                                                                      | 185 |
| 3.2.10.3. Αλγόριθμος ταξινόμησης πίνακα (μέθοδος της φουσαλιδας ή bubble sort)                | 192 |
| 3.2.10.4. Ο αλγόριθμος της δυαδικής αναζήτησης                                                | 194 |
| 3.2.10.5. Το κύριο πρόγραμμα                                                                  | 198 |
| 3.2.11. Ο αλγόριθμος της ταξινόμησης πίνακα με τη μέθοδο της επιλογής.                        | 206 |
| 3.2.12. Το πρόβλημα του ασανσέρ.                                                              | 209 |
| 3.2.13. Έξοδος από λαβύρινθο με άμεσο αναδρομικό αλγόριθμο.                                   | 212 |
| 3.2.14. Ταξινόμηση πίνακα με έμμεσο αναδρομικό αλγόριθμο.                                     | 215 |
| <b>3.3. ΣΥΝΟΛΑ</b>                                                                            |     |
| 3.3.1. Σύνολα.                                                                                | 222 |
| 3.3.2. Ασκήσεις στα σύνολα                                                                    | 226 |
| <b>3.4. ΕΓΓΡΑΦΕΣ</b>                                                                          |     |
| 3.4.1. Ο ορισμός μιάς εγγραφής (record)                                                       | 230 |
| 3.4.2. Η εντολή with.                                                                         | 232 |
| 3.4.3. Εγγραφές μεταβλητού μήκους.                                                            | 236 |
| 3.4.4. Η καρτέλα ενός ασθενούς.                                                               | 240 |
| 3.4.4.1. Ορισμός της καρτέλας.                                                                | 241 |
| 3.4.4.2. Το πρόγραμμα και πιά ευέλικτος τρόπος περιγραφής της εγγραφής.                       | 242 |
| 3.4.4.3. Διαδικασία γεμίματος εγγραφής (1ος τρόπος).                                          | 243 |
| 3.4.4.4. Διαδικασία γεμίματος εγγραφής (2ος τρόπος).                                          | 244 |
| 3.4.4.5. Διαδικασία τυπώματος εγγραφής.                                                       | 245 |
| 3.4.5. Μία δομή στοιβας και η άσκηση με το γκαράζ.                                            | 246 |
| 3.4.6. Η άσκηση με τα μονοπάτια.                                                              | 251 |
| 3.4.7. Ερωτήσεις στις εγγραφές.                                                               | 257 |
| <b>3.5. ΑΡΧΕΙΑ</b>                                                                            |     |
| 3.5.1. Η αναγκαιότητα των αρχείων.                                                            | 259 |
| 3.5.2. Τα αρχεία στην pascal.                                                                 | 261 |
| 3.5.3. Δημιουργία και τύπων σετσιακού αρχείου                                                 |     |
| 3.5.3.1. Το πρόγραμμα                                                                         | 262 |
| 3.5.3.2. Ορισμοί τύπων                                                                        | 263 |
| 3.5.3.3. Δηλώσεις των μεταβλητών                                                              | 264 |
| 3.5.3.4. Άνοιγμα του αρχείου για γράψιμο                                                      | 265 |
| 3.5.3.5. Γέμισμα και η αποθήκευση μιας εγγραφής                                               | 266 |
| 3.5.3.6. Γέμισμα με εγγραφές του αρχείου                                                      | 267 |
| 3.5.3.7. Κλείσιμο του αρχείου μετά το γράψιμο                                                 | 268 |
| 3.5.3.8. Το άνοιγμα του αρχείου για διάβασμα                                                  | 269 |
| 3.5.3.9. Μεταφορά απο αρχείο στη οθόνη μέσω μνήμης                                            | 270 |
| 3.5.3.10. Τύπων όλων των εγγραφών του αρχείου                                                 | 271 |
| 3.5.3.11. Το κλείσιμο του αρχείου μετά το διάβασμα                                            | 272 |
| 3.5.4. Διαδικασίες και συναρτήσεις της pascal με τις οποίες διαχειρίζεται τα σετσιακά αρχεία. | 273 |
| 3.5.4.1. Δημιουργία και τύπων σετσιακού αρχείου (ένα λίγο πιο ευέλικτο πρόγραμμα).            | 274 |
| 3.5.4.2. Δημιουργία και τύπων σετσιακού αρχείου (το όνομα του αρχείου ως παράμετρος τιμής).   | 275 |
| 3.5.4.3. Δημιουργία και τύπων σετσιακού αρχείου (το αρχείο ως παράμετρος μεταβλητής).         | 276 |
| 3.5.4.4. Ασκήσεις σε αρχεία                                                                   |     |
| 3.5.4.4.1. Το κύριο πρόγραμμα                                                                 | 278 |
| 3.5.4.4.2. Διαδικασία δημιουργίας αρχείου                                                     | 280 |
| 3.5.4.4.3. Διαδικασία εμφάνισης στην οθόνη αρχείου                                            | 281 |
| 3.5.4.4.4. Συνάρτηση: Υπάρχον άδειο αρχείο                                                    | 282 |
| 3.5.4.4.5. Διαδικασία: μέτρημα των στοιχείων αρχείου                                          | 283 |
| 3.5.4.4.6. Διαδικασία: μετρήμα μηδενικών στοιχείων αρχείου                                    | 284 |
| 3.5.4.4.7. Διαδικασία: Σετσιακή αναζήτηση σε αρχείο                                           | 285 |
| 3.5.4.4.8. Διαδικασία αντιγραφής αρχείου σε πίνακα                                            | 286 |
| 3.5.4.4.9. Διαδικασία αντιγραφής πίνακα σε αρχείο                                             | 287 |
| 3.5.4.4.10. Διαδικασία αντιγραφής αρχείου σε αρχείο                                           | 288 |
| 3.5.4.4.11. Διαδικασία ένωσης δύο αρχείων σε ένα τρίτο                                        | 289 |
| 3.5.4.4.12. Η λογική συνάρτηση XOR (Exclusive OR)                                             | 290 |
| 3.5.4.4.13. Πρόγραμμα σύγκρισης δύο αρχείων                                                   | 290 |
| 3.5.4.4.14. Η εντολή SEEK (turbo και SVS pascal)                                              | 291 |

## Περιεχόμενα

|                                                                        |     |
|------------------------------------------------------------------------|-----|
| 3.5.4.4.15. Χρησιμοποιώντας την εντολή SEEK                            |     |
| 3.5.4.4.15.1. Ανάγνωση συγκεκριμένης εγγραφής αρχείου                  | 292 |
| 3.5.4.4.15.2. Δημιουργία αρχείου με χρήση της εντολής SEEK             | 293 |
| 3.5.4.4.15.3. Αντιμετάθεση στοιχείων αρχείου                           | 294 |
| 3.5.5. Αρχεία κειμένων (text files)                                    |     |
| 3.5.5.1. Εισαγωγή                                                      | 295 |
| 3.5.5.2. Strings και texts                                             | 296 |
| 3.5.5.2.1. Εμφάνιση στην οθόνη ενός αρχείου text                       | 297 |
| 3.5.5.2.2. Σελιδοποίηση ενός αρχείου text                              | 297 |
| <br>                                                                   |     |
| <b>4. ΔΥΝΑΜΙΚΕΣ ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ</b>                                    |     |
| <b>4.1. ΔΕΙΚΤΕΣ</b>                                                    |     |
| 4.1.1. Έμμεση προσπέλαση δεδομένων και δείκτες.                        | 303 |
| 4.1.2. Παράδειγμα χειρισμού των δεικτών.                               | 304 |
| 4.1.3. Ορισμός τύπων δεικτών.                                          | 306 |
| 4.1.4. Δηλώσεις μεταβλητών δεικτών.                                    | 307 |
| 4.1.5. Η εντολή NEW (δέσμευση μνήμης).                                 | 308 |
| 4.1.6. Εκχώρηση τιμής εκεί που δείχνει ο δείκτης.                      | 309 |
| 4.1.7. Εκχώρηση της τιμής ενός δείκτη σε έναν άλλο.                    | 310 |
| 4.1.8. Εκχώρηση της διεύθυνσης ενός δείκτη στην διεύθυνση ενός άλλου.  | 311 |
| 4.1.9. Δύο δείκτες δείχνουν την ίδια μεταβλητή.                        | 312 |
| 4.1.10. Παιχνίδια με τους δείκτες είναι επικίνδυνα.                    | 313 |
| 4.1.11. Η εντολή NEW και η σπατάλη μνήμης.                             | 314 |
| 4.1.12. Η εντολή DISPOSE (αποδέσμευση μνήμης).                         | 315 |
| 4.1.13. Η εντολή NEW και η επαναδέσμευση μνήμης.                       | 316 |
| 4.1.14. Εκχώρηση μνήμης σε αποδεσμευμένο δείκτη.                       | 317 |
| 4.1.15. Λογικές σχέσεις μεταξύ διευθύνσεων δεικτών.                    | 319 |
| 4.1.16. Η "διεύθυνση" nil.                                             | 320 |
| 4.1.17. Δέσμευση μνήμης για δείκτες που δείχνουν σε ακεραίους.         | 322 |
| 4.1.18. Εκχώρηση τιμής που δείχνει δείκτη σε στατική μεταβλητή.        | 324 |
| <br>                                                                   |     |
| <b>4.2. ΛΙΣΤΕΣ</b>                                                     |     |
| 4.2.1. Η χρησιμότητα μιάς λίστας με σύνδεση.                           | 326 |
| 4.2.2. Δημιουργία μιας αταξινόμητης λίστας με σύνδεση (στοίβα).        | 327 |
| 4.2.3. Δημιουργία μιας αταξινόμητης λίστας με σύνδεση (ουρά).          | 328 |
| 4.2.4. Μενού διαχείρισης ταξινομημένης λίστας με σύνδεση.              | 329 |
| 4.2.4.1. Η διαδικασία BuildOrderedList.                                | 330 |
| 4.2.4.2. Το λογικό διάγραμμα της διαδικασίας SearchInsert.             | 331 |
| 4.2.4.3. Ο αλγόριθμος της διαδικασίας SearchInsert.                    | 332 |
| 4.2.4.4. Η διαδικασία SearchInsert.                                    | 333 |
| 4.2.4.5. "Τρέξιμο με το χέρι" της διαδικασίας SearchInsert.            | 334 |
| 4.2.4.6. Οι διαδικασίες SearchList και SearchDelete.                   | 339 |
| <br>                                                                   |     |
| <b>4.3. ΔΕΝΔΡΑ</b>                                                     |     |
| 4.3.1. Αμφίδρομες λίστες με σύνδεση και δυαδικά δένδρα.                | 351 |
| 4.3.2. Διάφορες διαδικασίες σε δένδρα. Το κύριο πρόγραμμα και το menu. | 352 |
| 4.3.3. Η διαδικασία KlisnEpillegisasDiadikasias.                       | 353 |
| 4.3.4. Οι διαδικασίες FtiakseDyadiko και DevdroFtiakseFyllo.           | 354 |
| 4.3.5. Δημιουργία δυαδικού δένδρου.                                    | 355 |
| 4.3.6. Η διαδικασία PsakseDevdro.                                      | 387 |
| 4.3.7. Η διαδικασία EisagwgnNeouFyllou.                                | 388 |
| 4.3.8. Περίπατοι σε δένδρα.                                            | 389 |
| <br>                                                                   |     |
| <b>5. Βιβλιογραφία.</b>                                                | 391 |

## Η αναγκαιότητα των εντολών επανάληψης.

### Το πρόβλημα

Να βρεθεί ο μεγαλύτερος από πέντε ακέραιους αριθμούς που δίνονται από το πληκτρολόγιο.

### Ο αλγόριθμος

Το πρόγραμμα `EuresnMaximum_3` που έχουμε δει προηγουμένως βρίσκει το μέγιστο μεταξύ δύο αριθμών. Το πρόγραμμα `EuresnMaxTriwn` βρίσκει το μέγιστο μεταξύ τριών αριθμών χρησιμοποιώντας δύο φορές τον αλγόριθμο του προγράμματος `EuresnMaximum_3` (μία φορά για να βρούμε το μεγαλύτερο από τους δύο πρώτους αριθμούς και την επόμενη για τον προηγούμενο μεγαλύτερο και τον τρίτο αριθμό). Ας στηριχθούμε σ' αυτή την σκέψη και ας την εφαρμόσουμε για να βρούμε το μέγιστο πέντε αριθμών.

### Το πρόγραμμα

```
PROGRAM EuresnMaximumPeute_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 ) ;
```

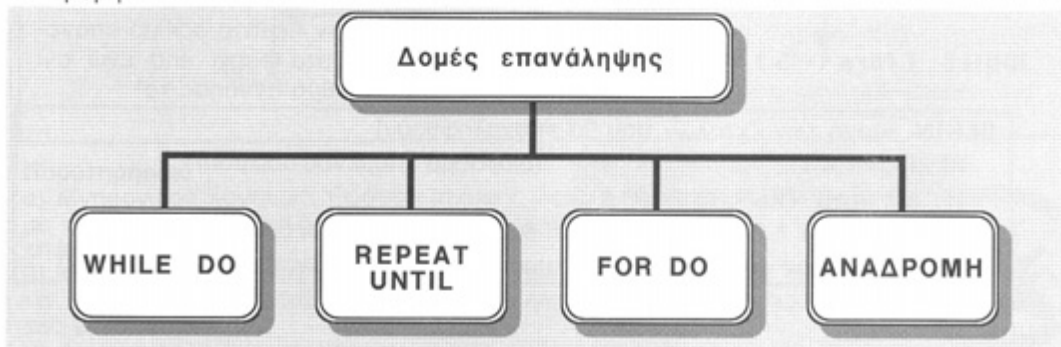
```
END .
```

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

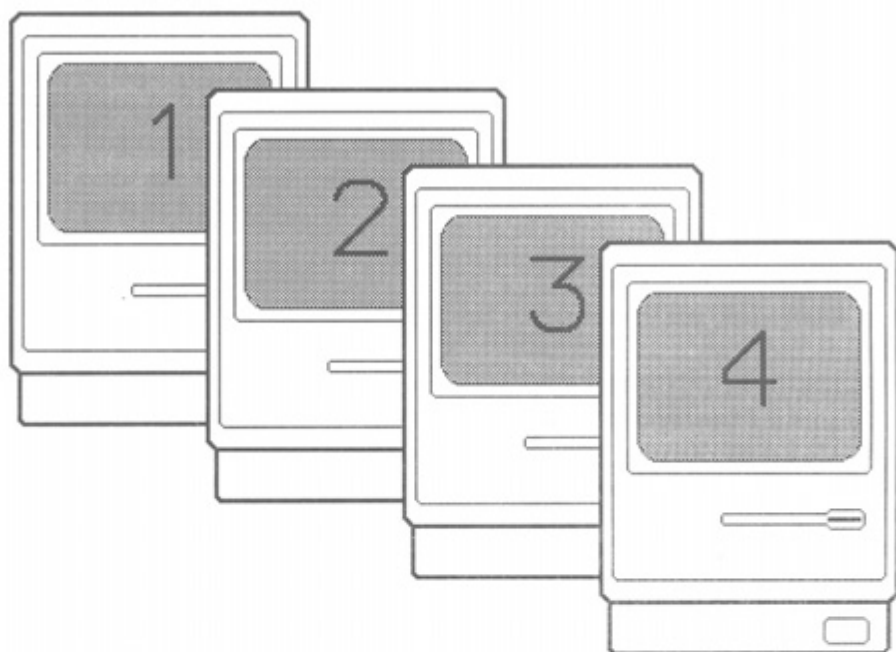
```
READLN ( a ) ;
```

```
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 τρεις 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 μία 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 }

**WHILE** ( **fora** <= 5 ) **DO** {εφόσον δεν διάβασες και τον πέμπτο αριθμό επανέ-  
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 ξαναπήγαينه 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 παρέκαμψε το.

Αρχή πακέτου εντολών.

Εντολή 1.

Εντολή 2.

.....  
Εντολή ν.

Τέλος πακέτου εντολών.

.....

### Pascal

.....  
WHILE λογική\_σχέση(=TRUE) DO

BEGIN

Εντολή\_1 ;

Εντολή\_2 ;

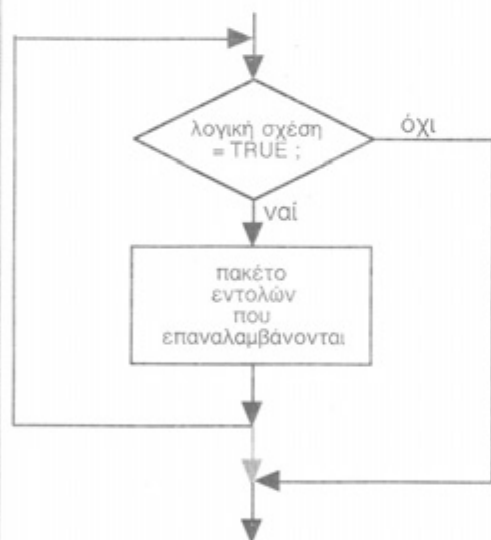
.....

Εντολή\_N ;

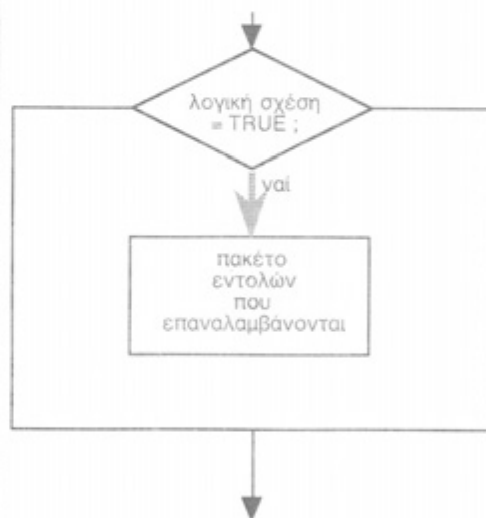
END ;

.....

### Λογικό διάγραμμα χρησιμοποιώντας τις 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.



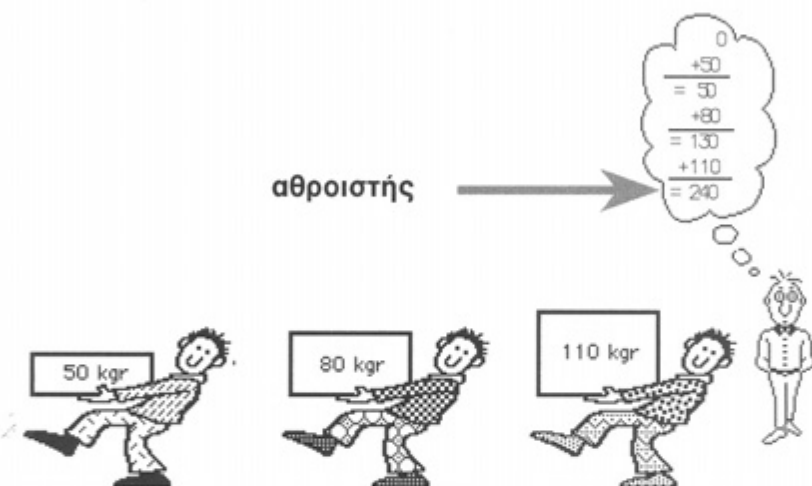
## Η εντολή επανάληψης WHILE. Η έννοια του αθροιστή.

### 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 μία 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, syvolikoBaros : INTEGER ;**

**BEGIN**

**syvolikoBaros := 0 ; { αθροιστής }**

**READLN ( baros ) ;**

**WHILE baros > 0 DO**

**BEGIN**

**syvolikoBaros := syvolikoBaros + baros ;**

**READLN ( baros ) ;**

**END ;**

**WRITELN ( syvolikoBaros ) ;**

**END .**

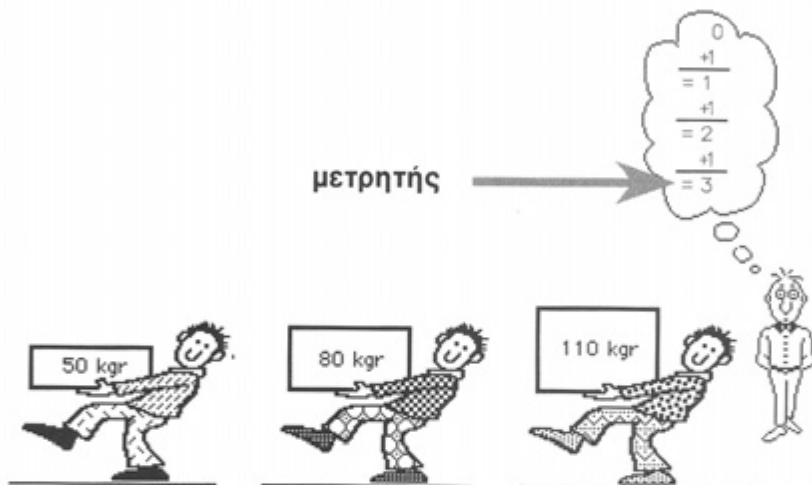
## Η εντολή επανάληψης WHILE. Η έννοια του μετρητή.

### ΜΕΤΡΗΤΕΣ

Συνήθως στα προβλήματα που αντιμετωπίζονται στον προγραμματισμό χρειάζεται να καταμετρηθεί το πλήθος κάποιων συμβάντων ή η ποσότητα μεγεθών που μετριοούνται 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 μία 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 .**

## Η εντολή επανάληψης WHILE.

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, syvolikoBaros, metrntnsKibwtiwn : INTEGER ;**

**MesoBarosKibwtiwn : REAL ;**

**BEGIN**

**syvolikoBaros := 0 ;**

**metrntnsKibwtiwn := 0 ;**

**READLN ( baros ) ;**

**WHILE baros > 0 DO**

**BEGIN**

**syvolikoBaros := syvolikoBaros + baros ;** {αθροιστής}

**metrntnsKibwtiwn := metrntnsKibwtiwn + 1 ;** {μετρητής}

**READLN ( baros ) ;**

**END ;**

**MesoBarosKibwtiwn := ( syvolikoBaros / 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 ;**

## Η εντολή επανάληψης WHILE.

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

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

IF metrMik <> 0
  THEN MesBarMik := ( synMik / 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. δέν υπήρχε κανένα ελαφρύ κιβώτιο. Τότε ο metrMeg θα ισούτο με μηδέν και η διαίρεση synMeg διά metrMeg θα δημιουργούσε πρόβλημα.

Κανόνες: 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 synMik περιέχει το συνολικό βάρος των μικρών κιβωτίων ενώ η μεταβλητή MesBarMes είναι ο μέσος όρος των βαρών των κιβωτίων μεσαίου μεγέθους.

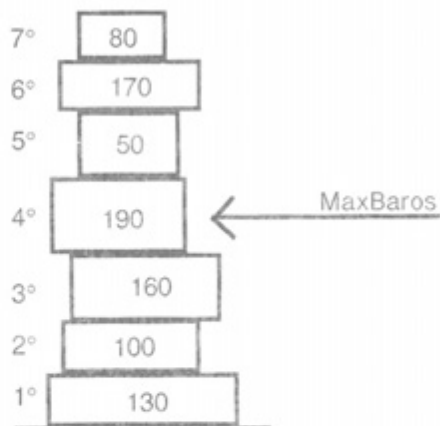
3) Τα αποτελέσματα πρέπει να συνοδεύονται από μηνύματα σαφή και περιεκτικά όπως γίνεται με τα τρία τελευταία writeln.



## 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 για να δηλώνεται στο τέλος αν βρέθηκε κιβώτιο με το ζητούμενο βάρος.

### PROGRAM Loop9 ;

VAR

arithmos, baros : INTEGER ;

yparkei : BOOLEAN ;

BEGIN

READLN ( arithmos ) ;

yparkei := FALSE ; {κατεβάζει την σημαία}

READLN ( baros ) ;

WHILE baros > 0 DO

BEGIN

IF baros=arithmos

THEN {ανέβασε την σημαία}

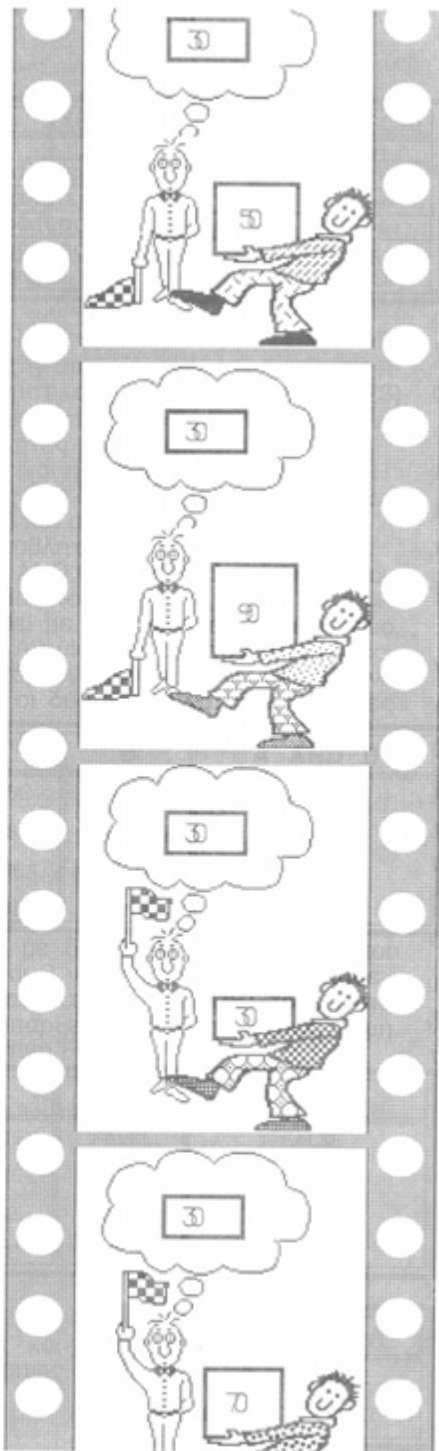
yparkei := TRUE ;

READLN ( baros ) ;

END ;

WRITELN ( yparkei ) ;

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 για 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, syvolikoBaros : INTEGER ;
        yperbasn : BOOLEAN ;

BEGIN
    syvolikoBaros := 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
                syvolikoBaros := syvolikoBaros + baros ;
                yperbasn := TRUE ;
            END ;
            READLN ( baros ) ;
        END ;
        WRITELN ( syvolikoBaros ) ;
    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 πρό-πλασμα 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  
Ση. Τσικουπη 16 ΑΘΗΝΑ τηλ. 3608408-3604492

ISBN 960-209-118-5