

**Ανάπτυξη Εφαρμογών  
σε Προγραμματιστικό Περιβάλλον**

**Τόμος 2ος**

**Ομάδα Συγγραφής**

**ΑΘΗΝΑ ΒΑΚΑΛΗ,   
Λέκτωρ Πληροφορικής ΑΠΘ**

**ΗΛΙΑΣ ΓΙΑΝΝΟΠΟΥΛΟΣ,   
Μηχανικός Πληροφορικής**

**ΝΕΣΤΩΡ ΙΩΑΝΝΙΔΗΣ,   
Επίκουρος Καθηγητής Τμήματος**

**ΧΡΗΣΤΟΣ ΚΟΙΛΙΑΣ,   
Επίκουρος Καθηγητής Τμήματος Πληροφορικής ΤΕΙ Αθήνας**

**ΚΩΝ/ΝΟΣ ΜΑΛΑΜΑΣ, M.Sc. Πληροφορικής, Σύμβουλος Επιχειρήσεων,**

**ΙΩΑΝΝΗΣ ΜΑΝΩΛΟΠΟΥΛΟΣ, Αναπληρωτής Καθηγητής Τμήματος Πληροφορικής ΑΠΘ**

**ΠΑΝΑΓΙΩΤΗΣ ΠΟΛΙΤΗΣ,   
Δρ. Διδακτικής Πληροφορικής, Καθηγητής ΠΕ 19**

**Υπεύθυνος για το Παιδαγωγικό Ινστιτούτο  
ΓΕΩΡΓΙΟΣ ΠΑΠΑΔΟΠΟΥΛΟΣ, ΣΥΜΒΟΥΛΟΣ Π.Ι. (κατά τη συγγραφή)**

**Υπεύθυνος Μαθήματος  
ΑΔΑΜ Κ. ΑΓΓΕΛΗΣ,   
Πάρεδρος Πληροφορικής Π.Ι.**

**Επιτροπή Αξιολόγησης  
ΚΩΝ/ΝΟΣ ΓΙΑΛΟΥΡΗΣ,   
Καθηγητής ΠΕ 19   
ΕΥΑΓΓΕΛΙΑ ΓΟΥΛΗ,   
Καθηγήτρια ΠΕ 19   
ΔΗΜΗΤΡΙΟΣ ΔΕΣΠΟΤΗΣ, Αναπληρωτής Καθηγητής Πανεπιστημίου Πειραιώς   
ΚΩΝ/ΝΟΣ ΖΑΧΑΡΗΣ,   
ΠΛΗΝΕΤ Καρδίτσας   
ΑΛΕΞΑΝΔΡΟΣ ΣΙΔΕΡΙΔΗΣ, Καθηγητής Γεωπονικού Πανεπιστημίου Αθηνών**

**Εικονογράφηση   
ΑΓΓΕΛΟΣ ΑΓΙΟΣΤΡΑΤΙΤΗΣ**

**Ηλεκτρονική σελιδοποίηση   
ΑΝΝΑ ΑΡΓΥΡΟΠΟΥΛΟΥ - ΧΡΗΣΤΟΣ ΠΙΓΚΑΣ**

**Εξώφυλλο:   
ΣΠΥΡΟΣ ΣΙΑΚΑΣ - ΝΙΚΟΣ ΓΕΩΡΓΙΟΥ**

**Επιμέλεια:   
ΕΚΔΟΣΕΙΣ ΝΕΩΝ ΤΕΧΝΟΛΟΓΙΩΝ, Στουρνάρη 49Α, 106 82, Αθήνα, Τηλ. 38.45.594**

**Φορέας:**  
**ΕΛΛΗΝΙΚΗ ΕΤΑΙΡΙΑ ΕΠΙΣΤΗΜΟΝΩΝ Η/Υ ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ (ΕΠΥ), Μαυρομιχάλη 16, Αθήνα,   
τηλ.: 3645274, e.mail:epy@epy.gr**

**Συντονιστές έργου:**

**Σπ. Μπακογιάννης, πρόεδρος Δ.σ. Βασ. Μιχαλακόπουλος, μέλος Δ.Σ.**

**Ενέργεια 1.1.α: «Προγράμματα Βιβλία»**

**Επιστημονικός Υπεύθυνος Ενέργειας**

**Θεόδωρος Γ. Εξαρχάκος,**

**Καθηγητής του Πανεπιστημίου**

**Αθηνών, Πρόεδρος του Παιδαγωγικού Ινστιτούτου**

**Έργο No 15: «Αναμόρφωση / εκ νέου σύνταξη και συγγραφή Προγραμμάτων Σπουδών και Σχολικών Βιβλίων για το Ενιαίο Λύκειο»**

**Επιστημονικός Υπεύθυνος Έργου**

**Γιάννης Σαλβαράς,**

**Επίκουρος Καθηγητής του**

**Δημοκρίτειου Πανεπιστημίου Θράκης**

**Καλλιτεχνικός Υπεύθυνος Έργου**

**Σπύρος I. Παπασπύρου**

**Καθηγητής Εφαρμογών του ΤΕΙ Ηπείρου**

**Η επανέκδοση του παρόντος βιβλίου πραγματοποιήθηκε από το Ινστιτούτο Τεχνολογίας Υπολογιστών & Εκδόσεων «Διόφαντος» μέσω ψηφιακής μακέτας, η οποία δημιουργήθηκε με χρηματοδότηση από το ΕΣΠΑ / ΕΠ «Εκπαίδευση & Διά Βίου Μάθηση» / Πράξη «ΣΤΗΡΙΖΩ».**

**Οι αλλαγές που ενσωματώθηκαν στην παρούσα επανέκδοση έγιναν με βάση τις διορθώσεις του Παιδαγωγικού Ινστιτούτου**

**ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ INΣΤΙΤΟΥΤΟ ΕΚΠΑΙΔΕΥΤΙΚΗΣ ΠΟΛΙΤΙΚΗΣ**

**Α. ΒΑΚΑΛΗ, Η. ΓΙΑΝΝΟΠΟΥΛΟΣ, Ν. ΙΩΑΝΝΙΔΗΣ, Χ. ΚΟΙΛΙΑΣ, Κ. ΜΑΛΑΜΑΣ, Ι. ΜΑΝΩΛΟΠΟΥΛΟΣ, Π. ΠΟΛΙΤΗΣ**

**Η συγγραφή και η επιμέλεια του βιβλίου πραγματοποιήθηκε υπό την αιγίδα του Παιδαγωγικού Ινστιτούτου**

**Ανάπτυξη Εφαρμογών**

**σε Προγραμματιστικό Περιβάλλον**

**Βιβλίο Μαθητή**

**Γ' Γενικού Λυκείου**

**(Τεχνολογικής Κατεύθυνσης)**

**ΙΝΣΤΙΤΟΥΤΟ ΤΕΧΝΟΛΟΓΙΑΣ ΥΠΟΛΟΓΙΣΤΩΝ ΚΑΙ   
ΕΚΔΟΣΕΩΝ «ΔΙΟΦΑΝΤΟΣ»**

**ΠΡΟΣΑΡΜΟΓΗ ΤΟΥ ΒΙΒΛΙΟΥ ΓΙΑ ΜΑΘΗΤΕΣ ΜΕ ΜΕΙΩΜΕΝΗ ΟΡΑΣΗ**

**Ομάδα εργασίας για το Ινστιτούτου Εκπαιδευτικής**

**Πολιτικής**

**Προσαρμογή: Μαρία Γκλεζάκου, Εκπαιδευτικός**

**Eπιμέλεια: Αρετή Ανδριώτου, Εκπαιδευτικός**

**Επιστημονικός υπεύθυνος: Βασίλης Κουρμπέτης,**

**Σύμβουλος Α΄ του Υ.ΠΟ.ΠΑΙ.Θ**

**Υπεύθυνη του έργου: Μαρία Γελαστοπούλου,**

**M.Ed. Ειδικής Αγωγής**

**Τεχνική υποστήριξη: Κωνσταντίνος Γκυρτής,**

**Δρ. Πληροφορικής**

**Κεφάλαιο 6**

**Εισαγωγή στον**

**Προγραμματισμό**

**ΠΕΡΙΕΧΟΜΕΝΑ**

|  |  |
| --- | --- |
|  | **Η έννοια του προγράμματος** |
|  | **Ιστορική αναδρομή** |
|  | **Φυσικές και τεχνητές γλώσσες** |
|  | **Τεχνικές σχεδίασης προγραμμάτων** |
|  | **Αντικειμενοστραφής προγραμματισμός** |
|  | **Παράλληλος προγραμματισμός** |
|  | **Προγραμματιστικά περιβάλλοντα**  **5 / 103** |

**• Εισαγωγή**

**Στα προηγούμενα κεφάλαια αναφερθήκαμε αναλυτικά στην ανάπτυξη των αλγορίθμων και των διαφόρων τεχνικών. Στα επόμενα κεφάλαια θα ασχοληθούμε με τον προγραμματισμό δηλαδή τη διατύπωση των αλγορίθμων σε τέτοια μορφή, ώστε να μπορούν να υλοποιηθούν από τον υπολογιστή. Το κεφάλαιο αυτό ασχολείται με γενικές έννοιες που είναι απαραίτητες πριν από την ενασχόληση με τη διαδικασία του προ-γραμματισμού. Ορίζεται η έννοια του προγράμματος, παρουσιάζεται μία σύντομη ιστορία των γλωσσών προγραμματισμού και των ειδών προγραμματισμού, καθώς και οι τεχνικές που χρησιμοποιούνται για τη σωστή δημιουργία προγραμμάτων. Συγκεκριμένα παρουσιάζονται οι τεχνικές της ιεραρχικής σχεδίασης προγραμμάτων, του τμηματικού προγραμματισμού και κύρια οι αρχές του δομημένου προγραμματισμού. Επίσης περιγράφεται η διαδικασία που ακολουθείται από τη σύνταξη του προγράμματος μέχρι την τελική του εκτέλεση από τον υπολογιστή και τη λήψη των αποτε-λεσμάτων.**

**6 / 104**

**• Διδακτικοί στόχοι**

**Στόχοι του κεφαλαίου αυτού είναι ο μαθητής:**

* **Να ορίζει τι είναι πρόγραμμα και να κατατάσσει και να συγκρίνει τις γλώσσες προγραμματισμού.**
* **Να αναγνωρίζει τα κυριότερα είδη προγραμμα-τισμού και να περιγράφει τα βασικά χαρακτηρι-στικά των τεχνικών που χρησιμοποιούνται.**
* **Να διατυπώνει τα πλεονεκτήματα του δομημένου προγραμματισμού.**
* **Να περιγράφει τη διαδικασία εκτέλεσης ενός προγράμματος.**
* **Να αναφέρει τα βασικά προγράμματα που περιέχει ένα προγραμματιστικό περιβάλλον.**

**• Προερωτήσεις**

* **Η δημιουργία του αλγορίθμου αρκεί για να επιλύσουμε ένα πρόβλημα στον υπολογιστή;**
* **Πώς διαχειρίζεται τις πληροφορίες ο υπολογιστής;**
* **Γνωρίζεις κάποιες γλώσσες προγραμματισμού;**
* **Πώς και γατί εξελίσσονται οι γλώσσες;**
* **Η ύπαρξη συγκεκριμένων μεθοδολογιών και τεχνικών βοηθάει στην επίλυση των προβλημάτων;**

**7 / 104**

|  |  |
| --- | --- |
| **6.1** | **Η έννοια του προγράμματος** |

**Η επίλυση ενός προβλήματος με τον υπολογιστή περιλαμβάνει, όπως έχει ήδη αναφερθεί, τρία εξίσου 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.**

**Ο προγραμματισμός ασχολείται με το τρίτο αυτό στάδιο, τη δημιουργία του προγράμματος δηλαδή του συνόλου των εντολών που πρέπει να δοθούν στον υπολογιστή, ώστε να υλοποιηθεί ο αλγόριθμος για την επίλυση του προβλήματος. Το πρόγραμμα, το οποίο γράφεται σε κάποια γλώσσα προγραμματισμού, δεν είναι απλά η υλοποίηση του αλγορίθμου, αλλά βασικό στοιχείο του είναι τα δεδομένα και οι δομές δεδομένων επί των οποίων ενεργεί. Αναφέρθηκε ήδη ότι οι αλγό-ριθμοι και οι δομές δεδομένων είναι μια αδιάσπαστη ενότητα.**

**8 / 105**

|  |  |
| --- | --- |
| **πιν.jpg** | **Οι γλώσσες προγραμματισμού ανα-πτύχθηκαν με σκοπό την επικοινωνία του ανθρώπου (προγραμματιστή) με τη μηχανή (υπολογιστή).** |

**Ο προγραμματισμός είναι αυτός που δίνει την εντύ-πωση ότι, οι υπολογιστές είναι έξυπνες μηχανές που επιλύουν τα πολύπλοκα προβλήματα. Η εντύπωση αυτή όμως είναι απλώς μία ψευδαίσθηση. Ο υπολο-γιστής, ως γνωστόν, είναι μία μηχανή που καταλαβαίνει μόνο δύο καταστάσεις, οι οποίες αντιπροσωπεύονται με δύο αριθμούς το μηδέν και το ένα, τα ψηφία του δυαδικού συστήματος. Το μόνο πράγμα που κάνει ο υπολογιστής είναι στοιχειώδεις ενέργειες σε ακολου-θίες αυτών των δύο ψηφίων, αλλά αυτές τις ενέργειες τις εκτελεί μα ασύλληπτη ταχύτητα. Ο υπολογιστής μπορεί απλά να αποθηκεύει στη μνήμη τις ακολουθίες των δυαδικών ψηφίων, να τις ανακτά, να κάνει στοι-χειώδεις αριθμητικές πράξεις με αυτές και να τις συγκρίνει.**

|  |  |
| --- | --- |
| **6.2** | **Ιστορική αναδρομή** |

**Από τη δημιουργία του πρώτου υπολογιστή μέχρι σήμερα έχουν αλλάξει πάρα πολλά πράγματα. Οι πρώτοι υπολογιστές, τεράστιοι σε μέγεθος αλλά με πάρα πολύ περιορισμένες δυνατότητες και μικρές ταχύτητας επεξεργασίας εξελίχθηκαν σε πολύ μικρούς σε μέγεθος υπολογιστές με τεράστιες όμως δυνατό-τητες και ταχύτητες επεξεργασίας.**

**Ενώ λοιπόν το υλικό (hardware) των υπολογιστών βελτιώνεται, τελειοποιείται και ταυτόχρονα παρέχει νέες δυνατότητες επεξεργασίας, οι βασικές αρχές λειτουρ-γίας των υπολογιστών που διατυπώθηκαν το μακρινό 1945 από τον Φον Νόυμαν, δεν άλλαξαν πρακτικά καθό-λου. Την ίδια αργή εξέλιξη ουσιαστικά έχουν και οι γλώσσες προγραμματισμού, οι οποίες αν και εξελίσ-σονται και συνεχώς εμπλουτίζονται με νέες δυνατό-τητες, τα χαρακτηριστικά τους και οι βασικές τους ιδιότητες ουσιαστικά παραμένουν τα ίδια.**

**9 / 105**

**ΟΡΙΣΜΟΣ**

**Ένα πρόγραμμα σε γλώσσα μηχανής είναι μια ακο-λουθία δυαδικών ψηφίων, που αποτελούν εντολές προς τον επεξεργαστή για στοιχειώδεις λειτουργίες.**

**6.2.1 Γλώσσες μηχανής**

**Αρχικά για να μπορέσει ο υπολογιστής να εκτελέσει μία οποιαδήποτε λειτουργία, έπρεπε να δοθούν κατευθείαν οι κατάλληλες ακολουθίες από 0 και 1, δηλαδή εντολές σε μορφή κατανοητή από τον υπολογιστή αλλά ακατα-νόητες από τον άνθρωπο. Ο τρόπος αυτός ήταν επίπο-νος και ελάχιστοι μπορούσαν να τον υλοποιήσουν, αφού απαιτούσε βαθιά γνώση του υλικού και της αρχιτεκτονικής του υπολογιστή.**

**Ο πρώτος υπολογιστής ο περίφημος ENIAC για να "προγραμματιστεί", ώστε να εκτελέσει κάποιους υπολογισμούς, έπρεπε να αλλάξουν θέση εκατοντάδες διακόπτες και να ρυθμιστούν αντίστοιχα όλες οι καλω-διώσεις, διαδικασία εξαιρετικά επίπονη και χρονοβόρα. Ο "προγραμματισμός" των πρώτων αυτών υπολο-γιστών, δεν ήταν ουσιαστικά προγραμματισμός με τη σημερινή έννοια του όρου. Ο υπολογιστής αναδι-αρθρωνόταν, ώστε να εκτελέσει τους απαιτούμενους υπολογισμούς και στη συνέχεια έπρεπε να αλλάξει πάλι η διάρθρωσή του, ώστε να εκτελέσει έναν άλλο υπολογισμό.**

**Οι εντολές ενός προγράμματος και σήμερα μετατρέ-πονται σε ακολουθίες που αποτελούνται από 0 και 1, τις εντολές σε γλώσσα μηχανής, όπως ονομάζονται, οι οποίες εκτελούνται από τον υπολογιστή.**

**10 / 106**

**6.2.2 Συμβολικές γλώσσες ή γλώσσες χαμηλού επιπέδου**

**Από τα πρώτα χρόνια άρχισαν να γίνονται προσπά-θειες για τη δημιουργία μίας συμβολικής γλώσσας, η οποία ενώ θα έχει έννοια για τον άνθρωπο, θα μετατρέπεται εσωτερικά από τους υπολογιστές στις αντίστοιχες ακολουθίες από 0 και 1. Για παράδειγμα η λέξη ADD (πρόσθεσε) ακολουθούμενη από δύο αριθμούς, είναι κατανοητή από τον άνθρωπο και απομνημονεύεται σχετικά εύκολα. Η εντολή αυτή θα μεταφραστεί από τον υπολογιστή σε μία ακολουθία δυαδικών ψηφίων και στη συνέχεια μπορεί να εκτελεστεί. Το έργο της μετάφρασης το αναλαμβάνει ένα ειδικό πρόγραμμα, ο συμβολομεταφραστής (assembler).**

**Η χρήση των πρώτων αυτών συμβολικών γλωσσών, που συνεχίζουν να χρησιμοποιούνται για ειδικούς σκοπούς, ήταν σαφώς μια εξέλιξη από τις ακατανόητες ακολουθίες δυαδικών στοιχείων. Ωστόσο παρέμεναν στενά συνδεδεμένες με την αρχιτεκτονική του κάθε υπολογιστή. Επίσης δεν διέθεταν εντολές πιο σύνθετων λειτουργιών οδηγώντας έτσι σε μακροσκελή προ-γράμματα, που ήταν δύσκολο να γραφούν και κύρια να συντηρηθούν. Ακόμη τα προγράμματα δεν μπορούν να μεταφερθούν σε άλλον διαφορετικό υπολογιστή, ακόμη και του ίδιου κατασκευαστή. Οι γλώσσες αυτές ονομά-ζονται συμβολικές ή γλώσσες χαμηλού επιπέδου, αφού εξαρτώνται από την αρχιτεκτονική του υπολογιστή.**

**11 / 106**

|  |  |
| --- | --- |
| **πιν.jpg** | **Οι εντολές σε συμβολική γλώσσα αποτελούνται από συμβολικά ονό-ματα που αντιστοιχούν σε εντολές της γλώσσας μηχανής. Οι συμβολικές γλώσσες είναι συνδεδεμένες με την αρχιτεκτονική κάθε υπολογιστή.** |

**12 / 106**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **10101000**  **10101100**  **00111100**  **01010001**  **01010011**  **11000000**  **10001100**  **11111111** | **0001010**  **00000001**  **00000001**  **00000001**  **11111010**  **00000010** | **LOOP** | **INDEX=$01**  **SUM=$02**  **LDA#10**  **STA INDEX**  **CLA**  **ADD INDEX**  **DEC INDEX**  **BNE LOOP**  **STA SUM**  **BRK** | **sum=0**  **FOR index=1 TO 10**  **sum=sum+index**  **NEXT index**  **END** |
| **Σχ. 6.1** **Πρόγραμμα σε γλώσσα μηχανής, σε συμβολική γλώσσα και σε γλώσσα υψηλού επιπέδου για τον υπολογισμό του αθροίσματος των αριθμών 1 έως 10**  **13 / 107** | | | | |

**6.2.3 Γλώσσες υψηλού επιπέδου**

**Οι παραπάνω ανεπάρκειες των συμβολικών γλωσσών και η προσπάθεια για καλύτερη επικοινωνία ανθρώ-που-μηχανής, οδήγησαν στα τέλη της δεκαετίας του 50 στην εμφάνιση των πρώτων γλωσσών προγραμμα-τισμού υψηλού επιπέδου.**

**Το 1957 η IBM ανάπτυξε την πρώτη γλώσσα υψηλού επιπέδου τη FORTRAN. Το όνομα FORTRAN προ-έρχεται από τις λέξεις FORmula TRANslation, που σημαίνουν μετάφραση τύπων. Η FORTRAN αναπτύχθηκε ως γλώσσα κατάλληλη για την επίλυση μαθηματικών και επιστημονικών προβλημάτων. Το πρόγραμμα που γράφεται σε FORTRAN ή σε οποι-αδήποτε άλλη γλώσσα υψηλού επιπέδου, μεταφράζεται από τον ίδιο τον υπολογιστή στις ακολουθίες των εντολών της μηχανής με τη βοήθεια ενός ειδικού προγράμματος, που ονομάζεται μεταγλωττιστής. Το ίδιο πρόγραμμα FORTRAN μπορεί να εκτελεστεί σε οποιοδήποτε άλλο υπολογιστή, αρκεί να υπάρχει ο αντίστοιχος μεταγλωττιστής για τον υπολογιστή αυτό. Η γλώσσα FORTRAN μετά από πολλές αλλαγές, προσθήκες και βελτιώσεις χρησιμοποιείται ακόμη και σήμερα για επιστημονικές εφαρμογές.**

**14 / 107**

|  |  |
| --- | --- |
| **C** | **PROGRAM EQUATION** |
|  | **READ (\*, 1) A, B** |
| **1** | **FORMAT (F5.1)** |
|  | **IF (A.EQ.0) GO TO 20** |
|  | **X= (-1.) \*B/A** |
|  | **WRITE (\*,2) X** |
| **2** | **FORMAT (`X=΄, F10.2)** |
|  | **GO TO 50** |
| **20** | **IF (B. EQ. 0) WRITE (\*, 3)** |
|  | **IF (B. EQ. 0) WRITE (\*, 3)** |
| **3** | **FORMAT (`ΑΟΡΙΣΤΗ΄)** |
| **4** | **FORMAT (`ΑΔΥΝΑΤΗ΄)** |
| **50** | **STOP** |
|  | **END** |
| **Σχ.6.2. Η γλώσσα FORTRAN υπήρξε η πρώτη γλώσσα προγραμματισμού υψηλού επιπέδου. Πρόκειται για γλώσσα κατάλληλη για υπολογισμούς, ενώ υστερεί στη διαχείριση αρχείων δεδομένων και γενικότερα αλφαριθμητικών πληροφοριών. Γνώρισε πολλές βελτιώσεις με κυριότερους σταθμούς τις εκδόσεις 4, 77, 90/95 και Visual FORTRAN. Το πρόγραμμα του παραδείγματος επιλύει την εξίσωση α' βαθμού.**  **15 / 107** | |

**Η FORTRAN παρά τα ισχυρά χαρακτηριστικά της και τις συνεχείς αλλαγές που την καθιστούσαν συνεχώς αποτελεσματικότερη, δεν μπορούσε να καλύψει τις απαιτήσεις σε όλους τους τομείς δραστηριοτήτων, όπως και καμία άλλη γλώσσα προγραμματισμού δεν κατάφερε. Έτσι αναπτύχθηκαν και συνεχίζουν να αναπτύσσονται πολλές γλώσσες προγραμματισμού για διάφορες περιοχές δραστηριοτήτων.**

**Το 1960 αναπτύχθηκε μία άλλη γλώσσα, σταθμός στον προγραμματισμό η γλώσσα COBOL. Η COBOL όπως δηλώνει και το όνομα της (COmmon Business Oriented Language - Κοινή γλώσσα προσανατολισμένη στις επιχειρήσεις) είναι κατάλληλη για ανάπτυξη εμπορικών εφαρμογών, και γενικότερα διαχειριστικών εφαρμογών, τομέας όπου η FORTRAN υστερούσε. Η COBOL καθι-ερώθηκε ως πρότυπο και χρησιμοποιήθηκε από πολλές επιχειρήσεις καθώς και από όλη τη δημόσια διοίκηση. Η γλώσσα γνώρισε πολλές εκδόσεις και πάρα πολλές εφαρμογές βρίσκονται σε χρήση ακόμη και σήμερα.**

**Μια από τις σημαντικότερες γλώσσες προγραμμα-τισμού με ελάχιστη πρακτική εφαρμογή αλλά που επηρέασε ιδιαίτερα τον προγραμματισμό και τις επόμενες γλώσσες, είναι η ALGOL (ALGOrithmic Language - Αλγοριθμική γλώσσα). Αναπτύχθηκε από Ευρωπαίους επιστήμονες, αρχικά το 1960, με σκοπό τη δημιουργία γενικής φύσης προγραμμάτων που να μη συνδέονται με συγκεκριμένες εφαρμογές.**

**16 / 108 & 109**

**IDENTIFICATION DIVISION.**

**PROGRAM-ID. EQUATION.**

**ENVIRONMENT DIVISION.**

**CONFIGURATION SECTION.**

**SOURCE-COMPUTER. IBM-PC.**

**OBJECT-COMPUTER. IBM-PC.**

**SPECIAL-NAMES DECIMAL-POINT IS COMMA.**

**DATA DIVISION.**

**WORKING-STORAGE SECTION.**

**77 X PIC S9(6)V9.**

**77 A PIC S9(6).**

**77 B PIC S9(6).**

**77 W-X PIC -(6),-.**

**PROCEDURE DIVISION.**

**ARXH.**

**DISPLAY ‘ ΔΩΣΕ A’.**

**ACCEPT B.**

**DISPLAY ‘ ΔΩΣΕ B’.**

**ACCEPT B.**

**DISPLAY ‘ ’.**

**IF A = 0 GO TO ROYT-1.**

**COMPUTE X = B \* ( - 1 ) / A.**

**MOVE X TO W-X.**

**DISPLAY ‘ Η ΛΥΣΗ ΕΙΝΑΙ : ’ W-X.**

**STOP RUN.**

**ROYT-1.**

**IF B = 0**

**DISPLAY ‘ ΑΟΡΙΣΤΗ’**

**ELSE**

**DISPLAY ‘ ΑΔΥΝΑΤΗ’.**

**17 / 108**

**STOP RUN**

**Σχ. 6.3. Η γλώσσα COBOL δημιουργήθηκε από την Grace Marray Hopper αξιωματικό του πολεμικού ναυτικού των ΗΠΑ το 1960. Η COBOL έκανε δυνατή   
την αξιοποίηση των υπολογιστών από τις επιχειρήσεις και τους οργανισμούς παρέχοντας ισχυρότατες δυνατότητες διαχείρισης αρχείων δεδομένων. Ένα πρόγραμμα COBOL διαθέτει τέσσερις υποδιαιρέσεις (divisions). Η γλώσσα χρησιμοποιεί περιγραφικό τρόπο για τη σύνταξη των εντολών με χρήση ρημάτων της αγγλικής γλώσσας, όπως ADD, MULTIPLY, MOVE κ.λπ. με αποτέλεσμα τη δημιουργία γενικά μακροσκελών προγραμμάτων. Κάθε εντολή της γλώσσας τερματίζεται με τελεία. Το πρόγραμμα του παραδείγματος επιλύει την εξίσωση α' βαθμού.**

**18 / 108**

**10 REM ΕΠΙΛΥΣΗ ΕΞΙΣΩΣΗΣ Α’ ΒΑΘΜΟΥ**

**20 INPUT “A=”, A**

**30 INPUT “B=”, B**

**40 IF A=0 THEN 100**

**50 X= -B/A**

**60 PRINT “X=”; X**

**70 END**

**100 IF B=0 THEN PRINT “ΑΟΡΙΣΤΗ”**

**ELSE PRINT “ΑΔΥΝΑΤΗ”**

**110 END**

|  |
| --- |
| **Σχ. 6.4. Η γλώσσα BASIC δημιουργήθηκε το 1964 στο Dartmouth College από τους καθηγητές Kemeny και Kurtz. Στόχος των δημιουργών της ήταν η υλοποίηση μιας απλής γλώσσας προγραμματισμού για εκπαιδευτικούς σκοπούς. Η BASIC γνώρισε εκπληκτική διάδοση με την εμφάνιση των μικροϋπολογιστών (1975). Ο πρώτος διερμηνευτής της γλώσσας σε 8Κ ROM ήταν έργο των Bill Gates και Paul Allen. Ας σημειωθεί ότι, η εταιρία Microsoft ιδρύθηκε από τον Bill Gates για την εμπορική εκμετάλλευση αυτού του διερμηνευτή. Η έκδοση GWBASIC παρουσιάζεται το 1981 με τον IBM PC. Κύριο χαρακτηριστικά της γλώσσας είναι η ύπαρξη αριθμών των γραμμών του προγράμματος, οι οποίοι χρησιμοποιούνται και ως θέσεις προορισμού εντολών διακλάδωσης. Εξέλιξη της γλώσσας είναι οι εκδόσεις QuickBASIC και Visual BASIC. Το πρόγραμμα του παραδείγματος επιλύει την εξίσωση α' βαθμού.**  **19 / 109** |

**Στα μέσα της δεκαετίας του 60 αναπτύχθηκε η γλώσσα PL/1 (Programming Language/1 - Γλώσσα Προγραμ-ματισμού υπ' αριθμόν 1) που προσπάθησε, χωρίς επιτυχία να καλύψει όλους τους τομείς του προ-γραμματισμού, επιστημονικούς και εμπορικούς, αντικαθιστώντας τόσο τη FORTRAN όσο και την COBOL.**

**Στο χώρο της Τεχνητής Νοημοσύνης αναπτύχθηκαν δύο γλώσσες αρκετά διαφορετικές από όλες τις άλλες. Στα μέσα του 60 αναπτύχθηκε στο ΜΙΤ η LISP (LISt Processor- Επεξεργαστής Λίστας), γλώσσα η οποία προσανατολίζεται σε χειρισμό λιστών από σύμβολα και η PROLOG (PROgramming LOGic –Λογικός Προ-γραμματισμός) στις αρχές του 70. Οι δύο αυτές γλώσ-σες χρησιμοποιούνται σε προβλήματα Τεχνητής νοημοσύνης (έμπειρα συστήματα, παιγνίδια, επεξεργασία φυσικών γλωσσών κ.λπ.).**

**Δύο σημαντικότατες γλώσσες γενικού σκοπού, οι οποίες αναπτύχθηκαν τη δεκαετία του 60 αλλά χρησιμοποιούνται πάρα πολύ στις ημέρες μας, είναι η BASIC και η PASCAL.**

**20 / 109**

**TO KYBOS :A**

**REPEAT 4 [ FD :A RT 90 ]**

**PU SETPOS [ 20, 20 ] PD**

**REPEAT 4 [ FD :A RT 90 ]**

**PU HOME PD**

**REPEAT 2 [ FD :A RT 45 FD 29 RT 135 ]**

**PU SETX :A SETY 0 PD**

**REPEAT 2 [ FD :A RT 45 FD 29 RT 135 ]**

**HOME**

**END**

|  |
| --- |
| **Σχ. 6.5. Η γλώσσα προγραμματισμού LOGO ολοκλη-ρώθηκε το 1967 στη Βοστώνη από τον Seymoyr Papert. Το όνομά της προέρχεται από την ελληνική λέξη "λόγος". Πρόκειται για γλώσσα κατάλληλη για την εισαγωγή στον προγραμματισμό μαθητών μικρής ηλικίας. Το πρόγραμμα του παραδείγματος σχεδιάζει έναν κύβο ακμής Α με διαδοχικές κινήσεις της χελώνας.** |

**Η γλώσσα προγραμματισμού BASIC (Beginner's All Purpose Symbolic Instruction Code - Συμβολικός Κώδικας Εντολών Γενικής Χρήσης για Αρχάριους) αρχικά αναπτύχθηκε, όπως δηλώνει και το όνομά της, ως γλώσσα για την εκπαίδευση αρχαρίων στον προγραμματισμό. Σχεδιάστηκε για να γράφονται σύντομα προγράμματα, τα οποία εκτελούνται με τη βοήθεια διερμηνευτή (interpreter). Η ανάπτυξη όμως των μικροϋπολογιστών και οι συνεχείς εκδόσεις της γλώσσας βοήθησαν στην εξάπλωσή της, τόσο ώστε να γίνει ίσως η δημοφιλέστερη γλώσσα στους προσω-πικούς υπολογιστές. Η τυποποίηση της δε από τη Microsoft με τις εκδόσεις QuickBasic και κύρια με τη Visual Basic, καθιέρωσε τη γλώσσα ως πρότυπο για ανάπτυξη εφαρμογών σε προσωπικούς υπολογιστές.**

**21 / 109-110**

**Η γλώσσα PASCAL (δημιούργημα του καθηγητή Niklaus Wirth) έφερε μεγάλες αλλαγές στον προγραμματισμό. Παρουσιάστηκε το 1970 και στηρίχτηκε πάνω στην ALGOL. Είναι μία γλώσσα γενικής χρήσης, η οποία είναι κατάλληλη τόσο για την εκπαίδευση όσο και τη δημιουργία ισχυρών προγραμμάτων κάθε τύπου. Χαρακτηριστικό της γλώσσας είναι η καταλληλότητα για τη δημιουργία δομημένων προγραμμάτων. Η PASCAL γνώρισε και συνεχίζει να γνωρίζει τεράστια εξάπλωση ειδικά στο χώρο των μικροϋπολογιστών και αποτέλεσε τη βάση για την ανάπτυξη άλλων ισχυ-ρότερων γλωσσών όπως η ADA και η Modula-2.**

**22 / 110**

**Στα μέσα του 1960 παρουσιάστηκε για πρώτη φορά μία τεχνική σχεδίασης προγραμμάτων που έμελλε να αλλάξει ριζικά τον τρόπο ανάπτυξης προγραμμάτων καθώς και τις ίδιες τις γλώσσες προγραμματισμού. Η τεχνική του δομημένου προγραμματισμού η οποία εξασφαλίζει τη δημιουργία προγραμμάτων απλών στη συγγραφή και την κατανόηση και εύκολων στη διόρθωση. Ο δομημένος προγραμματισμός και τα χαρακτηριστικά του θα παρουσιαστούν εκτενώς σε επόμενη παράγραφο.**

**(DEFUN a-exisosi (a b)**

**(setf apot ( - ( / b a ) ) )**

**(princ “Η εξίσωση ”)**

**(princ a)**

**(princ “x + ”)**

**(princ b)**

**(princ “ = 0 έχει σαν λύση x = ”)**

**(princ apot) )**

**Σχ. 6.6. Η γλώσσα LISP δημιουργήθηκε το 1959 στο ΜΙΤ. Πρόκειται για μη διαδικασιακή γλώσσα που προ-ορίζεται για την επεξεργασία συμβολικών δεδομένων. Βασικός τύπος δεδομένων, από τον οποίο εξ άλλου πήρε και το όνομά της, είναι η συνδεδεμένη λίστα. Στο παρά-δειγμα φαίνεται μια συνάρτηση της γλώσσας, που επι-λύει την εξίσωση α' βαθμού. Το πρόγραμμα εκτελείται δίνοντας στη γραμμή εντολών π.χ. (a-exisosi 2 5).**

**Μία ακόμη γλώσσα που γνώρισε μεγάλη διάδοση είναι η γλώσσα C. Η C αναπτύχθηκε στα εργαστήρια της εταιρείας BELL και χρησιμοποιήθηκε για την ανάπτυξη του λειτουργικού συστήματος Unix, γλώσσα με ισχυρά χαρακτηριστικά, μερικά από αυτά κοινά με την Pascal κατάλληλη για ανάπτυξη δομημένων εφαρμογών αλλά και με πολλές δυνατότητες γλώσσας χαμηλού επιπέδου. Η C εξελίχτηκε στη γλώσσα C++, που είναι αντικειμενοστραφής. Η ιδέα του αντικειμενοστραφούς προγραμματισμού παρουσιάστηκε για πρώτη φορά στη δεκαετία του 70 και συνεχίζει ακόμη να απλώνεται αλλάζοντας τον παραδοσιακό προγραμματισμό. Λόγω της σημασίας του αντικειμενοστραφούς προγραμματισμού μερικά στοιχεία του παρουσιάζονται σε ξεχωριστή παράγραφο.**

**23 / 110-111**

**Τα τελευταία χρόνια χρησιμοποιείται ιδιαίτερα, ειδικά για προγραμματισμό στο Διαδίκτυο (Internet), η JAVA. Η JAVA είναι μία αντικειμενοστραφής γλώσσα που αναπτύχθηκε από την εταιρεία SUN με σκοπό την ανάπτυξη εφαρμογών, που θα εκτελούνται σε κατά-νεμημένα περιβάλλοντα, δηλαδή σε διαφορετικούς υπολογιστές οι οποίοι είναι συνδεδεμένοι στο Διαδί-κτυο. Τα προγράμματα αυτά μπορούν να εκτελούνται από διαφορετικούς υπολογιστές, προσωπικούς ή μεγάλα συστήματα με διαφορετικά λειτουργικά συστήματα χωρίς αλλαγές.**

**Η εμφάνιση των γραφικών περιβαλλόντων εργασίας δημιούργησε την ανάγκη για ανάπτυξη προγραμμάτων που να εκμεταλλεύονται τον γραφικό αυτό τρόπο επικοινωνίας χρήστη-υπολογιστή. Στα περισσότερα προγραμματιστικά περιβάλλοντα που υπήρχαν, ήταν πολύ δύσκολη έως αδύνατη η ανάπτυξη εφαρμογών, ικανών να εκμεταλλεύονται τα γραφικά αυτά χαρα-κτηριστικά.**

**Έτσι εμφανίστηκαν γλώσσες ή νέες εκδόσεις των γλωσσών που υλοποιούσαν τις έννοιες του οδηγού-μενου από το γεγονός προγραμματισμού (event driven programming) και του οπτικού προγραμματισμού (visual programming).**

**24 / 111-112**

**#include <stdio.h>**

**int main ( int argc, char\* argv[ ] )**

**{**

**float a, b;**

**printf(“A = ”);**

**scanf(“%f”, &a);**

**printf(“B = ”);**

**scanf(“%f”, &b);**

**if (a == 0) {**

**if (b == 0) {**

**printf(“ΑΟΡΙΣΤΗ\n”);**

**}**

**else {**

**printf(“ΑΔΥΝΑΤΗ\n”);**

**}**

**}**

**else {**

**printf(“X = %f\n”, -b/a);**

**}**

**return 0;**

**}**

**Σχ. 6.7. Η γλώσσα προγραμματισμού C αναπτύχθηκε στα εργαστήρια Bell της αμερικανικής εταιρίας Ατ&Τ το 1972 από τον Dennis Ritchie. Η γλώσσα C είναι μια δομημένη διαδικασιακή γλώσσα γενικής χρήσης που χαρακτηρίζεται από λιτότητα στην έκφραση και μια πλούσια συλλογή τελεστών και τύπων δεδομένων. Στενά συνδεδεμένη με το UNIX, η γλώσσα C χρησιμοποιείται ευρύτατα για τη δημιουργία λειτουργικών συστημάτων και άλλων πακέτων λογισμικού. Το πρόγραμμα του παραδείγματος επιλύει την εξίσωση α' βαθμού.**

**25 / 111**

**CLEAR**

**26 / 112**

**? “1. ΕΙΣΑΓΩΓΗ ΣΤΟΙΧΕΙΩΝ”**

**? “2. ΕΚΤΥΠΩΣΗ ΕΤΙΚΕΤΩΝ”**

**? “3. ΠΡΟΒΟΛΗ”**

**? “4. ΤΕΛΟΣ”**

**INPUT “Επιλέξτε [1..4] : ” TO CHOICE**

**DO CASE**

**CASE CHOICE=1**

**APPEND**

**CASE CHOICE=2**

**LABEL FORM PELATES**

**CASE CHOICE=3**

**BROWSE**

**OTHERWISE**

**QUIT**

**END CASE**

**Σχ. 6.8. Η dBASE παρουσιάστηκε στα τέλη της δεκαετίας του 70 από την εταιρία Ashton-Tate αρχικά για μικροϋπολογιστές 8- bit και αργότερα για προσω-πικούς υπολογιστές. Η dBASE υπήρξε ο σπουδαιότερος εκπρόσωπος εξελιγμένων γλωσσών και εργαλείων προγραμματισμού της εποχής με κύριο χαρακτηριστικό τις εξαιρετικές δυνατότητες διαχείρισης αρχείων (βάσεων) δεδομένων. Η dBASE μπορούσε να λειτουργεί με άμεση εκτέλεση των εντολών της μέσα από μια διαλογική διεπαφή χρήστη, με συνέπεια να μπορεί να χρησιμοποιηθεί και από άτομα με ελάχιστες γνώσεις προγραμματισμού. Το γεγονός αυτό συνέβαλε στην ευρεία διάδοση των προσωπικών υπολογιστών. Η πλέον διαδεδομένη έκδοση ήταν η dBASE III Plus. Εξέλιξή της υπήρξε ο Clipper μεταγλωττιζόμενη γλώσσα, με την οποία δημιουργήθηκαν πλήθος εμπορικών εφαρμογών σε προσωπικούς υπολογιστές. Τα προϊόντα αυτά, που συχνά επονομάζονται XBASE, υπήρξαν προάγγελοι των σημερινών πακέτων διαχείρισης βάσεων δεδομένων όπως η Access. Το απόσπασμα προγράμματος του παραδείγματος παρουσιάζει ένα μενού επιλογών και εκτελεί διακλάδωση στην αντίστοιχη λειτουργία.**

**Με τον όρο οπτικό εννοούμε τη δυνατότητα να δημιουργούμε γραφικά ολόκληρο το περιβάλλον της εφαρμογής για παράδειγμα τα πλαίσια διαλόγου ή τα μενού. Με τον όρο οδηγούμενο από το γεγονός εννοούμε τη δυνατότητα να ενεργοποιούνται λειτουργίες του προγράμματος με την εκτέλεση ενός γεγονότος, για παράδειγμα την επιλογή μίας εντολής από ένα μενού ή το κλικ του ποντικιού.**

**Οι πιο διαδεδομένες γλώσσες προγραμματισμού σε γραφικό περιβάλλον για προσωπικούς υπολογιστές είναι η Visual Basic, η Visual C+ + και η Java.**

**27 / 112**

**Πλεονεκτήματα των γλωσσών υψηλού επιπέδου**

**Στα πλεονεκτήματα των γλωσσών προγραμματισμού υψηλού επιπέδου σε σχέση με τις συμβολικές μπορούν να αναφερθούν:**

* **Ο φυσικότερος και πιο "ανθρώπινος" τρόπος έκφρασης των προβλημάτων. Τα προγράμματα σε γλώσσα υψηλού επιπέδου είναι πιο κοντά στα προβλήματα που επιλύουν.**
* **Η ανεξαρτησία από τον τύπο του υπολογιστή. Προγράμματα σε μία γλώσσα υψηλού επιπέδου μπορούν να εκτελεστούν σε οποιονδήποτε υπολογιστή με ελάχιστες ή καθόλου μετατροπές. Η δυνατότητα της μεταφερσιμότητας των προγραμμάτων είναι σημαντικό προσόν.**
* **Η ευκολία της εκμάθησης και εκπαίδευσης ως απόρροια των προηγουμένων.**
* **Η διόρθωση λαθών και η συντήρηση προγραμ-μάτων σε γλώσσα υψηλού επιπέδου είναι πολύ ευκολότερο έργο.**

**Συνολικά οι γλώσσες υψηλού επιπέδου ελάττωσαν σημαντικά το χρόνο και το κόστος παραγωγής νέων προγραμμάτων, αφού λιγότεροι προγραμματιστές μπορούν σε μικρότερο χρόνο να αναπτύξουν προ-γράμματα που χρησιμοποιούνται σε περισσότερους υπολογιστές.**

**28 / 113**

|  |  |
| --- | --- |
| **πιν.jpg** | **Οι γλώσσες υψηλού επιπέδου χρη-σιμοποιούν ως εντολές απλές λέξεις της αγγλικής γλώσσας ακολουθώντας αυστηρούς κανόνες σύνταξης, οι οποίες μεταφράζονται από τον ίδιο τον υπολογιστή σε εντολές σε γλώσσα μηχανής.** |

**29 / 113**

**6.2.4 Γλώσσες 4ης γενιάς**

**Οι γλώσσες υψηλού επιπέδου (γλώσσες 3ης γενιάς) γνώρισαν μεγάλη επιτυχία λόγω των πλεονεκτημάτων που παρουσιάζουν. Ωστόσο απευθύνονται μόνο σε προγραμματιστές. Ο χρήστης ενός υπολογιστή δεν είχε τη δυνατότητα να επιφέρει αλλαγές σε κάποιο πρό-γραμμα, προκειμένου να ικανοποιήσει μια νέα ανάγκη του. Σταδιακά όμως πολλές γλώσσες εφοδιάστηκαν με εργαλεία προγραμματισμού που αποκρύπτουν πολλές λεπτομέρειες από τις τεχνικές υλοποίησης και με αυτά ο χρήστης μπορεί να επιλύει μόνος του μικρά προ-βλήματα εφαρμογών. Αυτή η αυξανόμενη τάση απόκρυψης της αρχιτεκτονικής του υλικού και της τεχνικής του προγραμματισμού οδήγησε στις γλώσσες 4ης γενιάς.**

**Στις γλώσσες αυτές ο χρήστης ενός υπολογιστή έχει τη δυνατότητα, σχετικά εύκολα, να υποβάλει ερωτήσεις στο σύστημα ή να αναπτύσσει εφαρμογές που ανα-κτούν πληροφορίες από βάσεις δεδομένων και να καθορίζει τον ακριβή τρόπο εμφάνισης αυτών των πληροφοριών, όπως στο παράδειγμα που ακολουθεί.**

**SELECT ENAME, JOB, SAL**

**FROM EMPLOYEES**

**WHERE DEPTNO=20**

**AND SAL > 3000;**

**Η ερώτηση αυτή σε SQL εκτελεί αναζήτηση στη βάση δεδομένων EMPLOYEES και επιστρέφει το όνομα, τη θέση και το μισθό των υπαλλήλων της διεύθυνσης 20 που κερδίζουν πάνω από 3.000 €.**

**30 / 113**

**31 / 114**

**Ταξινόμηση γλωσσών προγραμματισμού**

**Όλες οι γλώσσες προγραμματισμού που έχουν αναπτυχθεί μέχρι σήμερα αντιπροσωπεύουν διάφορες ιδέες πάνω στον προγραμματισμό και η κάθε μία είναι συνήθως καλύτερα προσαρμοσμένη σε ορισμένες κατηγορίες προβλημάτων. Η μεγάλη πλειοψηφία των γλωσσών ανήκει στην κατηγορία των διαδικασιακών (procedural) γλωσσών. Είναι γνωστές επίσης και ως αλγοριθμικές γλώσσες, γιατί είναι σχεδιασμένες για να επιτρέπουν την υλοποίηση αλγορίθμων. Άλλες κατηγορίες γλωσ-σών υψηλού επιπέδου είναι:**

* **Αντικειμενοστραφείς γλώσσες (object -oriented languages)**
* **Συναρτησιακές γλώσσες (functional languages) π.χ. LISP**
* **Μη διαδικασιακές γλώσσες (non procedural languages) π.χ. PROLOG. Χαρακτηρίζονται επίσης και ως γλώσσες πολύ υψηλού επι-πέδου.**
* **Γλώσσες ερωταπαντήσεων (query languages) π.χ. SQL.**

**Μια άλλη ταξινόμηση μπορεί να προκύψει με βάση την περιοχή χρήσης. Με αυτό το κριτήριο διακρίνουμε:**

* **Γλώσσες γενικής χρήσης. Θεωρητικά κάθε γλώσσα γενικής χρήσης μπορεί να χρησιμο-ποιηθεί για την επίλυση οποιουδήποτε προ-βλήματος. Στην πράξη ωστόσο κάθε γλώσσα έχει σχεδιαστεί για να ανταποκρίνεται καλύ-τερα σε ορισμένη κατηγορία προβλημάτων.**

**Διακρίνονται σε:**

* + - **Γλώσσες επιστημονικής κατεύθυνσης (science-oriented languages) π.χ. FORTRAN.**
    - **Γλώσσες εμπορικής κατεύθυνσης (business- oriented languages) π.χ. COBOL.**

**Ας σημειωθεί ότι ορισμένες γλώσσες τα καταφέρ-νουν εξίσου καλά και στους δύο προηγούμενους τομείς π.χ. BASIC, Pascal.**

* **Γλώσσες προγραμματισμού συστημάτων (system programming languages) π.χ. C.**
* **Γλώσσες τεχνητής νοημοσύνης (artificial intelligence languages) π.χ. LISP, PROLOG.**
* **Γλώσσες ειδικής χρήσης. Πρόκειται για γλώσσες που χρησιμοποιούνται σε ειδικές περιοχές εφαρμογών όπως π.χ. στα γραφικά με υπολογιστή, στη ρομποτική, στη σχεδίαση ολοκληρωμένων κυκλωμάτων, στα Συστήματα Διοίκησης Βάσεων Δεδομένων, στην εκπαί-δευση μέσω υπολογιστή κ.α.**

**32 / 114**

**Ποια είναι n καλύτερη γλώσσα προγραμματισμού**

**Στην ιστορία του προγραμματισμού έχουν ανα-πτυχθεί χιλιάδες γλώσσες και αυτή τη στιγμή χρησιμοποιούνται μερικές εκατοντάδες. Υπάρχουν γλώσσες κατάλληλες για ανάπτυξη ειδικών εφαρμογών και άλλες κατάλληλες για γενική χρήση. Υπάρχουν γλώσσες κατάλληλες για εκπαίδευση και άλλες για ανάπτυξη εμπορικών εφαρμογών. Γλώσσες που επιτρέπουν την εύκολη ανάπτυξη εφαρμογών σε γραφικό περιβάλλον και άλλες που εκμεταλλεύονται τα παράλληλα συστήματα. Υπάρ-χουν γλώσσες πολύ ισχυρές αλλά πολύπλοκες και γλώσσες χωρίς μεγάλες δυνατότητες αλλά απλές και εύκολες στην εκμάθηση. Ο προγραμματιστής καλείται να επιλέξει την "καλύτερη" γλώσσα για να υλοποιήσει το πρόγραμμα.**

**Μπορούμε να ισχυριστούμε με βεβαιότητα ότι μία γλώσσα προγραμματισμού που να είναι αντικει-μενικά καλύτερη από τις άλλες δεν υπάρχει, ούτε πρόκειται να υπάρξει.**

**Η επιλογή της γλώσσας για την ανάπτυξη μιας εφαρμογής εξαρτάται από το είδος της εφαρμογής, το υπολογιστικό περιβάλλον στο οποίο θα εκτε-λεστεί, τα προγραμματιστικά περιβάλλοντα που διαθέτουμε και κυρίως τις γνώσεις του προγραμμα-τιστή. Συνήθως ο προγραμματιστής επιλέγει μία γλώσσα, που φυσικά επιτρέπει και διευκολύνει την ανάπτυξη του είδους της εφαρμογής στο συγκεκρι-μένο περιβάλλον με βάση όμως τις προσωπικές του γνώσεις και προτιμήσεις.**

**33 / 114**

|  |  |
| --- | --- |
| **6.3** | **Φυσικές και τεχνητές γλώσσες** |

**Οι γλώσσες προγραμματισμού αναπτύχθηκαν, για να μπορεί ο προγραμματιστής να δίνει τις εντολές που πρέπει να εκτελέσει ο υπολογιστής. Χρησιμοποιούνται δηλαδή για την επικοινωνία του ανθρώπου και της μηχανής, όπως αντίστοιχα οι φυσικές γλώσσες χρησιμοποιούνται για την επικοινωνία μεταξύ των ανθρώπων. Οι γλώσσες προγραμματισμού, που είναι τεχνητές γλώσσες, ακολουθούν τις βασικές έννοιες και αρχές της γλωσσολογίας, επιστήμη που μελετά τις φυσικές γλώσσες.**

**Μία γλώσσα προσδιορίζεται από το αλφάβητο της, το λεξιλόγιό της, τη γραμματική της και τέλος τη σημα-σιολογία της.**

**Το αλφάβητο**

**Αλφάβητο μίας γλώσσας καλείται το σύνολο των στοιχείων που χρησιμοποιείται από τη γλώσσα.**

**Για παράδειγμα η ελληνική γλώσσα περιέχει τα εξής στοιχεία: Τα γράμματα του αλφαβήτου πεζά και κεφαλαία 48 δηλαδή χαρακτήρες (Α-Ω και α-ω), τα 10 ψηφία (0-9) και όλα τα σημεία στίξης. Αντίστοιχα η αγγλική γλώσσα περιλαμβάνει τα γράμματα του αγγλικού αλφαβήτου (Α-Ζ και a-z) καθώς και τα ψηφία και όλα τα σημεία στίξης που χρησιμοποιούνται.**

**Το λεξιλόγιο**

**Το λεξιλόγιο αποτελείται από ένα υποσύνολο όλων των ακολουθιών που δημιουργούνται από τα στοιχεία του αλφαβήτου, τις λέξεις που είναι δεκτές από την γλώσσα. Για παράδειγμα στην ελληνική γλώσσα η ακολουθία των γραμμάτων ΑΒΓΑ είναι δεκτή αφού αποτελεί λέξη, αλλά η ακολουθία ΑΒΓΔΑ δεν αποτελεί λέξη της ελληνικής γλώσσας, άρα δεν είναι δεκτή.**

**34 / 115**

**Η Γραμματική**

**Η Γραμματική αποτελείται από το τυπικό ή τυπολογικό (accidence) και το συντακτικό (syntax).**

**Τυπικό είναι το σύνολο των κανόνων που ορίζει τις μορφές με τις οποίες μία λέξη είναι αποδεκτή. Για παράδειγμα στην ελληνική γλώσσα οι λέξεις γλώσσα, γλώσσας, γλώσσες είναι δεκτές, ενώ η λέξη γλώσσατ δεν είναι αποδεκτή.**

**Συντακτικό είναι το σύνολο των κανόνων που κα-θορίζει τη νομιμότητα της διάταξης και της σύνδεσης των λέξεων της γλώσσας για τη δημιουργία προ-τάσεων.**

**Η γνώση του συντακτικού επιτρέπει τη δημιουργία σωστών προτάσεων στις φυσικές γλώσσες ενώ στις γλώσσες προγραμματισμού τη δημιουργία σωστών εντολών.**

**Η σημασιολογία**

**Η σημασιολογία (Semantics) είναι το σύνολο των κανόνων που καθορίζει το νόημα των λέξεων και κατά επέκταση των εκφράσεων και προτάσεων που χρη-σιμοποιούνται σε μία γλώσσα.**

**35 / 115**

**Στις γλώσσες προγραμματισμού οι οποίες είναι τεχνητές γλώσσες, ο δημιουργός της γλώσσας αποφασίζει τη σημασιολογία των λέξεων της γλώσσας.**

|  |  |
| --- | --- |
| **πιν.jpg** | **Κάθε γλώσσα προσδιορίζεται από το αλφάβητο της, το λεξιλόγιο της, τη γραμματική της και τη σημασιολογία της.** |

**Διαφορές φυσικών και τεχνητών γλωσσών.**

**Μία βασική διαφορά μεταξύ φυσικών και τεχνητών γλωσσών είναι η δυνατότητα εξέλιξής τους. Οι φυσικές γλώσσες εξελίσσονται συνεχώς, νέες λέξεις δημι-ουργούνται, κανόνες γραμματικής και σύνταξης αλλάζουν με την πάροδο του χρόνου και αυτό γιατί η γλώσσα χρησιμοποιείται για την επικοινωνία μεταξύ ανθρώπων, που εξελίσσονται και αλλάζουν ανάλογα με τις εποχές και τον κοινωνικό περίγυρο.**

**Αντίθετα οι τεχνητές γλώσσες χαρακτηρίζονται από στασιμότητα, αφού κατασκευάζονται συνειδητά για ένα συγκεκριμένο σκοπό.**

**Ωστόσο συχνά οι γλώσσες προγραμματισμού βελτι-ώνονται και μεταβάλλονται από τους δημιουργούς τους, με σκοπό να διορθωθούν αδυναμίες ή να καλύψουν μεγαλύτερο εύρος εφαρμογών ή τέλος να ακολουθήσουν τις νέες εξελίξεις. Οι γλώσσες προ-γραμματισμού αλλάζουν σε επίπεδο διαλέκτου (για παράδειγμα GW-Basic και QuickBasic) ή σε επίπεδο επέκτασης (για παράδειγμα Basic και Visual Basic).**

**36 / 115-116**

|  |  |
| --- | --- |
| **6.4** | **Τεχνικές σχεδίασης προγραμμάτων** |

**Από την αρχή της εμφάνισης των υπολογιστών γίνονται συνεχείς προσπάθειες ανάπτυξης μεθο-δολογιών και τεχνικών προγραμματισμού, που θα εξασφαλίζουν τη δημιουργία απλών και κομψών προγραμμάτων, την εύκολη γραφή τους όσο και την κατανόησή τους.**

**6.4.1 Ιεραρχική σχεδίαση προγράμματος**

**Η τεχνική της ιεραρχικής σχεδίασης και επίλυσης ή η διαδικασία σχεδίασης "από επάνω προς τα κάτω" πως συχνά ονομάζεται (top-down program design) περιλαμβάνει τον καθορισμό των βασικών λειτουργιών ενός προγράμματος, σε ανώτερο επίπεδο, και στη συνέχεια τη διάσπαση των λειτουργιών αυτών σε όλο και μικρότερες λειτουργίες, μέχρι το τελευταίο επίπεδο που οι λειτουργίες είναι πολύ απλές, ώστε να επιλυθούν εύκολα.**

**Σκοπός της ιεραρχικής σχεδίασης είναι η διάσπαση λοιπόν του προβλήματος σε μια σειρά από απλού-στερα υποπροβλήματα, τα οποία να είναι εύκολο να επιλυθούν οδηγώντας στην επίλυση του αρχικού προβλήματος.**

**Για την υποβοήθηση της ιεραρχικής σχεδίασης χρησιμοποιούνται διάφορες διαγραμματικές τεχνικές, όπως για παράδειγμα το διάγραμμα του σχήματος 6.9.**

**37 / 116**



**Σχ. 6.9. Ιεραρχική σχεδίαση υπολογισμού του φόρου εισοδήματος**

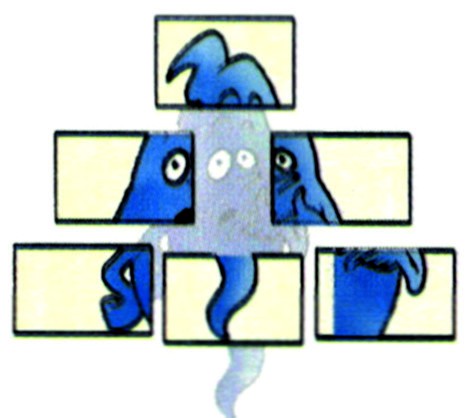
**000 / 000**

**38 / 117**

**000 / 000**

**000 / 000**

**6.4.2 Τμηματικός προγραμματισμός**

**Η ιεραρχική σχεδίαση προ-γράμματος υλοποιείται με τον τμηματικό προγραμματισμό. Μετά την ανάλυση του προ-βλήματος σε αντίστοιχα υπο-προβλήματα, κάθε υποπρό-βλημα αποτελεί ανεξάρτητη ενότητα (module), που γράφεται ξεχωριστά από τα υπόλοιπα τμήματα προγράμματος.**

**Η σωστή διαίρεση του αρχικού προβλήματος σε υποπροβλήματα και κατά συνέπεια του αρχικού προγράμματος σε τμήματα προγράμματος είναι μία διαδικασία αρκετά πολύπλοκη και θα εξεταστεί σε επόμενο κεφάλαιο.**

**Εδώ πρέπει να σημειωθεί ότι ο τμηματικός προ-γραμματισμός διευκολύνει τη δημιουργία του προγράμματος, μειώνει τα λάθη και επιτρέπει την ευκολότερη παρακολούθηση, κατανόηση και συντήρηση του προγράμματος από τρίτους.**

|  |  |
| --- | --- |
| **πιν.jpg** | **Η ιεραρχική σχεδίαση ή ιεραρχικός προγραμματισμός χρησιμοποιεί τη στρατηγική της συνεχούς διαίρεσης του προβλήματος σε υποπροβλήματα.** |

**39 / 116-117**

**6.4.3 Δομημένος προγραμματισμός**

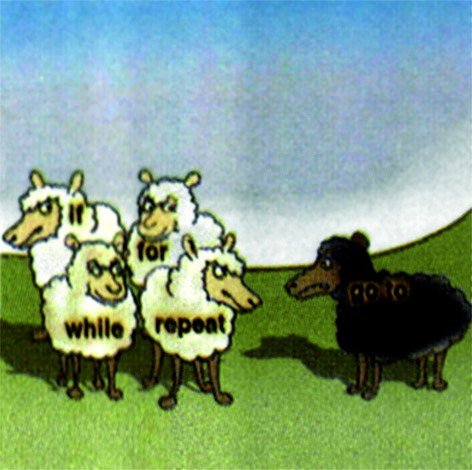
**Η μεθοδολογία που σήμερα έχει επικρατήσει απόλυτα και σχεδόν όλες οι σύγχρονες γλώσσες προγραμ-ματισμού υποστηρίζουν, είναι ο δομημένος προγραμματισμός (structured programming). Ο δομημένος προγραμματισμός παρουσιάστηκε στα μέσα του 1960.**

**Συγκεκριμένα το 1964 σε ένα συνέδριο στο Ισραήλ παρουσιάστηκε ένα κείμενο των Bohm και Jacopini με τις θεωρητικές αρχές του δομημένου προγραμματισμού. Οι απόψεις τους δεν έγιναν αρχικά ευρύτερα γνωστές και αποδεκτές, αλλά το 1968 ο καθηγητής Edsger Dijkstra δημοσίευσε ένα κείμενο που έκανε ιδιαίτερη αίσθηση και έμελλε να αλλάξει σταδιακά τον τρόπο προγραμματισμού καθώς και τις ίδιες τις γλώσσες προγραμματισμού. Ο τίτλος της μελέτης αυτής ήταν "GO TO Statement Considered Harmful η εντολή GOTO θεωρείται επιβλαβής" και θεμελίωνε το δομημένο προγραμματισμό. Χρειάστηκε όμως να περάσουν αρκετά χρόνια, ώστε να αρχίσει να διαδίδεται η χρήση του δομημένου προγραμματισμού.**

**Την εποχή εκείνη δεν υπήρχε μία μεθοδολογία για την ανάπτυξη των προγραμμάτων, τα προγράμματα ήταν μεγάλα και ιδιαίτερα μπερδεμένα με αποτέλεσμα να ξοδεύεται πάρα πολύς χρόνος τόσο στην συγγραφή όσο κύρια στη διόρθωση και τη μετέπειτα συντήρηση τους. Βασικός λόγος για τα προβλήματα αυτά ήταν η αλόγιστη χρήση μίας εντολής, της εντολής GOTO που χρησιμοποιούμενη άλλαζε διαρκώς τη ροή του προγράμματος.**

**40 / 117-118**

**GOTO: Το μαύρο πρόβατο του προγραμματισμού**

****

**Στην ιστορία του προγραμματισμού καμία άλλη εντολή δεν συζητήθηκε τόσο πολύ όσο η εντολή GOTO (πήγαινε). Η εντολή GOTO έχει ως αποτέ-λεσμα την αλλαγή της ροής του προγράμματος, της διακλάδωσης σε μία άλλη εντολή του προγράμ-ματος εκτός από την επόμενη. Η εντολή αυτή χώρισε τους προγραμματιστές σε δύο αντιμα-χόμενες ομάδες. Η μία αποτελείτο από φανατικούς υποστηρικτές της χρήσης του GOTO, οι οποίοι με τη χρήση της έλυναν εύκολα και αβασάνιστα προβλήματα της ανάπτυξης των προγραμμάτων τους προγραμμάτων τους και η δεύτερη με πολέμιους που έβλεπαν ότι η εντολή αυτή ήταν υπεύθυνη για τη δυσκολία στην αρχική σχεδίαση της λύσης, στην παρακολούθηση και κατανόηση του προγράμματος και τέλος στη συντήρηση. Ο δομημένος προγραμματισμός προήλθε από την ανάγκη του περιορισμού της ανεξέλεγκτης χρήσης του GOTO.**

**Η χρήση της εντολής αυτής θα παρουσιαστεί με ένα απλό παράδειγμα, ενώ για λόγους σύγκρισης δίνεται ο ψευδοκώδικας με χρήση της δομής επιλογής, όπως παρουσιάστηκε στα προηγούμενα κεφάλαια.**

**41 / 118**

**ΑΝ Αριθμός > 0 ΤΟΤΕ GOTO 1**

**ΑΝ Αριθμός < 0 ΤΟΤΕ GOTO 2**

**ΓΡΑΨΕ `Αρνητικός**

**GOTO 4**

**1: ΓΡΑΨΕ `Θετικός΄**

**GOTO 4**

**2: ΓΡΑΨΕ `Μηδέν΄**

**GOTO 4**

**4: ! Συνέχεια,**

**΄΄΄΄**

**ΑΝ Αριθμός > 0 ΤΟΤΕ ΓΡΑΨΕ `Θετικός΄**

**ΑΛΛΙΩΣ\_ΑΝ Αριθμός = 0 ΤΟΤΕ ΓΡΑΨΕ `Μηδέν΄**

**ΑΛΛΙΩΣ ΓΡΑΨΕ `Αρνητικός΄**

**ΤΕΛΟΣ\_ ΑΝ**

**Η χρήση του GOTO κάνει ακόμα και αυτό το μικρό τμήμα προγράμματος δύσκολο στην κατανόηση του και στην παρακολούθησή του.**

**Όλες οι σύγχρονες γλώσσες προγραμματισμού, υποστηρίζουν το δομημένο προγραμματισμό και διαθέτουν εντολές που καθιστούν τη χρήση του GOTO περιττή. Για λόγους όμως συμβατότητας με τις παλιότερες εκδόσεις τους καθώς και για λόγους συντήρησης παλιών προγραμμάτων, μερικές τη διατηρούν στο ρεπερτόριο των εντολών τους.**

**Στη συνέχεια αυτού του βιβλίου η εντολή GOTO δεν θα μας απασχολήσει και καλό είναι να μη χρησιμοποιείται στην ανάπτυξη προγραμμάτων.**

**Ο δομημένος προγραμματισμός αναπτύχθηκε από την ανάγκη να υπάρχει μία κοινή μεθοδολογία στην ανάπτυξη των προγραμμάτων και τη μείωση των εντολών GOTO που χρησιμοποιούνται στο πρόγραμμα.**

**42 / 118-119**

**Ο δομημένος προγραμματισμός δεν είναι απλώς ένα είδος προγραμματισμού, είναι μία μεθοδολογία σύνταξης προγραμμάτων που έχει σκοπό να βοηθήσει τον προγραμματιστή στην ανάπτυξη σύνθετων προγραμμάτων, να μειώσει τα λάθη, να εξασφαλίσει την εύκολη κατανόηση των προγραμμάτων και να διευκολύνει τις διορθώσεις και τις αλλαγές σε αυτά.**

**Ο δομημένος προγραμματισμός στηρίζεται στη χρήση τριών και μόνο στοιχειωδών λογικών δομών: τη δομή της ακολουθίας, τη δομή της επιλογής και τη δομή της επανάληψης. Όλα τα προγράμματα μπορούν να γραφούν χρησιμοποιώντας μόνο αυτές τις τρεις δομές καθώς και συνδυασμό τους. Κάθε πρόγραμμα όπως και κάθε ενότητα προγράμματος έχει μόνο μία είσοδο και μόνο μία έξοδο.**

**Οι τρεις αυτές δομές παρουσιάστηκαν στο κεφάλαιο 2 και θα επαναληφθούν στα επόμενα κεφάλαια.**

**Αν και ο δομημένος προγραμματισμός αρχικά εμφα-νίστηκε σαν μία προσπάθεια περιορισμού των εντολών GOTO, σήμερα αποτελεί τη βασική μεθοδολογία προγραμματισμού.**

**43 / 119**

**Ο δομημένος προγραμματισμός ενθαρρύνει και βοηθάει την ανάλυση του προγράμματος σε επιμέρους τμήματα, έτσι ώστε σήμερα ο όρος δομημένος προγραμματισμός περιέχει τόσο την ιεραρχική σχεδίαση όσο και τον τμηματικό προγραμματισμό.**

**Πλεονεκτήματα του δομημένου προγραμματισμού**

**Επιγραμματικά μπορούμε να αναφέρουμε τα εξής 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** |

**Μία νέα ιδέα στον προγραμματισμό γεννήθηκε στις παγωμένες νορβηγικές ακτές στα τέλη της δεκαετίας του '70 και πέρασε πολύ γρήγορα στην άλλη μεριά του Ατλαντικού. Πρόκειται για μια νέα τάση αντιμετώπισης προγραμματιστικών αντιλήψεων και δομών που ονομάζεται αντικειμενοστραφής (object-oriented) προγραμματισμός. Την τελευταία δεκαετία έχει γίνει η επικρατούσα κατάσταση και έχει αλλάξει ριζικά τα μέχρι πριν από λίγα χρόνια γνωστά και σταθερά σημεία αναφοράς των προγραμματιστών.**

**44 / 119**

**Η ιδέα του αντικειμενοστραφούς προγραμματισμού ή της αντικειμενοστραφούς σχεδίασης έχει τις ρίζες της σε πολύ απλοϊκή ιδέα. Ένα πρόγραμμα περιγράφει "ενέργειες" (επεξεργασία) που εφαρμόζονται πάνω σε δεδομένα. Ένα βασικό ερώτημα που τίθεται είναι αν η φιλοσοφία, η δομή του προγράμματος είναι προτι-μότερο να στηρίζεται στις "ενέργειες" ή στα δεδομένα. Η απάντηση σε αυτό το ερώτημα προσδιορίζει και τη βασική διαφορά ανάμεσα στις παραδοσιακές προ-γραμματιστικές τεχνικές και στην αντικειμενοστραφή προσέγγιση.**

**Η αντικειμενοστραφής σχεδίαση εκλαμβάνει ως πρωτεύοντα δομικά στοιχεία ενός προγράμματος τα δεδομένα, από τα οποία δημιουργούνται με κατάλληλη μορφοποίηση τα αντικείμενα (objects). Αυτή η σχεδί-αση αποδείχθηκε ότι επιφέρει καλύτερα αποτελέσματα, αφού τα προγράμματα που δημιουργούνται είναι περισσότερο ευέλικτα κα επαναχρησιμοποιήσιμα. Βέβαια, δημιουργούνται μία σειρά από εύλογα ερωτήματα, όπως "Τι ακριβώς είναι ένα αντικείμενο;", "Πώς προσδιορίζουμε και περιγράψουμε ένα αντικείμενο;", "Πώς το πρόγραμμα χειρίζεται τα αντικείμενα;", "Πώς τα αντικείμενα συσχετίζονται μεταξύ τους;". Απαντήσεις σε αυτά τα ερωτήματα καθώς και αναλυτική παρουσίαση του αντικειμε-νοστραφούς προγραμματισμού υπάρχουν στο κεφάλαιο 11.**

**Φυσικά ο αντικειμενοστραφής προγραμματισμός χρησιμοποιεί την ιεραρχική σχεδίαση, τον τμηματικό προγραμματισμό και ακολουθεί τις αρχές του δομη-μένου προγραμματισμού.**

**45 / 119-120**

|  |  |
| --- | --- |
| **6.6** | **Παράλληλος προγραμματισμός** |

**Μία άλλη μορφή προγραμματισμού που αναπτύσσεται τελευταία και πιθανόν στο μέλλον να γνωρίσει μεγάλη άνθηση, είναι ο παράλληλος προγραμματισμός. Σχετικά πρόσφατα εμφανίστηκαν υπολογιστές που ξεφεύγουν από την κλασική αρχιτεκτονική και διαθέτουν περισσότερους από έναν επεξεργαστές. Οι επεξεργαστές αυτοί μοιράζονται την ίδια μνήμη και λειτουργούν παράλληλα εκτελώντας διαφορετικές εντολές του ιδίου προγράμματος. Οι υπολογιστές αυτοί εμφανίζονται θεωρητικά να πετυχαίνουν ταχύτητες, που είναι ασύλληπτες για τους τυπικούς υπολογιστές με έναν επεξεργαστή. Για να εκμεταλλευτούμε όμως την ταχύτητα που προσφέρει η αρχιτεκτονική τους, πρέπει το πρόβλημα να διαιρεθεί σε τμήματα που εκτελούνται παράλληλα και στη συνέχεια να προγραμματιστεί σε ένα προγραμματιστικό περιβάλλον που να επιτρέπει τον παράλληλο προγραμματισμό.**

**Όπως αναφέρθηκε και στο κεφάλαιο 5, ο παράλληλος προγραμματισμός αποτελεί μία σημαντική επιστημονική περιοχή, η οποία ξεφεύγει από τα όρια αυτού του βιβλίου.**

**46 / 120**

|  |  |
| --- | --- |
|  | **Μια γλώσσα προγραμματισμού που υποστηρίζει παράλληλο προγραμ-ματισμό είναι η OCCAM.** |

|  |  |
| --- | --- |
| **6.7** | **Προγραμματιστικά περιβάλλοντα** |

**Κάθε πρόγραμμα που γράφτηκε σε οποιαδήποτε γλώσσα προγραμ-ματισμού, πρέπει να μετατραπεί σε μορφή αναγνωρίσιμη και εκτελέσι-μη από τον υπολογιστή, δηλαδή σε εντολές γλώσσας μηχανής.**

**Η μετατροπή αυτή επιτυγχάνεται με τη χρήση ειδικών μεταφραστικών προγραμμάτων. Υπάρχουν δύο μεγάλες κατηγορίες τέτοιων προ-γραμμάτων, οι μεταγλωττιστές (compilers) και οι διερμηνευτές (interpreters). Ο μεταγλωττιστής δέχεται στην είσοδο ένα πρόγραμμα γραμμένο σε μια γλώσσα υψηλού επιπέδου και παρά-γει ένα ισοδύναμο πρόγραμμα σε γλώσσα μηχανής. Το τελευταίο μπορεί να εκτελείται οποτεδήποτε από τον υπολογιστή και είναι τελείως ανεξάρτητο από το αρχικό πρόγραμμα. Αντίθετα ο διερμηνευτής διαβάζει μία προς μία τις εντολές του αρχικού προγράμματος και για κάθε μια εκτελεί αμέσως μια ισοδύναμη ακολουθία εντολών μηχανής.**

**Το αρχικό πρόγραμμα λέγεται πηγαίο πρόγραμμα (source), ενώ το πρόγραμμα που παράγεται από το μεταγλωττιστή λέγεται αντικείμενο πρόγραμμα (object).**

**Το αντικείμενο πρόγραμμα είναι μεν σε μορφή κατά-νοητή από τον υπολογιστή, αλλά συνήθως δεν είναι σε θέση να εκτελεστεί. Χρειάζεται να συμπληρωθεί και να συνδεθεί με άλλα τμήματα προγράμματος απαραίτητα για την εκτέλεσή του, τμήματα που είτε τα γράφει ο προγραμματιστής είτε βρίσκονται στις βιβλιοθήκες (libraries) της γλώσσας. Το πρόγραμμα που επιτρέπει αυτή τη σύνδεση ονομάζεται συνδέτης - φορτωτής (linker- loader). Το αποτέλεσμα του συνδέτη είναι η παραγωγή του εκτελέσιμου προγράμματος (executable), το οποίο είναι το τελικό πρόγραμμα που εκτελείται από τον υπολογιστή. Για το λόγο αυτό η συνολική διαδικασία αποκαλείται μεταγλώττιση και σύνδεση.**

**47 / 120-121**

**Η δημιουργία του εκτελέσιμου προγράμματος γίνεται μόνο στην περίπτωση, που το αρχικό πρόγραμμα δεν περιέχει συντακτικά λάθη. Τις περισσότερες φορές κάθε πρόγραμμα αρχικά θα έχει λάθη. Τα λάθη του προγράμ-ματος είναι γενικά δύο ειδών, λογικά και συντακτικά. Τα λογικά λάθη εμφανίζονται μόνο στην εκτέλεση, ενώ τα συντακτικά λάθη στο στάδιο της μεταγλώττισης.**

**48 / 121**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | |
|  | **Αρχικό Πρόγραμμα** | **Μεταγλωτ-**  **τιστής** | **Αντικείμενο**  **Πρόγραμμα** | **Συνδέτης** | **Εκτελέσιμο Πρόγραμμα** |  |

**Σχ. 6.10. Μεταγλώττιση και σύνδεση προγράμματος**

**000 / 000**

**49 / 121**

**Εκτενής παρουσίαση των λαθών και των τρόπων που αντιμετωπίζονται γίνεται σε επόμενο κεφάλαιο. Εδώ αναφέρουμε ότι τα λογικά λάθη που είναι τα πλέον σοβαρά και δύσκολα στη διόρθωσή τους, οφείλονται σε σφάλματα κατά την υλοποίηση του αλγορίθμου, ενώ τα συντακτικά οφείλονται σε αναγραμματισμούς ονομάτων εντολών, παράληψη δήλωσης δεδομένων και πρέπει πάντα να διορθωθούν, ώστε να παραχθεί το τελικό εκτελέσιμο πρόγραμμα.**

**Ο μεταγλωττιστής ή ο διερμηνευτής ανιχνεύει λοιπόν τα συντακτικά λάθη και εμφανίζει κατάλληλα διαγνωστικά μηνύματα. Το στάδιο που ακολουθεί είναι η διόρθωση των λαθών. Το διορθωμένο πρόγραμμα επαναϋπο-βάλεται για μεταγλώττιση και η διαδικασία αυτή επαναλαμβάνεται, μέχρις ότου εξαλειφθούν πλήρως όλα τα λάθη.**

**Η χρήση μεταγλωττιστή έχει το μειονέκτημα, ότι προτού χρησιμοποιηθεί ένα πρόγραμμα, πρέπει να περάσει από τη διαδικασία της μεταγλώττισης και σύνδεσης. Από την άλλη μεριά η χρήση διερμηνευτή έχει το πλεονέκτημα της άμεσης εκτέλεσης και συνεπώς και της άμεσης διόρθωσης. Όμως η εκτέλεση του προ-γράμματος καθίσταται πιο αργή, σημαντικά μερικές φορές, από εκείνη του ισοδύναμου εκτελέσιμου προγράμματος που παράγει ο μεταγλωττιστής. Πάντως τα σύγχρονα προγραμματιστικά περιβάλλοντα παρουσιάζονται συνήθως με μεικτές υλοποιήσεις, όπου χρησιμοποιείται διερμηνευτής κατά τη φάση δημι-ουργίας του προγράμματος και μεταγλωττιστής για την τελική έκδοση και εκμετάλλευση του προγράμματος.**

**50 / 121-122**



**100 / 139**

**Σχ. 6.11. Διαδικασία μετάφρασης και εκτέλεσης ενός προγράμματος**

**51 / 122**

**Για την αρχική σύνταξη των προγραμμάτων και τη διόρθωσή τους στη συνέχεια χρησιμοποιείται ένα ειδικό πρόγραμμα που ονομάζεται συντάκτης (editor). Ο συντάκτης είναι ουσιαστικά ένας μικρός επεξεργαστής κειμένου, με δυνατότητες όμως που διευκολύνουν τη γρήγορη γραφή των εντολών των προγραμμάτων.**

**Για τη δημιουργία, τη μετάφραση και την εκτέλεση ενός προγράμματος απαιτούνται τουλάχιστον τρία προγράμματα: ο συντάκτης, ο μεταγλωττιστής και ο συνδέτης. Τα σύγχρονα προγραμματιστικά περι-βάλλοντα παρέχουν αυτά τα προγράμματα με ενιαίο τρόπο.**

**Το κάθε προγραμματιστικό περιβάλλον έχει φυσικά διαφορετικά εργαλεία και ιδιότητες. Για παράδειγμα ένα περιβάλλον οπτικού (visual) προγραμματισμού πρέπει να περιέχει οπωσδήποτε και ειδικό συντάκτη που να διευκολύνει τη δημιουργία γραφικών αντικειμένων (για παράδειγμα φόρμες, λίστες, παράθυρα διαλόγου) παρέχοντας στον προγραμματιστή τα αντίστοιχα γραφικά εργαλεία.**

**52 / 122**

**• Ανακεφαλαίωση**

**Δημιουργία προγράμματος είναι η μετατροπή του αλγορίθμου που επιλύει ένα πρόβλημα σε εντολές προγράμματος. Οι εντολές γράφονται σε κάποια από τις εκατοντάδες γλώσσες προγραμματισμού, που έχουν αναπτυχθεί με σκοπό να διευκολύνουν την επίλυση συγκεκριμένου τύπου προβλημάτων. Η επιλογή της καταλληλότερης γλώσσας εξαρτάται από το είδος της εφαρμογής, το υπολογιστικό περιβάλλον που θα εκτελεστεί, τις γνώσεις και προτιμήσεις του προ-γραμματιστή.**

**Οι τεχνικές που χρησιμοποιούνται για την ανάπτυξη προγραμμάτων είναι της ιεραρχικής σχεδίασης, του τμηματικού προγραμματισμού και του δομημένου προγραμματισμού, που επιτρέπουν τη δημιουργία προγραμμάτων κατανοητών και απλών, ενώ διευκο-λύνουν τη διόρθωση και τη συντήρηση των εφαρ-μογών. Ένα είδος προγραμματισμού που γνωρίζει ιδιαίτερη άνθηση τελευταία, είναι ο αντικειμενοστραφής προγραμματισμός.**

**Κάθε πρόγραμμα για να εκτελεστεί από τον υπολογιστή χρειάζεται πρώτα να μετατραπεί σε μορφή κατανοητή από αυτόν. Η μετατροπή αυτή γίνεται από τους μετα-γλωττιστές ή τους διερμηνευτές, οι οποίοι επισημαί-νουν και τα συντακτικά λάθη, που έχει κάθε πρόγραμ-μα. Η σύνταξη του προγράμματος, η μετάφραση, η διόρθωση των λαθών και η εκτέλεσή του γίνεται με τα ολοκληρωμένα προγραμματιστικά περιβάλλοντα που διαθέτουν πολλά εργαλεία για την υποβοήθηση της ανάπτυξης των εφαρμογών.**

**53 / 123**

**Λέξεις κλειδιά**

**Πρόγραμμα, Γλώσσα μηχανής, Συμβολική γλώσσα, Γλώσσες υψηλού επιπέδου, Τμηματικός προγραμ-ματισμός, Δομημένος προγραμματισμός, Αντικει-μενοστραφής προγραμματισμός, Μεταγλωττιστής, Διερμηνευτής, Προγραμματιστικό περιβάλλον**

**• Ερωτήσεις Θέματα για συζήτηση**

**54 / 123**

1. **Τι ονομάζεται πρόγραμμα;**
2. **Τι είναι οι γλώσσες μηχανής;**
3. **Ποιες οι διαφορές των γλωσσών υψηλού επιπέδου από αυτές χαμηλού επιπέδου;**
4. **Ποιες γλώσσες υψηλού επιπέδου γνωρίζεις;**
5. **Τι ονομάζουμε οπτικό προγραμματισμό και τι οδηγούμενο από τα γεγονότα;**
6. **Πώς προσδιορίζεται μία φυσική γλώσσα;**
7. **Ποιες οι κυριότερες διαφορές των φυσικών και των τεχνητών γλωσσών;**
8. **Πώς γίνεται η παράσταση της ιεραρχικής σχεδίασης προγράμματος;**
9. **Ποιες οι αρχές του δομημένου προγραμματισμού;**
10. **Ποια τα πλεονεκτήματα του δομημένου προγραμματισμού;**
11. **Τι ονομάζεται αντικειμενοστραφής προγραμματισμός;**
12. **Ποια η διαδικασία για την μετάφραση και εκτέλεση ενός προγράμματος;**
13. **Ποιες οι διαφορές μεταγλωττιστή και διερμηνευτή;**
14. **Ποια προγράμματα και εργαλεία περιέχει ένα προγραμματιστικό περιβάλλον;**

•**Βιβλιογραφία**

1. **Ph. Breton, Ιστορία της Πληροφορικής, Εκδόσεις Δίαυλος, Αθήνα.**
2. **Γ. Μπαμπινιώτης, Θεωρητική Γλωσσολογία, Αθήνα, 1986.**
3. **Χρ. Κοίλιας-Στρ., Καλαφατούδης, Το πρώτο βιβλίο της Πληροφορικής, Εκδόσεις Νέων Τεχνολογιών, Αθήνα, 1992.**
4. **Εγκυκλοπαίδεια Πληροφορικής και Τεχνολογίας Υπολογιστών, Εκδόσεις Νέων Τεχνολογιών, Αθήνα, 1986.**
5. **Αθ. Τσουροπλής-Στ. Κλημόπουλος, Από τη FORTRAN 77 στη FORTRAN 90, Εκδόσεις Πελεκάνος, Αθήνα, 1995.**
6. **Χ. Κοίλας-Στρ. Μαραγκός, Η γλώσσα COBOL και οι εφαρμογές της, Εκδόσεις Νέων Τεχνολογιών, Αθήνα, 1992.**
7. **Κ. Μαρινάκης-Ν. Ιωαννίδης, Structure & Advanced COBOL, Εκδόσεις Έλιξ, Αθήνα, 1992.**
8. **Χ. Κοίλιας-Αλ. Τομαράς, CWBASIC Θεωρία και Εφαρμογές, Εκδόσεις Νέων Τεχνολογιών, Αθήνα, 1992.**
9. **Μ. Κατζουράκη-Μ. Γεργατσούλης-Σ. Κόκκοτος, PROγpαμματίζοντας στη LOG/κή, Έκδοση ΕΠΥ, Αθήνα, 1991.**

**55 / 123-124**

1. **Αικ. Γεωργόπουλου, LOGO Βήμα προς Βήμα, Εκδόσεις Νέων Τεχνολογιών, Αθήνα, 1991.**
2. **Αλ. Τομαράς, C Θεωρία και Πράξη, Εκδόσεις Νέων Τεχνολογιών, Αθήνα, 1995.**
3. **Μ. Μαλιάππης, SQL Περιβάλλοντα Ανάπτυξης Εφαρμογών 4ης Γενιάς, Εκδόσεις Νέων Τεχνολογιών, Αθήνα, 1995.**
4. **Ε. Horowitz, Βασικές αρχές γλωσσών προγραμματισμού, Κλειδάριθμος, Αθήνα, 1995.**
5. **R. Shackelford, Introduction to Computing and Algorithms, Addison-Wesley, USA, 1998.**
6. **W. Hutching-H Somers, An Introduction to Machine Translation, Academic Press, London, 1992.**

**56 / 124**

**Κεφάλαιο 7**

**Βασικές Έννοιες**

**Προγραμματισμού**

**ΠΕΡΙΕΧΟΜΕΝΑ**

|  |  |
| --- | --- |
| **57 / 125** | **Αλφάβητο και τύποι δεδομένων** |
|  | **Σταθερές και μεταβλητές** |
|  | **Τελεστές, συναρτήσεις και εκφράσεις** |
|  | **Εντολή εκχώρησης** |
|  | **Εντολές εισόδου - εξόδου** |
|  | **Δομή προγράμματος** |

**• Εισαγωγή**

**Κάθε γλώσσα προγραμματισμού, όπως αναφέρθηκε, έχει το δικό της λεξιλόγιο και τα προγράμματά της ακολουθούν αυστηρούς γραμματικούς και συντακτικούς κανόνες. Για τη δημιουργία σωστών προγραμμάτων είναι απαραίτητη η γνώση των εντολών και του τρόπου σύνταξης τους.**

**Στο κεφάλαιο αυτό θα παρουσιαστούν τα βασικά στοι-χεία της ΓΛΩΣΣΑΣ. Θα ασχοληθούμε με τους τύπους δεδομένων που υποστηρίζει, τα είδη των μεταβλητών της, τον τρόπο που υπολογίζονται οι παραστάσεις καθώς και τη δομή που πρέπει να ακολουθεί κάθε πρόγραμμα.**

**Επίσης θα παρουσιαστούν οι βασικές εντολές της ΓΛΩΣΣΑΣ, η εντολή εκχώρησης τιμών σε μεταβλητές και οι εντολές εισόδου εξόδου, με τις οποίες το πρό-γραμμα επικοινωνεί με το χρήστη.**

**• Διδακτικοί στόχοι**

**Να είναι σε θέση ο μαθητής:**

* **Να διακρίνει τις σταθερές από τις μεταβλητές.**
* **Να αναγνωρίζει τους διάφορους τύπους μεταβλητών.**

**58 / 126**

* **Να μετατρέπει τις αριθμητικές πράξεις σε εντολές προγράμματος.**
* **Να διατυπώνει τη δομή ενός προγράμματος.**

**59 / 126**

* **Να συντάσσει απλά προγράμματα, τα οποία εισάγουν δεδομένα, τα επεξεργάζονται και εμφανίζουν τα αποτελέσματα στην οθόνη.**

**• Προερωτήσεις**

* **Έχει ένα πρόγραμμα συγκεκριμένους κανόνες στον τρόπο που γράφεται;**
* **Πώς διαχειρίζεται ένα πρόγραμμα τα δεδομένα;**
* **Πώς εκτελούνται οι πράξεις σε ένα πρόγραμμα;**
* **Με ποιο τρόπο επικοινωνεί το πρόγραμμα με το χρήστη κατά την εκτέλεση του;**

**Εκατοντάδες γλώσσες προγραμματισμού χρησιμο-ποιούνται όπως αναφέρθηκε σήμερα για την επίλυση των προβλημάτων με τον υπολογιστή, τη δημιουργία σωστών προγραμμάτων. Η επιλογή της κατάλληλής γλώσσας δεν είναι εύκολη και εξαρτάται από το είδος του προγράμματος, το διαθέσιμο εξοπλισμό και σαφώς τις γνώσεις και τις ιδιαίτερες προτιμήσεις του προ-γραμματιστή. Συχνά το ίδιο πρόβλημα μπορεί να λυθεί εξίσου ικανοποιητικά με πολλές διαφορετικές γλώσσες προγραμματισμού.**

**Πρέπει να έχουμε πάντα υπόψη μας ότι:**

* **Κάθε γλώσσα προγραμματισμού σχεδιάζεται για συγκεκριμένο σκοπό, δίνοντας ιδιαίτερη έμφαση σε ορισμένα χαρακτηριστικά σε βάρος βέβαια κάποιων άλλων. Δεν υπάρχει καλύτερη γλώσσα προγραμματισμού, απλά υπάρχει γλώσσα καταλληλότερη για την ανάπτυξη συγκεκριμένου τύπου εφαρμογών.**
* **Οι γλώσσες προγραμματισμού περιέχουν πολλές πληροφορίες που σχετίζονται με τεχνικά θέματα. Αυτά τα χαρακτηριστικά αλλάζουν αρκετά συχνά, όπως η γλώσσα εξελίσσεται και εξαρτώνται σε μεγάλο βαθμό από τον εξοπλισμό και το λει-τουργικό σύστημα. Οι νεώτερες εκδόσεις των γλωσσών συνήθως διαθέτουν πλουσιότερο ρεπερτόριο εντολών και άλλων δυνατοτήτων, χωρίς όμως να προσθέτουν οτιδήποτε στην εκμάθηση της δημιουργίας σωστών προγραμ-μάτων.**
* **Σχεδόν όλες οι γλώσσες προγραμματισμού έχουν κοινά χαρακτηριστικά, επεξεργάζονται κατά κανόνα τους ίδιους τύπους δεδομένων, υποστηρίζουν τις ίδιες βασικές δομές και έχουν παρόμοιες εντολές.**

**Η γλώσσα προγραμματισμού που θα χρησιμοποι-ήσουμε στα επόμενα κεφάλαια που ονομάζεται ΓΛΩΣΣΑ, είναι σχεδιασμένη έτσι ώστε να αποτελέσει ένα εργαλείο προγραμματισμού κατάλληλο για εκπαιδευτικούς σκοπούς. Περιέχει τα χαρακτηριστικά, τις δομές και τις εντολές που περιέχονται σε διάφορες σύγχρονες γλώσσες προγραμματισμού όπως η Pascal, Visual Basic, C, C+ +, Java και άλλες, χωρίς όμως να ασχολείται με τις τεχνικές λεπτομέρειες αυτών.**

**60 / 127**

**Έτσι ο προγραμματισμός με τη ΓΛΩΣΣΑ εστιάζεται στην ανάπτυξη του αλγορίθμου και τη μετατροπή του σε σωστό πρόγραμμα.**

**Σε όλο το βιβλίο οι εντολές της ΓΛΩΣΣΑΣ είναι γραμ-μένες με μπλε χρώμα και είναι πάντα με κεφαλαία, ενώ οι μεταβλητές είναι με πεζά ή κεφαλαία αλλά με το πρώτο γράμμα πάντα κεφαλαίο. Τα σχόλια των προγραμμάτων είναι γραμμένα με πράσινο χρώμα.**

|  |  |
| --- | --- |
| **7.1** | **Το αλφάβητο της ΓΛΩΣΣΑΣ** |

**Το αλφάβητο της ΓΛΩΣΣΑΣ αποτελείται από τα γράμματα του ελληνικού και του λατινικού αλφαβήτου, τα ψηφία, καθώς και από ειδικά σύμβολα, που χρησι-μοποιούνται για προκαθορισμένες ενέργειες, στις οποίες θα αναφερθούμε στη συνέχεια.**

**61 / 127-128**

**Συγκεκριμένα**

**Γράμματα**

**Κεφαλαία ελληνικού αλφαβήτου (Α-Ω)**

**Πεζά ελληνικού αλφαβήτου (α-ω)**

**Κεφαλαία λατινικού αλφαβήτου (Α-Ζ)**

**Πεζά λατινικού αλφαβήτου (a-z)**

**Ψηφία**

**0-9**

**Ειδικοί χαρακτήρες**

**+ - \* / = • ( ) . , ΄ ! & κενός χαρακτήρας ^**

|  |  |
| --- | --- |
| **7.2** | **Τύποι δεδομένων** |

**Οι υπολογιστές επεξεργάζονται δεδομένα διαφόρων τύπων, γι αυτό είναι σημαντικό να κατανοήσουμε τους διαφορετικούς τύπους δεδομένων που χειρίζεται η ΓΛΩΣΣΑ.**

**Οι τύποι δεδομένων που υποστηρίζει η ΓΛΩΣΣΑ είναι οι αριθμητικοί, που περιλαμβάνουν τους ακέραιους και τους πραγματικούς αριθμούς, οι χαρακτήρες και τέλος οι λογικοί.**

**Ακέραιος τύπος. Ο τύπος αυτός περιλαμβάνει τους ακέραιους που είναι γνωστοί από τα μαθηματικά. Οι ακέραιοι μπορούν να είναι θετικοί, αρνητικοί ή μηδέν. Παραδείγματα ακεραίων είναι οι αριθμοί 1, 3409, 0, -980.**

**Πραγματικός τύπος. Ο τύπος αυτός περιλαμβάνει τους πραγματικούς αριθμούς που γνωρίζουμε από τα μαθηματικά. Οι αριθμοί 3.14159, 2.71828, -112.45, 0.45 είναι πραγματικοί αριθμοί. Και οι πραγματικοί αριθμοί μπορούν να είναι θετικοί, αρνητικοί ή μηδέν.**

**Χαρακτήρας. Ο τύπος αυτός αναφέρεται τόσο σε ένα χαρακτήρα όσο και μία σειρά χαρακτήρων. Τα δεδομένα αυτού του τύπου μπορούν να περιέχουν οποιοδήποτε χαρακτήρα παράγεται από το πληκτρολόγιο. Παραδείγματα χαρακτήρων είναι 'Κ', 'Κώστας', 'σήμερα είναι Τετάρτη', 'Τα πολλαπλάσια του 15 είναι'.**

**62 / 128**

**Οι χαρακτήρες πρέπει υποχρεωτικά να βρίσκονται μέσα σε απλά εισαγωγικά, ' '. Τα δεδομένα αυτού του τύπου, επειδή περιέχουν τόσο αλφαβητικούς όσο και αριθμητικούς χαρακτήρες, ονομάζονται συχνά αλφαριθμητικά.**

**63 / 128**

**Λογικός. Αυτός ο τύπος δέχεται μόνο δύο τιμές ΑΛΗΘΗΣ και ΨΕΥΔΗΣ. Οι τιμές αντιπροσωπεύουν αληθείς ή ψευδείς συνθήκες.**

|  |  |
| --- | --- |
| **πιν.jpg** | **Στην πραγματικότητα τα δεδομένα καταχωρούνται στη μνήμη του υπο-λογιστή καταλαμβάνοντας συγκεκρι-μένο αριθμό θέσεων (bytes). Ανάλογα με τον τύπο του δεδομένου και το διατιθέμενο αριθμό bytes ποικίλει και το εύρος τιμών που μπορούν να λά-βουν. Έτσι στον υπολογιστή διαθέ-τουμε ένα υποσύνολο ακεραίων ή πραγματικών αριθμών. Συνήθεις τύποι δεδομένων στις διάφορες γλώσσες προγραμματισμού είναι ο ακέραιος (integer) σε 1,2 ή 4 bytes και ο πραγματικός (real) σε 4 ή 8 bytes.** |

|  |  |
| --- | --- |
| **7.3** | **Σταθερές** |

**Οι σταθερές (constants) είναι προκαθορισμένες τιμές που δεν μεταβάλλονται κατά τη διάρκεια εκτέλεσης του προγράμματος. Οι σταθερές είναι αντίστοιχου τύπου δεδομένων, δηλαδή ακέραιες, πραγματικές, αλφαριθμητικές ή λογικές.**

**Συμβολικές σταθερές**

**Η ΓΛΩΣΣΑ επιτρέπει την αντιστοίχιση σταθερών τιμών με ονόματα, εφόσον αυτά δηλωθούν στην αρχή του προγράμματος (στο τμήμα δήλωσης σταθερών, βλέπε παρακάτω).**

**Σύνταξη**

**ΜΕΤΑΒΛΗΤΕΣ  
 Όνομα-1 = σταθερή-τιμή-1  
 Όνομα-2 = σταθερή-τιμή-2  
 .  
 .   
 .  
 Όνομα-v = σταθερή-τιμή-v**

**Παραδείγματα**

**ΣΤΑΘΕΡΕΣ  
 ΠΙ = 3.14159  
 ΦΠΑ = 0,19  
 ΟΝΟΜΑ = 'ΚΩΣΤΑΣ'**

**Λειτουργία**

**Αποδίδει ονόματα σε σταθερές τιμές. Κάθε ένα από αυτά τα ονόματα μπορεί να χρησιμοποιηθεί οπου-δήποτε στο πρόγραμμα, αλλά δεν είναι δυνατή η μεταβολή της τιμής κατά τη διάρκεια εκτέλεσης του προγράμματος.**

**Η χρήση ονομάτων σταθερών κάνει το πρόγραμμα πιο κατανοητό και κατά συνέπεια ευκολότερο να διορθωθεί και να συντηρηθεί.**

**64 / 128-129**

**Ονόματα**

**Κάθε πρόγραμμα, καθώς και τα δεδομένα που χρησιμοποιεί (συμβολικές σταθερές και μεταβλητές) έχουν ένα όνομα, με το οποίο αναφερόμαστε σε αυτά. Τα ονόματα αυτά μπορούν να αποτελούνται από γράμματα πεζά ή κεφαλαία του ελληνικού ή του λατινικού αλφαβήτου (Α-Ω, Α-Ζ), ψηφία (0-9) καθώς και τον χαρακτήρα κάτω παύλα (underscore) (\_), ενώ πρέπει υποχρεωτικά να αρχίζουν με γράμμα.**

**Επειδή μερικές λέξεις χρησιμοποιούνται από την ίδια τη ΓΛΩΣΣΑ για συγκεκριμένους λόγους, όπως οι λέξεις ΠΡΟΓΡΑΜΜΑ, ΑΚΕΡΑΙΕΣ, ΠΡΑΓΜΑΤΙΚΕΣ, ΑΝ κ.λπ. αυτές οι λέξεις δεν μπορούν να χρησιμοποιηθούν ως ονόματα. Οι λέξεις αυτές αποκαλούνται δεσμευμένες.**

**Παραδείγματα ονομάτων που είναι αποδεκτά από τη ΓΛΩΣΣΑ είναι: Α, Όνομα, Τιμή, Τυπική Απόκλιση, Α100, ΦΠΑ, μέγιστο, Υπολογισμός Ταχύτητας.**

**Παραδείγματα ονομάτων που δεν είναι αποδεκτά είναι: 100Α, Μέση Τιμή, Κόστος$.**

**65 / 129**

|  |  |
| --- | --- |
| **7.4** | **Μεταβλητές** |

**Η έννοια της μεταβλητής (variable) είναι γνωστή από τα μαθηματικά. Για παράδειγμα ο τύπος της γεωμετρίας**

**Ε=αβ**

**υπολογίζει το εμβαδόν (Ε) ενός ορθογωνίου με διαστάσεις, που συμβολίζονται με α και β. Αν στο α και στο β δοθούν οι αντίστοι-χες τιμές, τότε ο τύπος αυτός υπολογίζει το εμβαδόν του ορθογωνίου.**

**Μια μεταβλητή λοιπόν, παριστάνει μία ποσότητα που η τιμή της μπορεί να μεταβάλλεται.**

**Οι μεταβλητές που χρησιμοποιούνται σε ένα πρό-γραμμα, αντιστοιχούνται από το μεταγλωττιστή σε συγκεκριμένες θέσεις μνήμης του υπολογιστή. Η τιμή της μεταβλητής είναι η τιμή που βρίσκεται στην αντίστοιχη θέση μνήμης και όπως αναφέρθηκε μπορεί να μεταβάλλεται κατά διάρκεια της εκτέλεσης του προγράμματος.**

**Μπορούμε να παρομοιάσουμε τη μεταβλητή και την αντίστοιχη θέση μνήμης σαν ένα γραμματοκιβώτιο, το οποίο εξωτερικά έχει ως όνομα το όνομα της μετα-βλητής και ως περιεχόμενο εσωτερικά, την τιμή που έχει εκείνη τη συγκεκριμένη στιγμή η μεταβλητή.**

**Ενώ η τιμή της μεταβλητής μπορεί να αλλάζει κατά την εκτέλεση του προγράμματος, αυτό που μένει υποχρεωτικά αναλλοίωτο είναι ο τύπος της μετα-βλητής.**

**66 / 129-130**

**67 / 130**

**Η ΓΛΩΣΣΑ επιτρέπει τη χρήση μεταβλητών των τεσ-σάρων τύπων που αναφέρθηκαν, δηλαδή ακεραίων, πραγματικών, χαρακτήρων και λογικών ενώ η δήλωση του τύπου κάθε μεταβλητής γίνεται υποχρεωτικά στο τμήμα δήλωσης μεταβλητών.**

**Το όνομα κάθε μεταβλητής, ακολουθεί τους κανόνες δημιουργίας ονομάτων, δηλαδή αποτελείται από γράμματα, ψηφία καθώς και τον χαρακτήρα \_, ενώ το όνομα κάθε μεταβλητής είναι μοναδικό για κάθε πρόγραμμα.**

**Σύνταξη**

**ΜΕΤΑΒΛΗΤΕΣ  
 τύπος-1 = Λίστα μεταβλητών-1  
 τύπος-2 = Λίστα μεταβλητών -2  
 .  
 .   
 .  
 τύπος-v = Λίστα μεταβλητών -v**

**Παραδείγματα**

**ΜΕΤΑΒΛΗΤΕΣ  
 ΠΡΑΓΜΑΤΙΚΕΣ : Εμβαδόν, Α   
 ΑΚΕΡΑΙΕΣ : TIMH, N  
 ΧΑΡΑΚΤΗΡΕΣ : Όνομα  
 ΛΟΓΙΚΕΣ : Έλεγχος**

**Λειτουργία**

**Δηλώνει τον τύπο όλων των μεταβλητών που χρησιμοποιούνται στο πρόγραμμα.**

**Αν και όπως αναφέρθηκε, το όνομα των μεταβλητών μπορεί να είναι οποιοσδήποτε συνδυασμός χαρα-κτήρων, είναι καλή πρακτική να χρησιμοποιούνται ονόματα, τα οποία να υπονοούν το περιεχόμενό τους, κάνοντας το πρόγραμμα ευκολότερο στην ανάγνωση του και στην κατανόησή του.**

**Για παράδειγμα στην περίπτωση του υπολογισμού του εμβαδού είναι προτιμότερη η χρήση του ονόματος ΕΜΒΑΔΟ για την αντίστοιχη μεταβλητή, από ένα όνομα που αποτελείται από ένα μόνο γράμμα όπως Ε ή Α ή ένα οποιοδήποτε τυχαίο όνομα που δεν ανάγει στο πραγματικό περιεχόμενο της μεταβλητής, όπως Τιμή.**

|  |  |
| --- | --- |
|  | **Συνιστάται τα ονόματα των μεταβλητών και των σταθερών να ανάγουν στο περιεχόμενό τους** |

|  |  |
| --- | --- |
| **7.5** | **Αριθμητικοί τελεστές** |

**Οι αριθμητικοί τελεστές που υποστηρίζονται από τη ΓΛΩΣΣΑ καλύπτουν τις βασικές πράξεις: πρόσθεση, αφαίρεση, πολλαπλασιασμό και διαίρεση ενώ υποστηρίζεται και η ύψωση σε δύναμη, η ακέραια διαίρεση και το υπόλοιπο της ακέραιας διαίρεσης.**

**Οι τελεστές και οι αντίστοιχες πράξεις είναι:**

**68 / 130-131**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **Αριθμητικός τελεστής**  **69/ 131** | **Πράξη** | | **+** | **Πρόσθεση** | | **-** | **Αφαίρεση** | | **\*** | **Πολλαπλασιασμός** | | **/** | **Διαίρεση** | | **^** | **Ύψωση σε δύναμη** | | **DIV** | **Ακέραια διαίρεση** | | **MOD** | **Υπόλοιπο  ακέραιης διαίρεσης** | |

|  |  |
| --- | --- |
| **πιν.jpg** | **Ο τελεστής div χρησιμοποιείται για τον υπολογισμό του πηλίκου μιας διαίρεσης ακεραίων αριθμών, ενώ ο τελεστής mod για το υπόλοιπο.**  **Π.χ. 7 div 2 = 3 και 7 mod 2 = 1** |

|  |  |
| --- | --- |
| **7.6** | **Συναρτήσεις** |

**Πολλές γνωστές συναρτήσεις από τα μαθηματικά χρησιμοποιούνται συχνά και περιέχονται στη ΓΛΩΣΣΑ.   
Οι συναρτήσεις αυτές είναι:**

**ΗΜ(Χ) Υπολογισμός ημίτονου**

**ΣΥΝ(Χ) Υπολογισμός συνημίτονου**

**ΕΦ(Χ) Υπολογισμός εφαπτομένης**

**Τ\_Ρ(Χ) Υπολογισμός τετραγωνικής ρίζας**

**ΛΟΓ(Χ) Υπολογισμός φυσικού λογαρίθμου**

**Ε(Χ) Υπολογισμός του eχ**

**Α\_Μ(Χ) Ακέραιο μέρος του Χ**

**Α\_Τ(Χ) Απόλυτη τιμή του Χ**

**70 / 131**

|  |  |
| --- | --- |
| **7.7** | **Αριθμητικές εκφράσεις** |

**Όταν μια τιμή προκύπτει από υπολογισμό, τότε αναφερόμαστε σε εκφράσεις (expressions). Για τη σύνταξη μιας αριθμητικής έκφρασης χρησιμοποιούνται αριθμητικές σταθερές, μεταβλητές, συναρτήσεις, αριθμητικοί τελεστές και παρενθέσεις. Οι αριθμητικές εκφράσεις υλοποιούν απλές ή σύνθετες μαθηματικές πράξεις.**

**Κάθε έκφραση παριστάνει μια συγκεκριμένη αριθμητική τιμή, η οποία βρίσκεται μετά την εκτέλεση των πράξεων. Γι' αυτό είναι απαραίτητο όλες οι μεταβλητές, που εμφανίζονται σε μια έκφραση να έχουν οριστεί προηγούμενα, δηλαδή να έχουν κάποια τιμή.**

**Ιεραρχία**

**Οι πράξεις που παρουσιάζονται σε μια έκφραση, εκτελούνται σύμφωνα με την επόμενη ιεραρχία**

1. **Ύψωση σε δύναμη**
2. **Πολλαπλασιασμός και διαίρεση**
3. **Πρόσθεση και αφαίρεση**

**71 / 131-132**

**Παραδείγματα**

|  |  |  |
| --- | --- | --- |
| **Μαθηματικά** |  | **ΓΛΩΣΣΑ** |
| **a + 1** |  | **a + 1** |
| **½ a3** |  | **½ \* a ^3** |
|  |  | **(3\*x + 2\*y) / (a-b)** |
| **2 ημx** |  | **2 \* ΗΜ(x)** |

|  |  |
| --- | --- |
|  | **Πάντα πρέπει να χρησιμοποιούνται ζεύγη παρενθέσεων. Διαφορετικός αριθμός αριστερών από δεξιές παρεν-θέσεις στην ίδια έκφραση είναι ένα από τα πιο συνηθισμένα λάθη.** |

**Όταν η ιεραρχία είναι ίδια, τότε οι πράξεις εκτελούνται από τ' αριστερά προς τα δεξιά. Σε πολλές όμως περιπτώσεις είναι απαραίτητο να προηγηθεί μια πράξη χαμηλότερης ιεραρχίας. Αυτό επιτυγχάνεται με την εισαγωγή των παρενθέσεων. Η πράξη που πρέπει να προηγηθεί περικλείεται σε ένα ζεύγος παρενθέσεων, οπότε και εκτελείται πρώτη. Π.χ. η έκφραση 2 + 3\*4 δίδει ως αποτέλεσμα 14, ενώ η (2 + 3)\*4 δίδει 20, διότι εκτελείται πρώτα η πρόσθεση και μετά ο πολλαπλασιασμός.**

**72 / 132**

|  |  |
| --- | --- |
| **7.8** | **Εντολή εκχώρησης** |

**Η εντολή εκχώρησης χρησι-μοποιείται για την απόδοση τιμών στις μεταβλητές κατά τη διάρκεια εκτέλεσης του προγράμματος.**

**Σύνταξη**

**Όνομα μεταβλητής 🡨 έκφραση**

**Παραδείγματα**

**Α 🡨 132  
 ΜΗΝΑΣ 🡨 ‘Ιανουάριος’  
 ΕΜΒΑΔΟΝ 🡨 A \* B**

**Λειτουργία**

**Υπολογίζεται η τιμή της έκφρασης στη δεξιά πλευρά και εκχωρείται η τιμή αυτή στη μεταβλητή, που αναφέρεται στην αριστερή πλευρά.**

**Μια εντολή εκχώρησης σε καμία περίπτωση δεν πρέπει να εκλαμβάνεται ως εξίσωση. Στην εξίσωση το αριστερό μέλος ισούται με το δεξιό, ενώ στην εντολή εκχώρησης η τιμή του δεξιού μέλους εκχωρείται, μεταβιβάζεται, αποδίδεται στη μεταβλητή του αριστερού μέλους. Για το λόγο αυτό ως τελεστής εκχώρησης χρησιμοποιείται το σύμβολο 🡨 προκειμένου να διαφοροποιείται από το ίσον ( = ). Ωστόσο, ας σημειωθεί, ότι οι διάφορες γλώσσες προγραμματισμού χρησιμοποιούν διαφορετικά σύμβολα για το σκοπό αυτό.**

**73 / 132**

|  |  |
| --- | --- |
| **τρομακτικο.jpg** | **Σε μια εντολή εκχώρησης η μεταβλητή και η έκφραση πρέπει να είναι του ιδίου τύπου.** |

|  |  |
| --- | --- |
| **7.9** | **Εντολές εισόδου-εξόδου** |

**Σχεδόν όλα τα προγράμματα υπολογιστή δέχονται κάποια δεδομένα, τα επεξεργάζονται, υπολογίζουν τα αποτελέσματα και τέλος τα εμφανίζουν.**

**Τα δεδομένα εισάγονται κατά τη διάρκεια της εκτέλεσης του προ-γράμματος από μία μονάδα εισόδου, για παράδειγμα το πληκτρολόγιο και τα αποτελέσματα γράφονται σε μία μονάδα εξόδου, για παράδειγμα την οθόνη.**

**Η ΓΛΩΣΣΑ υποστηρίζει για την εισαγωγή δεδομένων από το πληκτρολόγιο την εντολή ΔΙΑΒΑΣΕ και για την εμφάνιση των αποτελεσμάτων την εντολή ΓΡΑΨΕ.**

**74 / 132-133**

**Σύνταξη**

**ΔΙΑΒΑΣΕ λίστα μεταβλητών**

**Παραδείγματα**

**ΔΙΑΒΑΣΕ Ποσότητα, Τιμή**

**Λειτουργία**

**Η εκτέλεση της εντολής οδηγεί στην είσοδο τιμών από το πληκτρολόγιο και την εκχώρησή τους στις μεταβλητές που αναφέρονται.**

**Η εντολή ΔΙΑΒΑΣΕ ακολουθείται πάντοτε από ένα ή περισσότερα ονόματα μεταβλητών. Αν υπάρχουν περισσότερες από μία μεταβλητές τότε αυτές χωρίζονται με κόμμα (,). Κατά την εκτέλεση του προγράμ-ματος η εντολή ΔΙΑΒΑΣΕ διακόπτει την εκτέλεσή του και το πρόγραμμα περιμένει την εισαγωγή από το πληκτρολόγιο τιμών, που θα εκχωρηθούν στις μεταβλητές. Μετά την ολοκλήρωση της εντολής η εκτέλεση του προγράμματος συνεχίζεται με την επόμενη εντολή.**

**Σύνταξη**

**ΓΡΑΨΕ λίστα στοιχείων**

**Παραδείγματα**

**ΓΡΑΨΕ ‘Η τετραγωνική ρίζα του’, A, ‘ είναι: ’, ΡΙΖΑ**

**75 / 133**

**Λειτουργία**

**Χρησιμοποιείται για την εμφάνιση σταθερών τιμών καθώς και των τιμών των μεταβλητών που αναφέρονται στη λίστα.**

**Η εντολή ΓΡΑΨΕ έχει ως αποτέλεσμα την εμφάνιση τιμών στη μονάδα εξόδου. Συσκευή εξόδου μπορεί να είναι η οθόνη του υπολογιστή, ο εκτυπωτής, βοηθητική μνήμη ή γενικά οποιαδήποτε συσκευή εξόδου έχει οριστεί στο πρόγραμμα. Για τα παραδείγματα αυτού του κεφαλαίου θεωρούμε ότι η εμφάνιση γίνεται πάντοτε στην οθόνη. Η λίστα των στοιχείων μπορεί να περιέχει σταθερές τιμές και ονόματα μεταβλητών.**

**Κατά την εκτέλεση του προγράμματος η εντολή ΓΡΑΨΕ προκαλεί την εμφάνιση στην οθόνη των σταθερών τιμών. Όταν κάποιο όνομα μεταβλητής περιέχεται στη λίστα τότε αρχικά ανακτάται η τιμή της και στη συνέχεια η τιμή αυτή εμφανίζεται στην οθόνη.**

**Η χρήση της εντολής ΓΡΑΨΕ είναι κυρίως η εμφάνιση μηνυμάτων από τον υπολογιστή, καθώς και αποτελεσμάτων που περιέχονται στις μεταβλητές.**

|  |  |
| --- | --- |
| **7.10** | **Δομή προγράμματος** |

**Όπως κάθε εντολή ακολουθεί αυστηρούς συντακτικούς κανόνες, έτσι και ολόκληρο το πρόγραμμα έχει αυστη-ρούς κανόνες για τον τρόπο που δομείται. Η πρώτη εντολή κάθε προγράμματος είναι υποχρεωτικά η επικεφαλίδα του προγράμματος, η οποία είναι η λέξη ΠΡΟΓΡΑΜΜΑ ακολουθούμενη από το όνομα του προγράμματος. Το τελευταίο πρέπει να υπακούει στους κανόνες δημιουργίας ονομάτων της ΓΛΩΣΣΑΣ.**

**76 / 133-134**

**Στη συνέχεια ακολουθεί ο τμήμα δήλωσης των σταθε-ρών του προγράμματος, αν βέβαια το πρόγραμμά μας χρησιμοποιεί σταθερές.**

**77 / 134**

**Αμέσως μετά είναι το τμήμα δήλωσης μεταβλητών,   
όπου δηλώνονται υποχρεωτικά τα ονόματα όλων των μεταβλητών καθώς και ο τύπος τους.**

**Ακολουθεί το κύριο μέρος του προγράμματος, που περιλαμβάνει όλες τις εκτελέσιμες εντολές. Οι εντολές αυτές περιλαμβάνονται υποχρεωτικά ανάμεσα στις λέξεις ΑΡΧΗ και ΤΕΛΟΣ\_ΠΡΟΓΡΑΜΜΑΤΟΣ.**

**Τέλος αν το πρόγραμμα χρησιμοποιεί διαδικασίες (βλ. κεφ. 10 Τόμος 3 ), αυτές γράφονται μετά το ΤΕΛΟΣ\_ΠΡΟΓΡΑΜΜΑΤΟΣ.**

**Κάθε εντολή γράφεται σε ξεχωριστή γραμμή. Αν μία εντολή πρέπει να συνεχιστεί και στην επόμενη γραμμή, τότε ο πρώτος χαρακτήρας αυτής της γραμμής πρέπει να είναι ο χαρακτήρας &.**

**Αν ο πρώτος χαρακτήρας είναι το θαυμαστικό (!), σημαίνει ότι αυτή η γραμμή περιέχει σχόλια και όχι εκτελέσιμες εντολές.**

**ΠΑΡΑΔΕΙΓΜΑ**

**Το επόμενο πρόγραμμα υπολογίζει το συνολικό κόστος παραγγελιών υπολογιστών. Το πρόγραμμα διαβάζει από το πληκτρολόγιο την ποσότητα της παραγγελίας και την τιμή του ενός υπολογιστή, υπολογίζει και γράφει το συνολικό κόστος καθώς και το αντίστοιχο κόστος του ΦΠΑ. Ο συντελεστής ΦΠΑ είναι 18%.**

ΠΡΟΓΡΑΜΜΑ **Κόστος\_Υπολογιστών**  
! Πρόγραμμα υπολογισμού κόστους παραγγελίας υπολογιστών

ΣΤΑΘΕΡΕΣ

ΦΠΑ=0.18

ΜΕΤΑΒΛΗΤΕΣ

ΑΚΕΡΑΙΕΣ: **Ποσότητα, Τιμή\_μονάδας, Κόστος**

ΠΡΑΓΜΑΤΙΚΕΣ: **Αξία\_ΦΠΑ, Συνολικό\_Κόστος**

ΑΡΧΗ  
! Εισαγωγή δεδομένων

ΓΡΑΨΕ **‘Δώσε την ποσότητα της παραγγελίας’**

ΔΙΑΒΑΣΕ **Ποσότητα**

ΓΡΑΨΕ **‘Δώσε την τιμή του υπολογιστή’**

ΔΙΑΒΑΣΕ **Τιμή\_μονάδας**

! Υπολογισμοί  
**Κόστος 🡨 Ποσότητα \* Τιμή\_μονάδας**

**Αξία\_ΦΠΑ 🡨 Κόστος \* ΦΠΑ**

**Συνολικό\_Κόστος 🡨 Κόστος + Αξία\_ΦΠΑ**

! Εμφάνιση αποτελεσμάτων  
ΓΡΑΨΕ **‘Το κόστος των ’, Ποσότητα, ‘υπολ. είναι ’, Κόστος**

ΓΡΑΨΕ **‘Η αξία του ΦΠΑ είναι ’, Αξία\_ΦΠΑ**

ΓΡΑΨΕ **‘Το συνολικό κόστος είναι ’, Συνολικό\_Κόστος**

ΤΕΛΟΣ\_ΠΡΟΓΡΑΜΜΑΤΟΣ

Στη συνέχεια παρουσιάζεται το πρόγραμμα αυτό σε γλώσσες προγραμματισμού Pascal και Basic.

**78 / 134-135**

Προγραμματιστικό περιβάλλον Pascal

PROGRAM computers;

CONST  
 fpa=0.18  
VAR   
 cost, value, quantity: INTEGER;  
 total, cost\_fpa: REAL;

**BEGIN**

**write(‘Δώσε την παραγγελία’);**

**readln(quantity);**

**write(‘Δώσε την τιμή του υπολογιστή’);**

**readln(value);**

**cost:= quantity \* value;**

**cost\_fpa:= cost\*fpa;**

**total:= cost + cost\_fpa;**

**writeln(‘Το κόστος των ’, quantity, ‘ είναι:’, cost);**

**writeln** **(‘Η αξία του ΦΠΑ : ’, cost\_fpa:7:0);**  
 **writeln** **(‘Το συνολικό κόστος είναι: ’, total:7:0);**  
**END.**

Προγραμματιστικό περιβάλλον Basic

‘ Κόστος υπολογιστών

fpa = .18

INPUT “Δώσε την ποσότητα : ”, Quantity

INPUT “Δώσε την τιμή του υπολογιστή : ”,   
Value

Cost = Quantity \* Value

CostFpa = Cost \* fpa

Total=Cost + CostFpa

PRINT “Το κόστος των”; Quantity; “υπολογιστών

είναι : ”; Cost

PRINT USING “ Η αξία ΦΠΑ είναι : #####”; CostFpa

PRINT USING “Το συνολικό κόστος είναι : #####”; Total

END

**79 / 135**

**• Ανακεφαλαίωση**

**Σε αυτό το κεφάλαιο παρουσιάστηκαν τα βασικά χαρακτηριστικά της ΓΛΩΣΣΑΣ, το αλφάβητό της, οι τύποι δεδομένων που υποστηρίζει, οι κανόνες για τα ονόματα που χρησιμοποιούνται, οι αριθμητικές πράξεις, η εντολή εκχώρησης, οι εντολές εισόδου και εξόδου καθώς και η δομή που πρέπει να έχει κάθε πρόγραμμα.**

**Συγκεκριμένα:**

* **Οι τύποι δεδομένων που υποστηρίζονται είναι: Ακέραιοι, Πραγματικοί, Χαρακτήρες, Λογικοί.**
* **Οι μεταβλητές πρέπει να έχουν τον τύπο των δεδομένων που καταχωρούν.**
* **Κάθε μεταβλητή παίρνει τιμή με εντολή εκχώρησης ή με εντολή ΔΙΑΒΑΣΕ.**
* **Κάθε πρόγραμμα έχει τον τίτλο του, ακολουθεί το τμήμα δηλώσεων (σταθερών και μεταβλητών) και μετά ανάμεσα στις λέξεις ΑΡΧΗ και ΤΕΛΟΣ ΠΡΟΓΡΑΜΜΑΤΟΣ βρίσκονται όλες οι εκτελέσιμες εντολές.**
* **Η επικοινωνία του προγράμματος με τον χρήστη γίνεται με τις εντολές εισόδου εξόδου ΓΡΑΨΕ και ΔΙΑΒΑΣΕ.**

**Λέξεις κλειδιά**

**Πρόγραμμα, Τύποι δεδομένων, Μεταβλητή, Σταθερά,   
Εντολή, Εκχώρηση τιμής, Είσοδος- έξοδος προγράμματος**

**80 /136**

**• Ερωτήσεις Θέματα για συζήτηση**

1. **Ποιους τύπους δεδομένων γνωρίζετε. Αναφέρατε δύο παραδείγματα για κάθε τύπο.**
2. **Σε ποια θέση του προγράμματος αναγράφονται οι δηλώσεις των σταθερών;**
3. **Ποια η διαφορά μεταβλητών και σταθερών;**
4. **Ποια η σειρά εκτέλεσης των πράξεων;**
5. **Ποιος ο σκοπός των εντολών εισόδου εξόδου;**
6. **Ποια η διαφορά των εντολών ΔΙΑΒΑΣΕ και ΓΡΑΨΕ;**
7. **Περιγράψτε τη δομή ενός προγράμματος.**

**• Βιβλιογραφία**

1. **Θ. Αλεβίζος, Α. Καμπουρέλης, Εισαγωγή με τη γλώσσα Pascal, Αθήνα, 1984.**
2. **Γ. Βουτυράς, Basic: Αλγόριθμοι και εφαρμογές, Κλεψύδρα, Αθήνα, 1991.**
3. **Χρ. Κοιλίας, Η QuickBasic και οι εφαρμογές της, Εκδόσεις Νέων Τεχνολογιών, Αθήνα, 1992.**
4. **R. Shackelford, Introduction to Computing and Algorithms, Addison-Wesley, USA, 1998.**
5. **S. Leestma-L.Nyhoff, Turbo Pascal, Programming and Solving, McMillan, New York, 1990.**
6. **N. Wirth, Systematic Programming: An introduction, Prentice Hall, 1973.**

**81 / 136**

**Διαδίκτυο**

[**http://qbasic.com/**](http://qbasic.com/)

**Περιέχει εκπαιδευτικό οδηγό, κώδικα πολλών ασκήσεων και γενικές πληροφορίες για την Qbasic.**

[**www.basicguru.com**](http://digitalschool.minedu.gov.gr/modules/ebook/show.php/DSGL-C101/36/198,1065/www.basicguru.com)

**Διεύθυνση που αναφέρεται αποκλειστικά στη Basic. Περιέχει πολλά έτοιμα παραδείγματα, πληροφορίες για εκδόσεις της γλώσσας, μεταφραστές για διάφορα λειτουργικά συστήματα.**

**82 / 136**

**83 / 137**

**Κεφάλαιο 8**

**Επιλογή και Επανάληψη**

**ΠΕΡΙΕΧΟΜΕΝΑ**

|  |  |
| --- | --- |
|  | **Εντολές επιλογής** |
|  | **Εντολές επανάληψης** |

**• Εισαγωγή**

**Στο προηγούμενο κεφάλαιο αναπτύξαμε προγράμματα, τα οποία ήταν πολύ απλά και οι εντολές των οποίων εκτελούνται η μία μετά την άλλη. Αυτή η σειριακή εκτέλεση των εντολών είναι κατάλληλη όμως μόνο για πολύ απλά προγράμματα,τα οποία εισάγουν δεδομένα, τα επεξεργάζονται και τυπώνουν το αποτέλεσμα, χωρίς να υπάρχει η δυνατότητα της επιλεκτικής εκτέλεσης τμημάτων του προγράμματος, σύμφωνα με την τιμή κάποιων δεδομένων ή την επανάληψη τμημάτων του προγράμματος. Όπως έχουμε αναφέρει οι τρεις βασικές δομές, είναι η δομή της ακολουθίας, της επιλογής και της επανάληψης. Οι δομές αυτές αποτελούν τη βάση του δομημένου προγραμματισμού και με τη χρήση αυτών μπορούν να υλοποιηθούν όλα τα προγράμματα υπολογιστών. Στο κεφάλαιο αυτό θα ασχοληθούμε με τις δύο αυτές βασικές δομές της επιλογής και της επανάληψης που θα μας επιτρέψουν την συγγραφή πληρέστερων και πιο πολύπλοκων προγραμμάτων.**

**84 / 138**

**• Διδακτικοί στόχοι**

**Να είναι σε θέση ο μαθητής:**

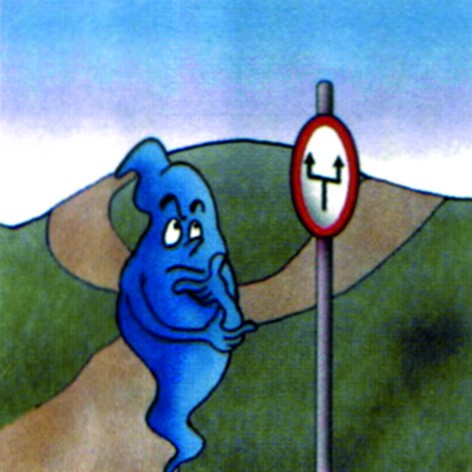
* **Να σχηματίζει λογικές εκφράσεις, απλές και σύνθετες.**
* **Να διατυπώνει τις μορφές της εντολής ελέγχου (επιλογής) ΑΝ.**
* **Να διακρίνει τις διαφορές των μορφών της εντολής ΑΝ.**
* **Να επιλέγει την καλύτερη μορφή της εντολής ΑΝ για το κάθε πρόγραμμα.**
* **Να διατυπώνει τις εντολές επανάληψης.**
* **Να επιλέγει την καλύτερη δομή επανάληψης και να χρησιμοποιεί την κατάλληλη εντολή.**
* **Να συντάσσει προγράμματα τα οποία χρησιμο-ποιούν και τις τρεις βασικές δομές: της ακολουθίας, της επιλογής και της επανάληψης.**

**• Προερωτήσεις**

* **Γιατί χρησιμοποιούνται οι αλγοριθμικές δομές;**
* **Νομίζεις ότι όλες οι αλγοριθμικές δομές έχουν τις αντίστοιχες εντολές σε μία γλώσσα προγραμ-ματισμού;**
* **Η δομή της επιλογής είναι σημαντική για την επίλυση προβλημάτων;**
* **Αρκεί μία εντολή για να εκφράσει την δομή της επανάληψης;**
* **Πώς μπορεί να ελέγχεται ο τερματισμός μίας επανάληψης;**

**85 / 138**

|  |  |
| --- | --- |
| **8.1** | **Εντολές Επιλογής** |

**Μία από τις βασικότερες δομές που εμφανίζονται σε ένα πρόγραμμα, είναι η επιλογή. Σχεδόν σε όλα τα προβλήματα περιλαμβάνονται κάποιοι έλεγχοι και ανάλογα με το αποτέλεσμα αυτών των ελέγχων επιλέγονται οι ενέργειες που θα ακολουθήσουν.**

**Ας θεωρήσουμε το πολύ απλό πρόβλημα της κατά-μέτρησης των θετικών και των αρνητικών αριθμών. Πρέπει λοιπόν να γράψουμε ένα πρόγραμμα, το οποίο εισάγει αριθμούς και μετράει πόσοι από αυτούς είναι θετικοί και πόσοι αρνητικοί. Για να αποφασίσουμε, αν ένας αριθμός είναι θετικός ή αρνητικός, πρέπει να τον συγκρίνουμε με το 0. Το αποτέλεσμα αυτής της σύγκρισης καθορίζει το είδος του αριθμού, αν είναι μεγαλύτερος από το 0, τότε ο αριθμός είναι θετικός, ενώ αντίθετα αν είναι μικρότερος από το 0, είναι αρνητικός.**

**Λογική Έκφραση**

**Για τη σύνταξη μιας λογικής έκφρασης ή συνθήκης χρησιμοποιούνται σταθερές, μεταβλητές, αριθμητικές παραστάσεις, συγκριτικοί και λογικοί τελεστές, καθώς και παρενθέσεις. Στις λογικές εκφράσεις γίνεται σύγκριση της τιμής μίας έκφρασης, που βρίσκεται αριστερά από το συγκριτικό τελεστή με την τιμή μιας άλλης έκφρασης που βρίσκεται δεξιά. Το αποτέλεσμα είναι μία λογική τιμή ΑΛΗΘΗΣ ή ΨΕΥΔΗΣ.**

**86 / 139**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  |  | | --- | --- | --- | | **Συγκριτικοί τελεστές**  **000 / 000**  **87 / 139** | | | | **Τελεστής** | **Ελεγχόμενη σχέση** | **Παράδειγμα** | | **=** | **Ισότητα** | **Αριθμός=0** | | **< >** | **Ανισότητα** | **Ονομα1 < > 'Κώστας'** | | **>** | **Μεγαλύτερο από** | **Τιμή > 10000** | | **> =** | **Μεγαλύτερο ή ίσο** | **Χ+Υ >= (Α+Β)/Γ** | | **<** | **Μικρότερο από** | **ΒΛ2-4\*Α\*Γ < 0** | | **< =** | **Μικρότερο ή ίσο** | **Βάρος <= 500** |   **Οι χρησιμοποιούμενοι συγκριτικοί τελεστές παρουσιάζονται στον επόμενο πίνακα** | |

**Οι συγκρίσεις γίνονται σε δεδομένα αριθμητικά, αλφαριθμητικά και λογικά.**

**Η σύγκριση μεταξύ δύο αριθμών γίνεται με προφανή τρόπο. Στην περίπτωση των πραγματικών αριθμών θεωρούμε ότι οι αριθμοί μπορούν να έχουν άπειρο αριθμό ψηφίων.**

**Η σύγκριση ατομικών χαρακτήρων στηρίζεται στην αλφαβητική σειρά, για παράδειγμα το 'α' θεωρείται μικρότερο από το 'β'.**

**Η σύγκριση αλφαριθμητικών δεδομένων βασίζεται στη σύγκριση χαρακτήρα προς χαρακτήρα σε κάθε θέση μέχρις ότου βρεθεί κάποια διαφορά, για παράδειγμα η λέξη 'κακός' θεωρείται μικρότερη από τη λέξη 'καλός' αφού το γράμμα κ προηγείται του γράμματος λ.**

**Η σύγκριση λογικών έχει έννοια μόνο στην περίπτωση του ίσου ( = ) και του διάφορου (< >), αφού οι τιμές που μπορούν να έχουν είναι ΑΛΗΘΗΣ και ΨΕΥΔΗΣ.**

**88 / 139**

|  |  |
| --- | --- |
|  | **Όταν αριθμητικοί και συγκριτικοί τελεστές συνδυάζονται σε μια έκφραση, οι αριθμητικές πράξεις εκτελούνται πρώτες. Ακόμη, οι λογικοί τελεστές έχουν χαμηλότερη ιεραρχία από τους συγκριτικούς.** |

**Σύνθετες Εκφράσεις**

**Σε πολλά προβλήματα οι επιλογές δεν αρκεί να γίνονται με απλές λογικές παραστάσεις όπως αυτές οι οποίες αναφέρθηκαν, αλλά χρειάζεται να συνδυαστούν με μια ή περισσότερες λογικές παραστάσεις. Αυτό επιτυγχάνεται με τη χρήση των τριών βασικών λογικών τελεστών ΟΧΙ, ΚΑΙ, Ή.**

**Παραδείγματα**

**0<Χ<5 Χ>0 ΚΑΙ Χ<5**

**Χ=1 ή 2 ή 3 Χ=1 Ή Χ=2 Ή Χ=3**

**Η ιεραρχία των λογικών τελεστών είναι μικρότερη των αριθμητικών.**

**8.1.1 Εντολή AN**

**Η δομή επιλογής υλοποιείται στη ΓΛΩΣΣΑ με την εντολή ΑΝ. Η εντολή ΑΝ εμφανίζεται με τρεις διαφορετικές μορφές. Την απλή εντολή ΑΝ...ΤΟΤΕ, την εντολή ΑΝ...ΤΟΤΕ..ΑΛΛΙΩΣ και τέλος την εντολή ΑΝ...ΤΟΤΕ..ΑΛΛΙΩΣ\_ΑΝ. Κάθε εντολή ΑΝ πρέπει να κλείνει με ΤΕΛΟΣ\_ΑΝ.**

**Στην απλούστερη μορφή της η εντολή ΑΝ ελέγχει τη συνθήκη και αν αυτή ισχύει (είναι αληθής), τότε εκτελούνται οι εντολές που περιλαμβάνονται μεταξύ των λέξεων ΤΟΤΕ και ΤΕΛΟΣ\_ΑΝ.**

**89 / 140**

**Αν για παράδειγμα θέλουμε να υπολογίσουμε την τετραγωνική ρίζα των αριθμών που διαβάζουμε από το πληκτρολόγιο, τότε το αντίστοιχο τμήμα προγράμματος είναι:**

ΔΙΑΒΑΣΕ **α**

ΑΝ **α >= 0** ΤΟΤΕ  
      **Ρίζα** 🡨 **Τ\_Ρ(α)**  
ΤΕΛΟΣ\_ΑΝ

**Η γενική μορφή της εντολής ΑΝ είναι η εξής:**

**Σύνταξη**

ΑΝ **συνθήκη** ΤΟΤΕ **εντολή-1**

**εντολή-2**

**…**

**εντολή-ν**

ΤΕΛΟΣ\_ΑΝ

**Παράδειγμα**

ΑΝ **αριθμός > 0** ΤΟΤΕ **ΓΡΑΨΕ ‘Ο αριθμός είναι θετικός’**

**Πλήθος\_θετικών** 🡨**Πλήθος\_θετικών +1**

ΤΕΛΟΣ\_ΑΝ

**Λειτουργία**

**Αν η συνθήκη ισχύει, τότε εκτελούνται οι εντολές που βρίσκονται μεταξύ των λέξεων ΤΟΤΕ και ΤΕΛΟΣ\_ΑΝ, σε αντίθετη περίπτωση οι εντολές αυτές αγνοούνται. Η εκτέλεση του προγράμματος συνεχίζεται με την εντολή που ακολουθεί τη δήλωση ΤΕΛΟΣ\_ΑΝ.**

**90 / 140**

**Συχνά η εντολή ΑΝ εκτός από το τμήμα των εντολών, που εκτελούνται όταν η λογική έκφραση είναι Αληθής, περιέχει και το τμήμα των εντολών που εκτελούνται, αν δεν ισχύει η συνθήκη (είναι Ψευδής).**

**91 / 141**

**Η μορφή αυτής της εντολής ονομάζεται ΑΝ...ΤΟΤΕ...ΑΛΛΙΩΣ,**

**Στο παράδειγμα του υπολογισμού της τετραγωνικής ρίζας έχουμε**

ΔΙΑΒΑΣΕ **α**

ΑΝ **α >= 0** ΤΟΤΕ  
      **Ρίζα** 🡨 **Τ\_Ρ(α)**  
ΑΛΛΙΩΣ

ΓΡΑΨΕ ‘Η τετρ. ρίζα αρνητικού αριθμού δεν ορίζεται’

ΤΕΛΟΣ\_ΑΝ

**Η γενική μορφή της εντολής ΑΝ...ΤΟΤΕ...ΑΛΛΙΩΣ έχει ως εξής:**

**Σύνταξη**

ΑΝ **συνθήκη** ΤΟΤΕ **εντολή-1**

**εντολή-2**

**…**

**εντολή-ν**

ΑΛΛΙΩΣ

**εντολή-1**

**εντολή-2**

**…**

**εντολή-ν** ΤΕΛΟΣ\_ΑΝ

**Παράδειγμα**

ΑΝ **αριθμός > 0** ΤΟΤΕ **ΓΡΑΨΕ ‘Ο αριθμός είναι θετικός’**

**Πλήθος\_θετικών** 🡨**Πλήθος\_θετικών +1**

ΑΛΛΙΩΣ

**ΓΡΑΨΕ ‘Ο αριθμός είναι αρνητικός ή μηδέν’**

**Πλ\_μη\_θετικών** 🡨 Π**λ\_μη\_θετικών +1**

ΤΕΛΟΣ\_ΑΝ

**Λειτουργία**

**Αν η συνθήκη ισχύει, τότε εκτελούνται οι εντολές που βρίσκονται μεταξύ των λέξεων ΤΟΤΕ και ΑΛΛΙΩΣ, δια-φορετικά εκτελούνται οι εντολές μεταξύ ΑΛΛΙΩΣ και   
ΤΕΛΟΣ\_ΑΝ. Η εκτέλεση του προγράμματος συνεχίζεται με την εντολή που ακολουθεί τη δήλωση ΤΕΛΟΣ\_ΑΝ.**

**Η γενική μορφή της εντολής ΑΝ καλύπτει την επιλογή μιας από δύο εναλλακτικές περιπτώσεις.**

**Όταν οι εναλλακτικές περιπτώσεις είναι περισσότερες από τις δύο, τότε μπορούν να χρησιμοποιηθούν πολλές εντολές ΑΝ η μία μέσα στην άλλη, οι εμφωλευμένες εντολές ΑΝ, όπως ονομάζονται.**

**ΟΡΙΣΜΟΣ**

**Εμφωλευμένα ΑΝ ονομάζονται δύο ή περισσότερες εντολές της μορφής ΑΝ...ΤΟΤΕ...ΑΛΛΙΩΣ που περιέχονται η μία μέσα στην άλλη.**

**92 / 141-142**

**Για παράδειγμα οι παρακάτω εντολές προγράμματος**

,,,,,,

ΔΙΑΒΑΣΕ **Βάρος, Ύψος**

ΑΝ**Βάρος < 80** ΤΟΤΕ **Ρίζα**🡨**Τ\_Ρ(α)** ΑΝ**Ύψος < 1.70** ΤΟΤΕΓΡΑΨΕ ‘Ελαφρύς, Κοντός’

ΤΕΛΟΣ\_ΑΝ

ΤΕΛΟΣ\_ΑΝ

,,,,,,

**Η χρήση εμφωλευμένων εντολών ΑΝ οδηγεί συνήθως σε πολύπλοκες δομές που αυξάνουν την πιθανότητα του λάθους καθώς και τη δυσκολία κατανόησης του προγράμματος.**

**Πολύ συχνά οι εντολές που έχουν γραφεί με εμφωλευμένα ΑΝ, μπορούν να γραφούν πιο απλά χρησιμοποιώντας σύνθετες εκφράσεις ή την εντολή επιλογής ΑΝ ... ΤΟΤΕ ... ΑΛΛΙΩΣ\_ΑΝ, που θα παρουσιαστεί στη συνέχεια.**

**Το προηγούμενο τμήμα προγράμματος μπορεί να γραφεί ως εξής**

,,,,,,

ΔΙΑΒΑΣΕ**Βάρος, Ύψος**

ΑΝ**Βάρος < 80** ΚΑΙ **Ύψος < 1.70** ΤΟΤΕΓΡΑΨΕ ‘Ελαφρύς, Κοντός’

ΤΕΛΟΣ\_ΑΝ

,,,,,,

**93 / 142**

**Μία άλλη μορφή επιλογής είναι η εντολή ΑΝ...ΤΟΤΕ...ΑΛΛΙΩΣ\_ΑΝ**

**Σύνταξη**

ΑΝ **συνθήκη-1** ΤΟΤΕ **εντολή-1**

**εντολή-2**

**…**

**εντολή-ν**

ΑΛΛΙΩΣ\_ΑΝ **συνθήκη-2** ΤΟΤΕ

**εντολή-1**

**εντολή-2**

**…**

**εντολή-ν**

ΑΛΛΙΩΣ

**εντολή-1**

**εντολή-2**

**…**

**εντολή-ν** ΤΕΛΟΣ\_ΑΝ

**Παράδειγμα**

ΑΝ **αριθμός > 0** ΤΟΤΕ **ΓΡΑΨΕ ‘Ο αριθμός είναι θετικός’**

**Πλήθος\_θετικών** 🡨**Πλήθος\_θετικών +1**

ΑΛΛΙΩΣ\_ ΑΝ **αριθμός < 0** ΤΟΤΕ

**ΓΡΑΨΕ ‘Ο αριθμός είναι αρνητικός’**

**Πλ\_αρνητικών** 🡨 Π**λ\_αρνητικών +1**

ΑΛΛΙΩΣ

**ΓΡΑΨΕ ‘Ο αριθμός είναι 0’**

**Πλήθος\_0**🡨 **Πλήθος\_0+1**

ΤΕΛΟΣ\_ΑΝ

**94 / 142-143**

**Λειτουργία**

**Εκτελούνται οι εντολές που βρίσκονται στο αντίστοιχο τμήμα, όταν η συνθήκη είναι αληθής. Η εκτέλεση του προγράμματος συνεχίζεται με την εντολή που ακολουθεί τη δήλωση ΤΕΛΟΣ\_ΑΝ.**

**ΠΑΡΑΔΕΙΓΜΑ 1**

**Στο πρόγραμμα του προηγούμενου κεφαλαίου (πωλήσεις υπολογιστών) υποθέτουμε ότι η τιμή των υπολογιστών εξαρτάται από την ποσότητα παραγγελίας. Συγκεκριμένα ισχύουν οι παρακάτω τιμές αγοράς υπολογιστών.**

|  |  |
| --- | --- |
| **ΠΟΣΟΤΗΤΑ** | **ΤΙΜΗ ΜΟΝΑΔΑΣ** |
| **1-50** | **580** |
| **51-100** | **520** |
| **101-200** | **470** |
| **πάνω από 200** | **440** |

**Ο υπολογισμός με χρήση εμφωλευμένων εντολών ΑΝ είναι:**

**95 / 143**

ΑΝ **Ποσότητα <= 50** ΤΟΤΕ  
      **Κόστος** 🡨 **Ποσότητα \* 580**  
ΑΛΛΙΩΣ

ΑΝ **Ποσότητα <= 100** ΤΟΤΕ  
 **Κόστος** 🡨 **Ποσότητα \* 520**

ΑΛΛΙΩΣ

ΑΝ **Ποσότητα <= 200** ΤΟΤΕ  
 **Κόστος** 🡨 **Ποσότητα \* 470**

ΑΛΛΙΩΣ

**Κόστος** 🡨 **Ποσότητα \* 440**

ΤΕΛΟΣ\_ΑΝ

ΤΕΛΟΣ\_ΑΝ

ΤΕΛΟΣ\_ΑΝ

**Τo ίδιο πρόγραμμα με τη χρήση της εντολής ΑΝ...ΤΟΤΕ...ΑΛΛΙΩΣ\_ΑΝ έχει ως εξής:**

ΑΝ **Ποσότητα <= 50** ΤΟΤΕ  
      **Κόστος** 🡨 **Ποσότητα \* 580**  
ΑΛΛΙΩΣ\_ΑΝ **Ποσότητα <= 100** ΤΟΤΕ  
 **Κόστος** 🡨 **Ποσότητα \* 520**

ΑΛΛΙΩΣ\_ΑΝ **Ποσότητα <= 200** ΤΟΤΕ  
 **Κόστος** 🡨 **Ποσότητα \* 470**

ΑΛΛΙΩΣ   
 **Κόστος** 🡨 **Ποσότητα \* 440**

ΤΕΛΟΣ\_ΑΝ

|  |  |
| --- | --- |
| **96 / 143-144** | **Να αποφεύγεται, αν είναι δυνατόν,  η χρήση των εμφωλευμένων ΑΝ, και στη θέση τους να χρησιμοποιούνται απλούστερες δομές που διευκολύνουν την ανάγνωση και την κατανόηση του προγράμματος** |

**Ένα συχνό λάθος που παρατηρείται στα προγράμματα είναι ο έλεγχος περιττών συνθηκών. Οι επιπλέον έλεγχοι αυξάνουν την πολυπλοκότητα του προγράμματος.**

**Στο προηγούμενο παράδειγμα για το οποίο θεωρούμε ότι η ποσότητα είναι θετικός αριθμός, ένα παράδειγμα περιττών ελέγχων είναι το ακόλουθο:**

ΑΝ **Ποσότητα <= 50** ΤΟΤΕ  
    **Κόστος** 🡨 **Ποσότητα \* 580**  
ΑΛΛΙΩΣ\_ΑΝ **Ποσότητα > 50** ΚΑΙ **Ποσότητα <= 100** ΤΟΤΕ **Κόστος** 🡨 **Ποσότητα \* 520**

ΑΛΛΙΩΣ\_ΑΝ **Ποσότητα > 100** ΚΑΙ **Ποσότητα <= 200** ΤΟΤΕ

**Κόστος** 🡨 **Ποσότητα \* 470**

ΑΛΛΙΩΣ **Κόστος** 🡨 **Ποσότητα \* 440**

ΤΕΛΟΣ\_ΑΝ

|  |  |
| --- | --- |
| **πιν.jpg** | **Στην πρώτη εντολή ΑΛΛΙΩΣ\_ΑΝ ο έλεγχος της συνθήκης Ποσότητα > 50 είναι περιττός** |

**8.1.2 Εντολή ΕΠΙΛΕΞΕ**

**Αν οι εναλλακτικές περιπτώσεις επιλογής είναι πολλές, μπορεί να χρησιμοποιηθεί η εντολή ΕΠΙΛΕΞΕ, η γενική μορφή της οποίας είναι:**

**97 / 144**

**Σύνταξη**

ΕΠΙΛΕΞΕ **έκφραση**

ΠΕΡΙΠΤΩΣΗ  **λίστα\_τιμών\_1**

**εντολές-1**

ΠΕΡΙΠΤΩΣΗ  **λίστα\_τιμών\_2**

**εντολές-2**

**…**

ΠΕΡΙΠΤΩΣΗ ΑΛΛΙΩΣ

**εντολές-αλλιώς**

ΤΕΛΟΣ\_ΕΠΙΛΟΓΩΝ

**Παράδειγμα**

ΔΙΑΒΑΣΕ **αριθμός**

ΕΠΙΛΕΞΕ **αριθμός**

ΠΕΡΙΠΤΩΣΗ **0  
 ΓΡΑΨΕ ‘Μηδέν’**

ΠΕΡΙΠΤΩΣΗ **1,3,5,7,9  
 ΓΡΑΨΕ ‘Μονός αριθμός’**

ΠΕΡΙΠΤΩΣΗ **2,4,6,8  
 ΓΡΑΨΕ ‘Ζυγός αριθμός’**

ΠΕΡΙΠΤΩΣΗ\_ΑΛΛΙΩΣ

**ΓΡΑΨΕ ‘Αριθμός <0 ή >9 ή   
 όχι ακέραιος’**

ΤΕΛΟΣ\_ΕΠΙΛΟΓΩΝ

**Λειτουργία**

**Υπολογίζεται η τιμή της έκφρασης και εκτελούνται οι εντολές που ανήκουν στην αντίστοιχη περίπτωση τιμών. Αν η τιμή της έκφρασης δεν αντιστοιχεί σε καμία περίπτωση, τότε εκτελούνται οι εντολές αλλιώς.**

**98 / 144-145**

**Στην εντολή αυτή οι λίστες τιμών που συνοδεύουν κάθε περίπτωση μπορούν να περιλαμβάνουν μία ή περισσότερες τιμές ή περιοχή τιμών από - έως.**

**Η χρήση της εντολής ΕΠΙΛΕΞΕ λόγω της συμπαγούς δομής της προσφέρει σημαντικά πλεονεκτήματα στον προγραμματισμό.**

|  |  |
| --- | --- |
| **8.2** | **Εντολές επανάληψης** |

**Η τρίτη βασική δομή είναι η δομή επανάληψης, ο βρόχος, η οποία επιτρέπει την εκτέλεση εντολών περισσότερες από μία φορά. Οι επαναλήψεις ελέγχονται πάντοτε από κάποια συνθήκη, η οποία καθορίζει την έξοδο από το βρόχο.**

**Η ΓΛΩΣΣΑ υποστηρίζει τρεις εντολές επανάληψης,**

**την εντολή ΟΣΟ όπου η επανάληψη ελέγχεται από μία λογική έκφραση στην αρχή και εκτελείται συνεχώς όσο η συνθήκη είναι Αληθής, την εντολή ΜΕΧΡΙΣ\_ΟΤΟΥ όπου η συνθήκη βρίσκεται στο τέλος του βρόχου και εκτελείται συνεχώς μέχρις ότου η συνθήκη αυτή γίνει Αληθής και τέλος την εντολή ΓΙΑ, με την οποία ο βρόχος επαναλαμβάνεται για προκαθορισμένο αριθμό φορών.**

**8.2.1 Εντολή ΟΣΟ...ΕΠΑΝΑΛΑΒΕ**

**Η γενικότερη δομή επανάληψης υλοποιείται στη ΓΛΩΣΣΑ με την εντολή ΟΣΟ... ΕΠΑΝΑΛΑΒΕ. Σε αυτή, η συνθήκη που ελέγχει την επανάληψη βρίσκεται στην αρχή της επανάληψης και ο βρόχος επαναλαμβάνεται συνεχώς, όσο η συνθήκη αυτή ισχύει. Με τη δομή αυτή μπορούν να εκφραστούν όλες οι επαναλήψεις και γι αυτό η εντολή ΟΣΟ... ΕΠΑΝΑΛΑΒΕ είναι η σημα-ντικότερη από όλες τις εντολές επανάληψης. Χαρακτηριστικό της επανάληψης αυτής είναι ότι ο αριθμός των επαναλήψεων δεν είναι γνωστός, ούτε μπορεί να υπολογιστεί πριν την εκτέλεση του προγράμματος.**

**99 / 145**

**Σύνταξη**

ΟΣΟ **συνθήκη** ΕΠΑΝΑΛΑΒΕ **εντολή-1**

**εντολή-2**

**…**

**εντολή-ν**

ΤΕΛΟΣ\_ΕΠΑΝΑΛΗΨΗΣ

**Παράδειγμα**

**Άθροισμα 🡨 0**

ΟΣΟ **Άθροισμα <1000** ΕΠΑΝΑΛΑΒΕΔΙΑΒΑΣΕ **Α  
 Άθροισμα 🡨 Άθροισμα + Α**

ΤΕΛΟΣ\_ΕΠΑΝΑΛΗΨΗΣ

**Λειτουργία**

**Ελέγχεται η συνθήκη και αν είναι Αληθής, εκτελούνται οι εντολές που βρίσκονται ανάμεσα στις ΟΣΟ… ΕΠΑΝΑΛΑΒΕ και ΤΕΛΟΣ\_ΕΠΑΝΑΛΗΨΗΣ. Στη συνέχεια ελέγχεται πάλι η συνθήκη και αν ισχύει, εκτελούνται πάλι οι ίδιες εντολές. Όταν η λογική έκφραση γίνει Ψευδής, τότε σταματάει η επανάληψη και εκτελείται η εντολή μετά το ΤΕΛΟΣ\_ΕΠΑΝΑΛΗΨΗΣ.**

**100 / 145-146**

**Εφόσον μετά από κάθε επανάληψη ελέγχεται εκ νέου η συνθήκη, πρέπει υποχρεωτικά μέσα στο βρόχο να υπάρχει μία εντολή, η οποία να μεταβάλει την τιμή της μεταβλητής που ελέγχεται με τη συνθήκη. Σε αντίθετη περίπτωση η επανάληψη δε θα τερματίζεται και θα εκτελείται συνεχώς.**

**ΠΑΡΑΔΕΙΓΜΑ 2**

**Να γραφεί πρόγραμμα το οποίο διαβάζει από το πληκτρολόγιο μία σειρά μετρήσεων, ακεραίων μη μηδενικών αριθμών, υπολογίζει και τυπώνει το άθροισμα τους καθώς και το μέσο τους όρο. Ως τέλος της διαδικασίας εισαγωγής στοιχείων χρησιμοποι-είται η τιμή 0.**

ΠΡΟΓΡΑΜΜΑ **Άθροισμα**  
ΜΕΤΑΒΛΗΤΕΣ

ΑΚΕΡΑΙΕΣ: **X,** **Άθροισμα, Πλήθος**  
 ΠΡΑΓΜΑΤΙΚΕΣ: **ΜΟ**

ΑΡΧΗ  
**Πλήθος** 🡨 **0**

**Άθροισμα**🡨 **0**  
ΓΡΑΨΕ **‘Δώσε αριθμό’**  
ΔΙΑΒΑΣΕ **X**  
ΟΣΟ **X <> 0**  ΕΠΑΝΑΛΑΒΕ

**Άθροισμα**🡨 **Άθροισμα** +  **X**

**Πλήθος** 🡨 **Πλήθος** + 1

ΓΡΑΨΕ **‘Δώσε αριθμό’**  
 ΔΙΑΒΑΣΕ **X**  
ΤΕΛΟΣ\_ΕΠΑΝΑΛΗΨΗΣ

**101 / 146-147**

ΑΝ **Πλήθος**  > 0 ΤΟΤΕ

**102 / 147**

**ΜΟ** 🡨 **Άθροισμα / Πλήθος**   
  ΓΡΑΨΕ **‘Το Άθροισμα είναι: ’, Άθροισμα**  
  ΓΡΑΨΕ **‘Ο Μέσος Όρος είναι: ’, ΜΟ**  
ΑΛΛΙΩΣ

  ΓΡΑΨΕ **‘Δεν δόθηκαν στοιχεία’**   
ΤΕΛΟΣ\_ΑΝ  
ΤΕΛΟΣ\_ΠΡΟΓΡΑΜΜΑΤΟΣ

**Παρατηρήσεις**

**Η χρήση τιμών για τον τερματισμό μίας επαναληπτικής διαδικασίας, όπως στο παράδειγμα η αυθαίρετη επιλογή του 0, είναι συνήθης στον προγραμματισμό.**

**Η τιμή αυτή ορίζεται από τον προγραμματιστή και αποτελεί μια σύμβαση για το τέλος του προγράμματος. Η τιμή αυτή είναι τέτοια, ώστε να μην είναι λογικά σωστή για το πρόβλημα, για παράδειγμα η τιμή 0 αποκλείεται από τις μετρήσεις σύμφωνα με την εκφώνηση του παραδείγματος. Η τιμή αυτή συχνά αποκαλείται "τιμή φρουρός".**

**8.2.2 Εντολή ΜΕΧΡΙΣ\_ΟΤΟΥ**

**Η δεύτερη εντολή επανάληψης που χρησιμοποιεί η ΓΛΩΣΣΑ είναι η εντολή ΜΕΧΡΙΣ\_ΟΤΟΥ. Σε αυτή οι εντολές του βρόχου εκτελούνται μέχρις ότου ικανο-ποιηθεί κάποια συνθήκη η οποία ελέγχεται στο τέλος της επανάληψης.**

**Σύνταξη**

ΑΡΧΗ\_ΕΠΑΝΑΛΗΨΗΣ

**εντολή-1**

**εντολή-2**

**…**

**εντολή-ν**

ΜΕΧΡΙΣ\_ΟΤΟΥ **λογική-έκφραση**

**Παράδειγμα**

ΑΡΧΗ\_ΕΠΑΝΑΛΗΨΗΣ

ΔΙΑΒΑΣΕ **Α**

**Άθροισμα🡨 Άθροισμα +  Α**

ΜΕΧΡΙΣ\_ΟΤΟΥ **Άθροισμα >=1000**

**Λειτουργία**

**Εκτελούνται οι εντολές μεταξύ των** ΑΡΧΗ\_ΕΠΑΝΑΛΗΨΗΣ **και** ΜΕΧΡΙΣ\_ΟΤΟΥ**. Στη συνέχεια ελέγχεται η λογική έκφραση και αν δεν ισχύει (είναι Ψευδής), τότε οι εντολές που βρίσκονται ανάμεσα στις** ΑΡΧΗ\_ΕΠΑΝΑΛΗΨΗΣ **και** ΜΕΧΡΙΣ\_ΟΤΟΥ**, εκτελούνται πάλι. Ελέγχεται ξανά η λογική έκφραση και αν δεν ισχύει, επαναλαμβάνεται η εκτέλεση των ίδιων εντολών.**

**Όταν η λογική έκφραση γίνει Αληθής τότε σταματάει η επανάληψη και εκτελείται η εντολή μετά από την** ΜΕΧΡΙΣ\_ΟΤΟΥ.

**103 / 147**

**Πολύ συχνά η ίδια επαναληπτική διαδικασία μπορεί να γραφεί εξίσου σωστά χρησιμοποιώντας είτε τη δομή ΟΣΟ...ΕΠΑΝΑΛΑΒΕ είτε τη δομή ΜΕΧΡΙΣ\_ΟΤΟΥ και είναι προσωπική επιλογή του προγραμματιστή ποια από τις δυο θα χρησιμοποιήσει.**

**Υπάρχουν όμως περιπτώσεις όπου η χρήση της εντολής ΜΕΧΡΙΣ\_ΟΤΟΥ οδηγεί σε απλούστερα και πιο ευκολονόητα προγράμματα. Γενικά σε περιπτώσεις όπου η επανάληψη θα συμβεί υποχρεωτικά μία φορά,   
είναι προτιμότερη η χρήση της ΜΕΧΡΙΣ\_ΟΤΟΥ.**

**Χαρακτηριστική περίπτωση όπου προτιμάται η εντολή ΜΕΧΡΙΣ\_ΟΤΟΥ είναι στον έλεγχο αποδεκτών τιμών καθώς και στην επιλογή από προκαθορισμένες απαντήσεις ή μενού**

**ΠΑΡΑΔΕΙΓΜΑ 3**

**Στο προηγούμενο παράδειγμα ας υποθέσουμε ότι οι μετρήσεις είναι υποχρεωτικό θετικοί αριθμοί και ότι μετά την εισαγωγή κάθε αριθμού υπάρχει η ερώτηση, αν θα εισάγουμε άλλο. Η διαδικασία θα τελειώσει, όταν η απάντηση θα είναι Όχι (ο ή Ο).**

ΠΡΟΓΡΑΜΜΑ **Άθροισμα2**  
ΜΕΤΑΒΛΗΤΕΣ

ΑΚΕΡΑΙΕΣ: **X,** **Άθροισμα, Πλήθος**  
ΠΡΑΓΜΑΤΙΚΕΣ: **ΜΟ**

ΧΑΡΑΚΤΗΡΕΣ: **Επιλογή**

ΑΡΧΗ

**Πλήθος** 🡨 **0**

**104 / 148**

**Άθροισμα**🡨 **0**  
ΑΡΧΗ\_ΕΠΑΝΑΛΗΨΗΣ   
 ΑΡΧΗ\_ΕΠΑΝΑΛΗΨΗΣ

**! Έλεγχος δεδομένων**

ΓΡΑΨΕ ‘Δώσε Αριθμό’

ΔΙΑΒΑΣΕ X

ΑN X <= 0 TOTE

**105 / 148**

ΓΡΑΨΕ ‘Λάθος Αριθμός, Παρακαλώ δώστε

ξανά…’

ΤΕΛΟΣ\_ΑN

**! Αν το X δεν είναι θετικό, εισάγουμε   
 ! νέο αριθμό**

ΜΕΧΡΙΣ\_ΟΤΟΥ X > 0

Άθροισμα ­­­🡨 **Άθροισμα + X**

**Πλήθος** 🡨 **Πλήθος** + 1

ΓΡ ΑΨΕ ‘Νέα μέτρηση;’

ΔΙΑΒΑΣΕ **Επιλογή**

**! Αν η επιλογή είναι Ο ή ο,   
! τότε σταματάει η επανάληψη**

ΜΕΧΡΙΣ\_ΟΤΟΥ **Επιλογή=’Ο’ Ή Επιλογή= ‘ο’**

**ΜΟ** 🡨 **Άθροισμα / Πλήθος**

 ΓΡΑΨΕ ‘Άθροισμα =’, **Άθροισμα**

ΓΡΑΨΕ ‘Μέσος Όρος =’, **ΜΟ**

ΤΕΛΟΣ\_ΠΡΟΓΡΑΜΜΑΤΟΣ

|  |  |
| --- | --- |
| **πιν.jpg** | **Η εντολή επανάληψης ΜΕΧΡΙΣ\_ΟΤΟΥ εκτελείται υποχρεωτικά τουλάχιστον μία φορά.** |

Προγραμματιστικό περιβάλλον Pascal

PROGRAM athroisma2;

VAR   
 x, sum, count: INTEGER;  
 avg: REAL;

choice: CHAR;

**BEGIN**

**count:=0; sum:=0;**

**REPEAT**

**REPEAT**

**write(‘Δώσε αριθμό’);**

**readln(**x**);**

**IF** x **<= 0 THEN**

**writeln(‘Λάθος αριθμός, Δώσε ξανά..’);**

**UNTIL** x **> 0;**

**sum:= sum+** x**;**

**count:=count + 1;**

**write(‘Νέα μέτρηση ; ’);**

**readln** **(choice);**  
 **UNTIL** (choice=**‘**O’ OR choice=**‘o’**)**;**

**avg:=sum/count;**

**writeln(‘Άθροισμα: ’, sum:5);  
writeln(‘Μέσος όρος: ’, avg:6:2);  
END.**

**106 / 148-149**

**8.2.3 Εντολή ΓΙΑ.. ΑΠΟ...ΜΕΧΡΙ**

|  |  |
| --- | --- |
|  | **Η εντολή ΓΙΑ...ΑΠΟ...ΜΕΧΡΙ χρησιμοποιείται στην περίπτωση που πρέπει να επαναληφθεί η εκτέλεση κάποιων εντολών για προκαθορισμένο αριθμό επαναλήψεων.** |

**Πολύ συχνά ο αριθμός των επαναλήψεων που πρέπει να εκτελεστούν, είναι γνωστός από την αρχή. Αν και αυτού του είδους οι επαναλήψεις μπορούν να αντιμετωπιστούν με τη χρήση των προηγούμενων εντολών επανάληψης, η ΓΛΩΣΣΑ διαθέτει και την εντολή ΓΙΑ. Η εντολή αυτή χειρίζεται μια μεταβλητή, στην οποία αρχικά εκχωρείται η αρχική τιμή. Η τιμή της μεταβλητής συγκρίνεται με την τελική τιμή και εφόσον είναι μικρότερη από αυτή, τότε εκτελούνται οι εντολές που βρίσκονται στο βρόχο (ανάμεσα στις εντολές ΓΙΑ και ΤΕΛΟΣ \_ ΕΠΑΝΑΛΗΨΗΣ). Στη συνέχεια η μεταβλητή ελέγχου αυξάνεται κατά την τιμή που ορίζει το ΒΗΜΑ. Αν η νέα τιμή είναι μικρότερη της τελικής, τότε ο βρόχος εκτελείται ξανά. Η διαδικασία αυτή επαναλαμβάνεται συνεχώς, έως ότου η τιμή ελέγχου γίνει μεγαλύτερη της τελικής τιμής, οπότε τερματίζεται η επανάληψη και το πρόγραμμα συνεχίζει με την εντολή που ακολουθεί το ΤΕΛΟΣ\_ΕΠΑΝΑΛΗΨΗΣ**.

**Ας σημειωθεί ότι, αν η τιμή του βήματος είναι 1, τότε μπορεί να παραληφθεί.**

**107 / 149**

**Σύνταξη**

ΓΙΑ **μεταβλητή** ΑΠΟ **τιμή1** ΜΕΧΡΙ **τιμή2** ΜΕ\_ΒΗΜΑ **τιμή3  
 εντολή-1**

**εντολή-2**

**…**

**εντολή-ν**

ΤΕΛΟΣ\_ΕΠΑΝΑΛΗΨΗΣ

**Παράδειγμα**

ΓΙΑ **Αριθμό** ΑΠΟ **1** ΜΕΧΡΙ **100** ΜΕ\_ΒΗΜΑ **2  
 Άθροισμα 🡨 Άθροισμα + Αριθμό**

ΤΕΛΟΣ\_ΕΠΑΝΑΛΗΨΗΣ€

**Λειτουργία**

**Οι εντολές του βρόχου εκτελούνται για όλες τις τιμές της μεταβλητής από την αρχική τιμή μέχρι την τελική τιμή, αυξανόμενες με την τιμή του βήματος. Αν το βήμα είναι ίσο με 1, τότε παραλείπεται.**

**ΠΑΡΑΔΕΙΓΜΑ 4**

**Το παρακάτω πρόγραμμα υπολογίζει το άθροισμα των περιττών αριθμών που είναι μικρότεροι από το 100.**

**108 / 150**

**ΠΡΟΓΡΑΜΜΑ Περιττοί  
ΜΕΤΑΒΛΗΤΕΣ  
 ΑΚΕΡΑΙΕΣ: Άθροισμα, Αριθμός  
ΑΡΧΗ  
 Άθροισμα 🡨 0  
 ΓΙΑ Αριθμός ΑΠΟ 1 ΜΕΧΡΙ 100 ΜΕ ΒΗΜΑ 2  
 Άθροισμα 🡨 Άθροισμα + Αριθμός  
 ΤΕΛΟΣ\_ΕΠΑΝΑΛΗΨΗΣ  
 ΓΡΑΨΕ Άθροισμα περιττών αριθμών είναι: ', Άθροισμα  
ΤΕΛΟΣ\_ΠΡΟΓΡΑΜΜΑΤΟΣ**

**Πολύ συχνά για την επίλυση των προβλημάτων απαιτείται η χρήση εμφωλευμένων βρόχων. Σε αυτή την περίπτωση ο ένας βρόχος βρίσκεται μέσα στον άλλο.**

|  |  |
| --- | --- |
| **πιν.jpg** | **Κάθε επανάληψη που εκτελείται με μία εντολή ΓΙΑ..ΑΠΟ..ΜΕΧΡΙ, μπορεί να υλοποιηθεί και με τη χρήση των βασικών εντολών επανάληψης ΟΣΟ..ΕΠΑΝΑΛΑΒΕ και ΜΕΧΡΙΣ..ΟΤΟΥ.** |

**Στη χρήση των εμφωλευμένων βρόχων ισχύουν συγκεκριμένοι κανόνες που πρέπει να ακολουθούνται αυστηρά για την σωστή λειτουργία των προγραμμάτων.**

**Συγκεκριμένα:**

* **Ο εσωτερικός βρόχος πρέπει να βρίσκεται ολόκληρος μέσα στον εξωτερικό. Ο βρόχος που ξεκινάει τελευταίος, πρέπει να ολοκληρώνεται πρώτος.**
* **Η είσοδος σε κάθε βρόχο υποχρεωτικά γίνεται από την αρχή του.**
* **Δεν μπορεί να χρησιμοποιηθεί η ίδια μεταβλητή ως μετρητής δύο ή περισσοτέρων βρόχων που ο ένας βρίσκεται στο εσωτερικό του άλλου.**

**109 / 150**

**ΠΑΡΑΔΕΙΓΜΑ 5**

**Να γραφεί πρόγραμμα το οποίο να εκτυπώνει την προπαίδεια του πολλαπλασιασμού.**

**110 / 151**

**ΠΡΟΓΡΑΜΜΑ Προπαίδεια  
! Πρόγραμμα εκτύπωσης της   
! προπαίδειας των αριθμών 1 έως 10**

**ΜΕΤΑΒΛΗΤΕΣ  
 ΑΚΕΡΑΙΕΣ: Α, Β, Γ  
! Α: Πολλαπλασιαστέος  
! Β: Πολλαπλασιαστής  
! Γ: Γινόμενο**

**ΑΡΧΗ  
 ΓΙΑ Α ΑΠΟ 1 ΜΕΧΡΙ 10   
 ΓΙΑ Β ΑΠΟ 1 ΜΕΧΡΙ 10   
 Γ 🡨 Α \* Β  
 ΓΡΑΨΕ Α, ‘X’, Β, ‘=’, Γ  
 ΤΕΛΟΣ\_ΕΠΑΝΑΛΗΨΗΣ  
 ΓΡΑΨΕ ! Εισαγωγή κενής γραμμής στην εκτύπωση**

**ΤΕΛΟΣ\_ΕΠΑΝΑΛΗΨΗΣ  
ΤΕΛΟΣ\_ΠΡΟΓΡΑΜΜΑΤΟΣ**

Προγραμματιστικό περιβάλλον Basic

‘ Προπαίδεια

DEFINT A-Z

FOR a = 1 TO 10

FOR b = 1 TO 10

c = a \* b

PRINT a; “x”; b; “=”; c

NEXT b

PRINT

NEXT a

END

**• Ανακεφαλαίωση**

**Στο κεφάλαιο αυτό παρουσιάστηκαν οι εντολές που χρησιμοποιεί η ΓΛΩΣΣΑ για να υλοποιήσει τις βασικές δομές της επιλογής και της επανάληψης.**

**Αρχικά παρουσιάζονται οι λογικές εκφράσεις καθώς και ο τρόπος που διατυπώνονται σύνθετες λογικές εκφράσεις με τη χρήση των λογικών τελεστών ΟΧΙ, Ή, ΚΑΙ. Η εντολή ΑΝ...ΤΟΤΕ υλοποιεί τη δομή της επιλογής.**

**Η εντολή αυτή εμφανίζεται με πιο σύνθετες μορφές: την εντολή ΑΝ...ΤΟΤΕ...ΑΛΛΙΩΣ καθώς και την εντολή ΑΝ...ΤΟΤΕ... ΑΛΛΙΩΣ\_ΑΝ. Μία άλλη εντολή επιλογής που υπάρχει είναι η εντολή ΕΠΙΛΕΞΕ.**

**Οι εντολές επανάληψης είναι τρεις. Η εντολή ΟΣΟ...ΕΠΑΝΑΛΑΒΕ, η εντολή ΜΕΧΡΙΣΟΤΟΥ και τέλος η εντολή ΓΙΑ. Η εντολή ΓΙΑ χρησιμοποιείται για καθορισμένο αριθμό επαναλήψεων, ενώ ο αριθμός επαναλήψεων των άλλων δύο δεν είναι γνωστός εκ των προτέρων και εξαρτάται από τις συνθήκες που τις ελέγχουν. Η εντολή ΟΣΟ...ΕΠΑΝΑΛΑΒΕ ελέγχει τη συνθήκη στην αρχή της επανάληψης, ενώ η εντολή ΜΕΧΡΙΣ\_ΟΤΟΥ κάνει τον έλεγχο στο τέλος της επανάληψης.**

**Λέξεις κλειδιά**

**Λογική έκφραση, Επιλογή, Επανάληψη, Βρόχος**

**111 / 152**

**• Ερωτήσεις Θέματα για συζήτηση**

1. **Ποιες είναι οι τιμές που μπορεί να πάρει μία λογική έκφραση;**
2. **Ποιοι είναι οι βασικοί λογικοί τελεστές;   
   Αναφέρατε δύο παραδείγματα για τη χρήση του καθενός;**
3. **Ποια είναι η σύνταξη της εντολής ΑΝ;**
4. **Ποια είναι η διαφορά της εντολής ΑΝ-ΑΛΛΙΩΣ και της ΑΝ - ΑΛΛΙΩΣ\_ΑΝ;**
5. **Τι είναι τα εμφωλευμένα ΑΝ;**
6. **Πότε χρησιμοποιείται η εντολή ΕΠΙΛΕΞΕ;**
7. **Ποιες οι εντολές επανάληψης;**
8. **Ποιες οι διαφορές της εντολής ΟΣΟ και της εντολής   
   ΜΕΧΡΙΣ\_ ΟΤΟΥ;**
9. **Πώς συντάσσεται η εντολή ΓΙΑ;**
10. **Ποια η βασική διαφορά της εντολής ΓΙΑ από τις άλλες δυο εντολές επανάληψης;**

**• Βιβλιογραφία**

1. **Κάβουρας, Δομημένος προγραμματισμός με Pascal, Κλειδάριθμος, Αθήνα, 1997.**
2. **Κ. Γιαλούρης-Κ. Σταθόπουλος, Προγραμματισμός σε Turbo Pascal, Αθήνα, 1996.**
3. **Χρ. Κοιλίας, Η QuickBasic και οι εφαρμογές της, Εκδόσεις Νέων Τεχνολογιών, Αθήνα, 1992.**
4. **R. Shackelford, Introduction to Computing and Algorithms, Addison-Wesley, USA, 1998.**

**112 / 152**

1. **S. Leestma-L.Nyhoff, Turbo Pascal, Programming and Solving, McMillan, New York, 1990.**
2. **N. Wirth, Systematic Programming: An introduction, Prentice Hall, 1973.**

**113 / 152**

**Κεφάλαιο 9**

**Πίνακες**

**ΠΕΡΙΕΧΟΜΕΝΑ**

|  |  |
| --- | --- |
| **115 / 153** | **Μονοδιάστατοι πίνακες** |
|  | **Πότε πρέπει να χρησιμοποιούνται πίνακες** |
|  | **Πολυδιάστατοι πίνακες** |
|  | **Τυπικές επεξεργασίες πινάκων** |

**• Εισαγωγή**

**Η χρήση των μεταβλητών με δείκτες στην άλγεβρα είναι ένας ιδιαίτερα δυναμικός τρόπος για τη διαχείριση μεγάλου αριθμού δεδομένων ιδίου τύπου. Οι γλώσσες προγραμματισμού, δανείζονται την έννοια των μεταβλητών με δείκτες και χρησιμοποιούν τους πίνακες για τον ίδιο λόγο.**

**Στο κεφάλαιο αυτό παρουσιάζονται οι έννοιες του πίνακα καθώς και οι βασικοί τρόποι επεξεργασίας τους από τη ΓΛΩΣΣΑ.**

**116 / 154**

**Παρουσιάζονται οι μονοδιάστατοι καθώς και οι πολύ-διάστατοι πίνακες, ο τρόπος με τον οποίο ορίζονται και χρησιμοποιούνται και τέλος συζητούνται οι πλέον κοινές διαδικασίες πάνω σε πίνακες, η εύρεση μεγίστου και ελαχίστου, η αναζήτηση, η ταξινόμηση και η συγχώνευση πινάκων.**

**• Διδακτικοί στόχοι**

**Να είναι σε θέση ο μαθητής**

* **Να επιλέγει το είδος του πίνακα.**
* **Να ορίζει τους πίνακες σε ένα πρόγραμμα.**
* **Να εισάγει, να επεξεργάζεται και να τυπώνει τα στοιχεία ενός πίνακα.**
* **Να αποφασίζει αν είναι απαραίτητη η χρήση πίνακα.**
* **Να αναφέρει τις βασικές επεξεργασίες σε ένα πίνακα.**
* **Να αναζητά και να ταξινομεί τα στοιχεία ενός πίνακα.**

**• Προερωτήσεις**

* **Πώς μπορούν να αποθηκευτούν πολλά παρόμοια δεδομένα στον υπολογιστή, για παράδειγμα τα ονόματα όλων των μαθητών μίας τάξης;**
* **Για ποιο λόγο χρησιμοποιούνται στην άλγεβρα οι μεταβλητές με δείκτες;**
* **Σε ποιες περιπτώσεις χρειάζεται η ταξινόμηση κάποιων δεδομένων;**
* **Είναι εύκολη η αναζήτηση ενός συγκεκριμένου δεδομένου σε μη ταξινομημένα δεδομένα;**

**117 / 154**

|  |  |
| --- | --- |
| **9.1** | **Μονοδιάστατοι πίνακες** |

**Πολλά από τα προβλήματα τα οποία παρουσιάστηκαν στα προηγούμενα κεφάλαια, απλά επεξεργάζονται μία σειρά δεδομένων.**

**Διαβάζουν ένα δεδομένο κάθε φορά, το εκχωρούν σε μία μεταβλητή, εκτελούν τους αντίστοιχους υπολο-γισμούς και στη συνέχεια επαναλαμβάνεται η ίδια διαδικασία μέχρι να τελειώσουν όλα τα δεδομένα.**

**118 / 155**

**Για παράδειγμα ένα πρόγραμμα το οποίο διαβάζει τις θερμοκρασίες διαφόρων ημερών του μήνα, έστω 30, και υπολογίζει τη μέση θερμοκρασία, μπορεί πολύ απλά να γραφεί ως εξής**

**…..**

**Σύνολο 🡨 0**

**ΓΙΑ Ημέρα ΑΠΟ 1 ΜΕΧΡΙ 30**

**ΔΙΑΒΑΣΕ Θερμοκρασία**

**Σύνολο 🡨Σύνολο+Θερμοκρασία**

**ΤΕΛΟΣ\_ΕΠΑΝΑΛΗΨΗΣ**

**Μέση 🡨 Σύνολο / 30**

**…..**

**Χρησιμοποιώντας λοιπόν μόνο μία μεταβλητή, τη μεταβλητή Θερμοκρασία*,* το πρόβλημα λύνεται πολύ απλά και το αντίστοιχο πρόγραμμα είναι σύντομο και κατανοητό.**

**Αν όμως στο προηγούμενο πρόγραμμα ζητείται και ο αριθμός των ημερών που η θερμοκρασία ήταν κατώτερη της μέσης, τότε η σύγκριση αυτή πρέπει να γίνει μετά τον υπολογισμό της μέσης θερμοκρασίας. Αυτό σημαίνει ότι όλες οι θερμοκρασίες πρέπει να επαναεισαχθούν για να συγκριθούν με τη μέση.**

**Μία άλλη λύση είναι να καταχωρηθεί κάθε θερμοκρασία σε διαφορετική μεταβλητή, έτσι ώστε κάθε τιμή που εισάγεται να διατηρείται στη μνήμη και να μπορεί να συγκριθεί με τη μέση, αφού αυτή υπολογιστεί. Τότε όμως πρέπει να δημιουργηθούν 30 διαφορετικές μεταβλητές Θερμοκρασία1, Θερμοκρασία2,..., Θερμοκρασία30. Για να γραφεί το πρόγραμμα χρειάζονται τριάντα εντολές ΔΙΑΒΑΣΕ και τριάντα εντολές ΑΝ.**

**119 / 155**

**Αν και αυτή η λύση είναι σωστή και πρακτική για μικρό αριθμό δεδομένων, προφανώς δεν εξυπηρετεί την επεξεργασία μεγάλου αριθμού δεδομένων.**

**Η καλύτερη λύση στο πρόβλημα αυτό είναι η χρήση μεταβλητής με δείκτες, έννοια που είναι γνωστή από τα μαθηματικά και υλοποιείται στον προγραμματισμό με τη δομή δεδομένων του πίνακα. Χρησιμοποιείται λοι-πόν μόνο ένα όνομα Θερμοκρασία, που αναφέρεται και στις τριάντα διαφορετικές θερμοκρασίες.**

**Το όνομα του πίνακα καθορίζει μία ομάδα διαδοχικών θέσεων στη μνήμη. Κάθε συγκεκριμένη θέση μνήμης καλείται στοιχείο του πίνακα και προσδιορίζεται από την τιμή ενός δείκτη, όπως φαίνεται και στο σχήμα 9.1.**

|  |  |  |
| --- | --- | --- |
|  | **ΜΝΗΜΗ** |  |
|  | **-** |  |
|  | **-** |  |
|  | **-** |  |
|  | **26** | **Θερμοκρασία[1]** | |
|  | **27** | **Θερμοκρασία[2]** | |
| **Θερμοκρασία** | **-** |  |
| **-** |  |
|  | **-** |  |
|  | **21** | **Θερμοκρασία[29]** |
|  | **25** | **Θερμοκρασία[30]** |
|  | **-** |  |
|  | **-** |  |
|  | **-** |  |

**Σχ. 9.1. Ο πίνακας Θερμοκρασία**

**120 / 156**

|  |  |
| --- | --- |
| **πιν.jpg** | **Ο δείκτης είναι μια μεταβλητή που μπορεί να έχει οποιοδήποτε δεκτό όνομα. Είναι σύνηθες όμως στον προγραμματισμό ως δείκτες να χρησιμοποιούνται οι μεταβλητές i, j, k.** |

**Οι πίνακες που χρησιμοποιούν ένα μόνο δείκτη για την αναφορά των στοιχείων τους, ονομάζονται μονο-διάστατοι πίνακες. Το όνομα του πίνακα μπορεί να είναι οποιοδήποτε δεκτό όνομα της ΓΛΩΣΣΑΣ και ο δείκτης είναι μία ακέραια έκφραση, σταθερή ή μεταβλητή που περικλείεται μέσα στα σύμβολα [ και ] . Το στοιχείο Θερμοκρασία[2], εκφράζει τη θερμοκρασία της δεύτερης ημέρας, αναφέρεται στο δεύτερο στοιχείο του πίνακα Θερμοκρασία και έχει την τιμή 27.**

**Γενικότερα το στοιχείο Θερμοκρασία**[**i**] **αναφέρεται στο  
 i -στό στοιχείο του πίνακα.**

**Κάθε πίνακας πρέπει υποχρεωτικά να περιέχει δεδο-μένα του ιδίου τύπου, δηλαδή ακέραια, πραγματικά, λογικά, ή αλφαριθμητικά. Ο τύπος του πίνακα δηλώ-νεται μαζί με τις άλλες μεταβλητές του προγράμματος στο τμήμα δήλωσης μεταβλητών. Εκτός από τον τύπο του πίνακα πρέπει να δηλώνεται και ο αριθμός των στοιχείων που περιέχει ή καλύτερα ο μεγαλύτερος αριθμός στοιχείων που μπορεί να έχει ο συγκεκριμένος πίνακας και αυτό για να δεσμευτούν οι αντίστοιχες συνεχόμενες θέσεις μνήμης.**

**121 / 156**

**Για παράδειγμα**

**ΜΕΤΑΒΛΗΤΕΣ**

**ΠΡΑΓΜΑΤΙΚΕΣ: Θερμοκρασία [30]**

**ΟΡΙΣΜΟΣ**

**Πίνακας είναι ένα σύνολο αντικειμένων ίδιου τύπου, τα οποία αναφέρονται με ένα κοινό όνομα. Κάθε ένα από τα αντικείμενα που απαρτίζουν τον πίνακα λέγεται στοιχείο του πίνακα. Η αναφορά σε ατομικά στοιχεία του πίνακα γίνεται με το όνομα του πίνακα ακολου-θούμενο από ένα δείκτη.**

**ΠΑΡΑΔΕΙΓΜΑ 1**

**Χρησιμοποιώντας μεταβλητές με δείκτες για το προηγούμενο παράδειγμα έχουμε το εξής πρόγραμμα**

ΠΡΟΓΡΑΜΜΑ **Θερμοκρασίες**  
ΜΕΤΑΒΛΗΤΕΣ

ΠΡΑΓΜΑΤΙΚΕΣ: **Θερμοκρασία**[ **30** ], **Μέση**, **Σύνολο**  
ΑΚΕΡΑΙΕΣ: **i**, **Ημέρες**

ΑΡΧΗ  
**Σύνολο** 🡨 **0**  
ΓΙΑ**i** ΑΠΟ **1** ΜΕΧΡΙ **30**  
    ΓΡΑΨΕ **'Δώσε τη θερμοκρασία'**  
    ΔΙΑΒΑΣΕ **Θερμοκρασία**[ **i** ]   
    **Σύνολο** 🡨 **Σύνολο** + **Θερμοκρασία**[ **i** ]   
ΤΕΛΟΣ\_ΕΠΑΝΑΛΗΨΗΣ  
**Μέση** 🡨 **Σύνολο** / **30**  
**Ημέρες** 🡨 **0**

**122 / 156-157**

ΓΙΑ**i** ΑΠΟ **1** ΜΕΧΡΙ **30**  
    ΑΝ **Θερμοκρασία**[ **i** ] < **Μέση** ΤΟΤΕ  
      **Ημέρες** 🡨 **Ημέρες** + **1**  
 ΤΕΛΟΣ\_ΑΝ  
ΤΕΛΟΣ\_ΕΠΑΝΑΛΗΨΗΣ

ΓΡΑΨΕ **'Μέση θερμοκρασία:'**, **Μέση**  
ΓΡΑΨΕ **'Ημέρες με μικρότερη θερμοκρασία'**,**Ημέρες**

ΤΕΛΟΣ\_ΠΡΟΓΡΑΜΜΑΤΟΣ

**Με τη χρήση του πίνακα, όλα τα δεδομένα καταχωρού-νται κάτω από το ίδιο όνομα μεταβλητής, στο παρά-δειγμα Θερμοκρασία. Η ανάγνωση όλων των δεδο-μένων απλοποιείται, αφού είναι μόνο μία εντολή,** ΔΙΑΒΑΣΕ **Θερμοκρασία**[ **i** ] **η οποία βρίσκεται μέσα σε ένα βρόχο και επαναλαμβάνεται όσες φορές απαιτείται όπως και ο υπολογισμός του αθροίσματος,**

**Σύνολο** 🡨**Σύνολο +Θερμοκρασία[ i ].**

**Οι τιμές μετά τον υπολογισμό της μέσης τιμής δεν χάνονται, αφού βρίσκονται στα στοιχεία του πίνακα και ξαναχρησιμοποιούνται για την εύρεση του αριθμού των ημερών, που η θερμοκρασία είναι μικρότερη από τη μέση. Ο υπολογισμός των ημερών είναι μόνο μία εντολή ΑΝ, η οποία βρίσκεται σε ένα βρόχο και επανα-λαμβάνεται 30 φορές.**

**123 / 157**

|  |  |
| --- | --- |
|  | **Η ανάγνωση, η επεξεργασία και η εκτύπωση των στοιχείων των πινάκων γίνεται πάντοτε από βρόχους, οι οποίοι επαναλαμβάνονται προκαθορισμένο αριθμό φορών, όσο είναι τα στοιχεία του πίνακα και υλοποιούνται καλύτερα στον προγραμματισμό με την εντολή επανάληψης ΓΙΑ.** |

**ΠΑΡΑΔΕΙΓΜΑ 2**

**Να γραφεί πρόγραμμα το οποίο να υπολογίζει τα βασικά στατιστικά μεγέθη, τη μέση τιμή, την τυπική απόκλιση και τη διάμεσο τιμή Ν ακεραίων αριθμών, όπου το Ν είναι από 2 μέχρι 100.**

**Τα δεδομένα εισάγονται από το πληκτρολόγιο και καταχωρούνται στον πίνακα Χ.**

**Η μέση τιμή δίνεται από τον τύπο**

**   
Η τυπική απόκλιση δίνεται από το ν τύπο**

****

**Για να βρεθεί η διάμεσος τιμή πρέπει υποχρεωτικά οι αριθμοί να ταξινομηθούν κατά αύξουσα σειρά. Τότε διάμεσος τιμή, είναι η τιμή για την οποία οι μισοί αριθμοί είναι μικρότεροι και οι άλλοι μισοί μεγαλύτεροι. Στην περίπτωση που το πλήθος των αριθμών είναι περιττό, τότε διάμεσος είναι ο μεσαίος, ενώ στην περίπτωση που είναι άρτιο, τότε διάμεσος είναι το ημιάθροισμα των δύο μεσαίων αριθμών.**

**124 / 157-158**

**Η ταξινόμηση των στοιχείων γίνεται με τη μέθοδο ταξινόμησης ευθείας ανταλλαγής, η οποία παρου-σιάστηκε στο κεφάλαιο 3.**

ΠΡΟΓΡΑΜΜΑ **Στατιστική**  
ΜΕΤΑΒΛΗΤΕΣ  
 ΑΚΕΡΑΙΕΣ: **i**, **Ν, Χ[ 100 ], Άθροισμα,  
 Άθροισμα\_2, Βοηθητική**  
 ΠΡΑΓΜΑΤΙΚΕΣ: **ΜΤ, Τυπ\_Απόκλιση, Διάμεσος**  
ΑΡΧΗ  
! Εισαγωγή δεδομένων

ΓΡΑΨΕ **'Δώσε το πλήθος των αριθμών (μέγιστο 100)'**

ΔΙΑΒΑΣΕ **Ν**

ΓΙΑ**i** ΑΠΟ **1** ΜΕΧΡΙ **Ν**  
    ΓΡΑΨΕ **'Δώσε τον ', i, '-το αριθμό'**  
    ΔΙΑΒΑΣΕ **X[ i ]**  
ΤΕΛΟΣ\_ΕΠΑΝΑΛΗΨΗΣ

! Υπολογισμός αθροισμάτων

**Άθροισμα** 🡨 **0**  
**Άθροισμα\_2** 🡨 **0**  
ΓΙΑ**i** ΑΠΟ **1** ΜΕΧΡΙ **Ν**

**Άθροισμα** 🡨 **Άθροισμα + X[ i ]**

**Άθροισμα**\_2 🡨 **Άθροισμα\_2 + X[ i ]^2**

ΤΕΛΟΣ\_ΕΠΑΝΑΛΗΨΗΣ

! Υπολογισμός μέσου όρου

**ΜΤ** 🡨 **Άθροισμα** / **Ν**

! Υπολογισμός τυπικής απόκλισης

**Τυπ\_Απόκλιση** 🡨 Τ\_Ρ( **Άθροισμα\_2 / Ν – ΜΤ^2 )**

! Ταξινόμηση των στοιχείων του πίνακα

**125 / 158**

ΓΙΑ**i** ΑΠΟ **2** ΜΕΧΡΙ **Ν**

ΓΙΑ**j** ΑΠΟ **Ν** ΜΕΧΡΙ **i** **ΜΕ\_ΒΗΜΑ** **-1**  
    ΑΝ **X[ j -1 ] >** **X[ j ]**  ΤΟΤΕ

! Αντιμετάθεση των στοιχείων j και j-1

**Βοηθητική** 🡨 **X[ j -1 ]**

**X[ j -1 ]** 🡨 **X[ j]  
 X[ j ]** 🡨 **Βοηθητική**

ΤΕΛΟΣ\_ΑΝ  
ΤΕΛΟΣ\_ΕΠΑΝΑΛΗΨΗΣ ! j

ΤΕΛΟΣ\_ΕΠΑΝΑΛΗΨΗΣ ! i

! Υπολογισμός διαμέσου

ΑΝ **N MOD 2= 0**  ΤΟΤΕ

**Διάμεσος** 🡨 ( **X[ N/2 ] + X[ N/2 +1] ) /2**

ΑΛΛΙΩΣ   
 **Διάμεσος** 🡨 **X[ (N+1) /2 ]** ΤΕΛΟΣ\_ΑΝ  
! Εκτύπωση αποτελεσμάτων

ΓΡΑΨΕ **'ΑΠΟΤΕΛΕΣΜΑΤΑ'**ΓΡΑΨΕ**'=============='**ΓΡΑΨΕ**'Πλήθος τιμών = ', N**ΓΡΑΨΕ**'Μέση τιμή = ', ΜΤ**ΓΡΑΨΕ**'Τυπική απόκλιση = ', Τυπ\_Απόκλιση**ΓΡΑΨΕ**'Διάμεσος = ', Διάμεσος**

ΤΕΛΟΣ\_ΠΡΟΓΡΑΜΜΑΤΟΣ

**126 / 158-159**

Προγραμματιστικό περιβάλλον Basic

`Στατιστική

DEFINT A-Z

DIM x(100)

sum = 0: sum2 = 0

' Εισαγωγή δεδομένων

INPUT "N=", n

FOR i = 1 TO n

PRINT "Δώσε τον "; i;"-το αριθμό:";

INPUT "", x(i)

NEXT i

' Υπολογισμός αθροισμάτων

FOR i = 1 TO n

sum = sum + x(i)

sum2 = sum2 + x(i)^2

NEXT i

mt! = sum / n 'Μέση τιμή

ta! = SQR(sum2 /n - mt!^ 2) 'Τυπ.Απόκλ.

'Ταξινόμηση πίνακα

**127 / 159**

FOR i = 2 TO n

FOR j = n TO i STEP -1

IF x(j-1) > x(j)

THEN SWAP x(j-1),x(j)

NEXT j

NEXT i

'Υπολογισμός διαμέσου

IF n MOD 2 = 0 THEN

median = ( x(n/2)+ x(n/2 +1) ) /2

ELSE median = x ( (n+1) /2 )

ENDIF

PRINT " ΑΠΟΤΕΛΕΣΜΑΤΑ"

PRINT "====================="

PRINT "Πλήθος τιμών ="; n

PRINT "Μέση τιμή ="; m!

PRINT "Τυπική απόκλιση ="; ta!

PRINT "Διάμεσος ="; median

END

|  |  |
| --- | --- |
| **9.2** | **Πότε πρέπει να χρησιμοποιούνται πίνακες** |

**Η χρήση πινάκων είναι ένας βολικός τρόπος για τη διαχείριση πολλών δεδομένων ιδίου τύπου, αλλά συχνά η χρήση τους είναι περιττή και επιζήμια στην ανάπτυξη του προγράμματος.**

**Πέρα από τα πλεονεκτήματα που αναφέρθηκαν, υπάρχουν και δύο μειονεκτήματα από τη χρήση πινάκων.**

**Οι πίνακες απαιτούν μνήμη. Κάθε πίνακας δεσμεύει από την αρχή του προγράμματος πολλές θέσεις μνήμης. Σε ένα μεγάλο και σύνθετο πρόγραμμα η άσκοπη χρήση μεγάλων πινάκων μπορεί να οδηγήσει ακόμη και σε αδυναμία εκτέλεσης του προγράμματος.**

**Οι πίνακες περιορίζουν τις δυνατότητες του προγράμματος. Στο προηγούμενο πρόγραμμα του υπολογισμού των στατιστικών μεγεθών, υπάρχει ανώτατο όριο στο πλήθος των αριθμών ίσο με 100. Αυτό γιατί οι πίνακες είναι στατικές δομές και το μέγεθος τους πρέπει να δηλώνεται στην αρχή του προγράμματος, ενώ παραμένει υποχρεωτικά σταθερό κατά την εκτέλεση του προγράμματος.**

**128 / 159-160**

**Η απόφαση για την χρήση ή όχι πίνακα για την διαχείριση των δεδομένων είναι κυρίως θέμα εμπειρίας στον προγραμματισμό.**

**Γενικά, αν τα δεδομένα που εισάγονται σε ένα πρό-γραμμα πρέπει να διατηρούνται στη μνήμη μέχρι το τέλος της εκτέλεσης, τότε η χρήση πινάκων βοηθάει ή συχνά είναι απαραίτητη για την επίλυση του προβλή-ματος. Σε άλλη περίπτωση μπορεί να αποφεύγεται η χρήση τους.**

**129 / 160**

**Ας επιστρέψουμε στο προηγούμενο παράδειγμα.**

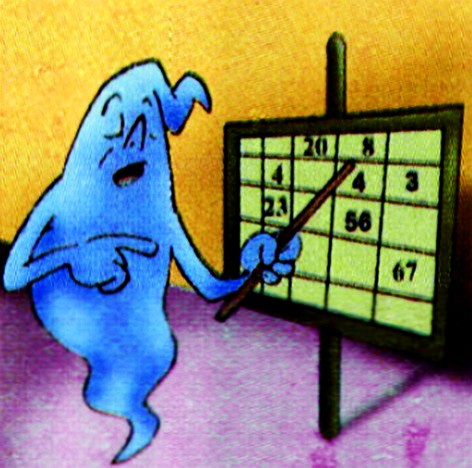
**Για τον υπολογισμό της μέσης τιμής και της τυπικής απόκλισης δεν είναι απαραίτητο να διατηρούνται οι τιμές στη μνήμη. Το πρόγραμμα θα δούλευε το ίδιο καλά και χωρίς τη χρήση πινάκων, αλλά με τη χρήση μίας και μόνο μεταβλητής. Ο υπολογισμός όμως της διάμεσης τιμής, που προϋποθέτει την ταξινόμηση των δεδομένων, απαιτεί τη χρήση πίνακα. Αν λοιπόν το πρόβλημα απαιτούσε μόνο τον υπολογισμό του μέσου όρου και της τυπικής απόκλισης, θα ήταν προτιμότερη μία λύση χωρίς τη χρήση πινάκων.**

|  |  |
| --- | --- |
| **9.3** | **Πολυδιάστατοι πίνακες** |

**Στο προηγούμενο παράδειγμα υπήρχαν 30 τιμές της θερμοκρασίας, μία για κάθε ημέρα του μήνα και για την επίλυσή του χρησιμοποιήθηκε ένας πίνακας 30 θέσεων, ο πίνακας Θερμοκρασία. Ο πίνακας αυτός περιέχει τις τιμές της θερμοκρασίας για κάθε ημέρα μίας πόλης και ο δείκτης δείχνει την ημέρα.**

**Έστω ότι οι θερμοκρασίες δίνονται από τον παρακάτω πίνακα.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ΗΜΕΡΑ** | **ΠΟΛΗ** | | | |
|
| **1** | **2** | **…** | **10** |
| **1** | **25** | **21** | **…** | **32** |
| **2** | **26** | **22** | **…** | **31** |
| **…** | **…** | **…** | **…** | **…** |
| **30** | **27** | **23** | **…** | **30** |

** Ο πίνακας αυτός έχει τις θερμο-κρασίες για 30 ημέρες αλλά για δέκα διαφορετικές πόλεις, δηλαδή υπάρχουν συνολικά 300 τιμές θερμοκρασίας. Για να καθοριστεί κάθε στοιχείο δεν αρκεί μόνο ένας δείκτης, αλλά απαιτούνται δύο δείκτες, ο ένας για την ημέρα και ο δεύτερος για την πόλη. Για παράδειγμα η πρώτη πόλη την τριακοστή ημέρα είχε θερμοκρασία 27.**

**Για την επεξεργασία των θερμοκρασιών μπορεί να χρη-σιμοποιηθεί ένας δισδιάστατος πίνακας, στον οποίο ο πρώτος δείκτης δείχνει τη γραμμή (στο παράδειγμα την ημέρα) και ο δεύτερος τη στήλη (την πόλη). Το στοιχείο Θερμοκρασία [30,1] έχει την τιμή 27.**

**Στην γενική περίπτωση κάθε στοιχείο του πίνακα είναι το στοιχείο Θερμοκρασία [ i, j ] και αναφέρεται στη θερμοκρασία στην i γραμμή και την j στήλη, όπου το i παίρνει τιμή από 1 έως 30 και το j από 1 έως 10. Ο πίνα-κας Θερμοκρασία είναι ένας δισδιάστατος πίνακας 30X10.**

**130 / 160-161**

**ΠΑΡΑΔΕΙΓΜΑ 3**

**Να γραφεί πρόγραμμα που να υπολογίζει τη μέση θερμοκρασία κάθε πόλης για τον προηγούμενο πίνακα θερμοκρασιών (δίδονται 30 θερμοκρασίες 10 πόλεων). Επίσης, για κάθε πόλη, να υπολογίζει πόσες ημέρες η θερμοκρασία ήταν κατώτερη από την αντίστοιχη μέση.**

**131 / 161**

ΠΡΟΓΡΑΜΜΑ **Θερμοκρασίες\_2**  
ΜΕΤΑΒΛΗΤΕΣ  
 ΠΡΑΓΜΑΤΙΚΕΣ: **Θερμοκρασία**[**30,10**],  **Μέση**[**10**]  
 ΑΚΕΡΑΙΕΣ: **i**,**j**, **Ημέρες, Σύνολο**ΑΡΧΗ  
! ' Εισαγωγή δεδομένων

ΓΙΑ**i** ΑΠΟ **1** ΜΕΧΡΙ **30**

ΓΙΑ**j** ΑΠΟ **1** ΜΕΧΡΙ **10**  
 ΓΡΑΨΕ **‘Δώσε τη θερμοκρασία’,i,j**    ΔΙΑΒΑΣΕ **Θερμοκρασία** [ **i, j** ]   
 ΤΕΛΟΣ\_ΕΠΑΝΑΛΗΨΗΣ  
ΤΕΛΟΣ\_ΕΠΑΝΑΛΗΨΗΣ

! ' Υπολογισμοί

ΓΙΑ**j** ΑΠΟ **1** ΜΕΧΡΙ **10**

**Σύνολο** 🡨 **0** ΓΙΑ **i** ΑΠΟ **1** ΜΕΧΡΙ **30**

**Σύνολο** 🡨 **Σύνολο** + **Θερμοκρασία**[**i,j**]

ΤΕΛΟΣ\_ΕΠΑΝΑΛΗΨΗΣ

**Μέση [ j ]** 🡨 **Σύνολο** / **30**

ΤΕΛΟΣ\_ΕΠΑΝΑΛΗΨΗΣ  
ΓΙΑ**j** ΑΠΟ **1** ΜΕΧΡΙ **10**

**Ημέρες** 🡨 **0** ΓΙΑ **i** ΑΠΟ **1** ΜΕΧΡΙ **30**

ΑΝ **Θερμοκρασία**[**i,j**] < **Μέση[j]** ΤΟΤΕ  
      **Ημέρες** 🡨 **Ημέρες** + **1**  
 ΤΕΛΟΣ\_ΑΝ  
 ΤΕΛΟΣ\_ΕΠΑΝΑΛΗΨΗΣ

ΓΡΑΨΕ ‘**Μέση θερμοκρασία ’**,**j, πόλης:’, Μέση[ j ]** ΓΡΑΨΕ **‘Ημέρες με μικρότερη θερμοκρασία’,**  **Ημέρες**

ΤΕΛΟΣ\_ΕΠΑΝΑΛΗΨΗΣ  
ΤΕΛΟΣ\_ΠΡΟΓΡΑΜΜΑΤΟΣ

|  |  |
| --- | --- |
|  | **Η ανάγνωση, η επεξεργασία καθώς και η εκτύπωση των στοιχείων πολυδιάστατων πινάκων γίνεται πάντοτε από βρόχους, οι οποίοι υλοποιούνται στον προγραμματισμό με εμφωλευμένες εντολές επανάληψης ΓΙΑ.** |

Προγραμματιστικό περιβάλλον Pascal

PROGRAM poleis;

VAR

temperature: array[1..30,1..10] OF REAL;

ave: array [1..10] OF REAL;

total: REAL;

i, j, days: INTEGER;

**BEGIN**

**FOR i:= 1 TO 30 DO**

**FOR j:=1 TO 10 DO**

**BEGIN**

**write(‘Δώσε τη θερμοκρασία: ’,i,‘ ’, j);**

**readln (**temperature[ i, j ]);

**132 / 161-162**

**END;**

**FOR j:= 1 TO 10 DO**

**BEGIN**

**total:=0;**

**FOR i:= 1 TO 30 DO**

**total:=total** **+**temperature[i, j];

**ave[ j ]:= total/30;**

**END;**

**FOR j:= 1 TO 10 DO**

**BEGIN**

**days:=0;**

**FOR i:= 1 TO 30 DO**

IF temperature[i, j] **< ave[ j ] THEN**

**days:=days +1;**

**writeln (‘Μέση θερμοκρασία :’,i, ‘ πόλης ’, ave[ j ]:4:1);  
 writeln (‘Ημέρες με μικρότερη θερμοκρασία :’,days);**

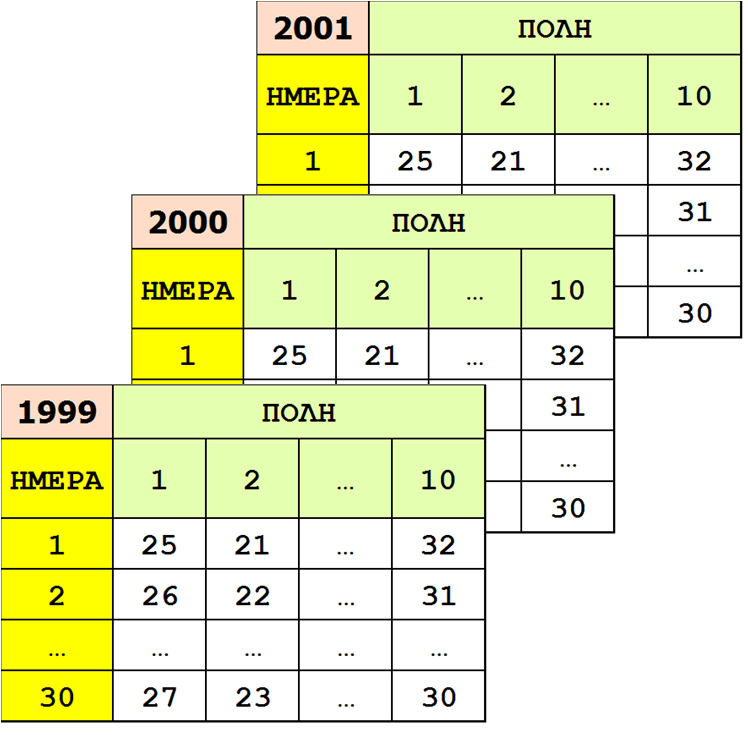
**133 / 162**

**END;**

**END.**

**Εκτός από μονοδιάστατους και δισδιάστατους πίνακες υπάρχουν πίνακες με περισσότερες διαστάσεις: τρισδιάστατοι, τετραδιάστατοι και γενικά πολυδιάστα-τοι, ανάλογα με τον αριθμό των δεικτών που χρησιμο-ποιούνται για τον καθορισμό των στοιχείων. Ωστόσο τα περισσότερα προβλήματα αντιμετωπίζονται με τη χρήση πινάκων μονοδιάστατων ή δισδιάστατων.**

**Στο προηγούμενο παράδειγμα έστω ότι έχουμε θερμοκρασίες για κάθε μία πόλη, για κάθε ημέρα αλλά και για κάθε έτος. Τότε κάθε στοιχείο χρειάζεται τρεις δείκτες. Έναν που θα δείχνει το έτος, έναν που θα δείχνει την πόλη και έναν που θα δείχνει την ημέρα. Για παράδειγμα η θερμοκρασία της πρώτης ημέρας, της πρώτης πόλης και του πρώτου έτους είναι 25 και είναι το στοιχείο Θερμοκρασία [1, 1, 1].**

****

**134 / 162-163**

**ΠΑΡΑΔΕΙΓΜΑ 4**

**135 / 163**

**Να γραφεί πρόγραμμα το οποίο:**

**α. Να διαβάζει τα ονόματα δέκα κινηματογράφων και τις αντίστοιχες εισπράξεις τους για κάθε ημέρα μιας εβδομάδας.**

**β. Να υπολογίζει και να εκτυπώνει το άθροισμα των εισπράξεων κάθε κινηματογράφου, καθώς και τον κινηματογράφο με τη μέγιστη συνολική είσπραξη.**

**γ. Να υπολογίζει και να εκτυπώνει το άθροισμα των εισπράξεων κάθε ημέρας, καθώς και την ημέρα με τη μέγιστη συνολική είσπραξη.**

**Για την επίλυση του προβλήματος πρέπει να χρησι-μοποιηθούν δύο πίνακες. Ο πρώτος θα περιέχει μόνο τα ονόματα των κινηματογράφων, θα είναι δηλαδή ένας μονοδιάστατος πίνακας χαρακτήρων με δέκα γραμμές. Ο δεύτερος θα περιέχει τις εισπράξεις, θα είναι ένας πίνακας δισδιάστατος ακεραίων αριθμών με δέκα γραμμές, μία για κάθε κινηματογράφο και επτά στήλες, μία για κάθε ημέρα.**

**Το πρόγραμμα ουσιαστικά αποτελείται από τρία τμήματα:**

1. **Την ανάγνωση των δεδομένων και την καταχώ-ρησή του στους αντίστοιχους πίνακες, Ονόματα και Εισπράξεις.**
2. **Τον υπολογισμό του συνόλου των εισπράξεων ανά κινηματογράφο και την εύρεση της μέγιστης συνολικής είσπραξης, δηλαδή του αθροίσματος των γραμμών του πίνακα.**
3. **Τον υπολογισμό του συνόλου των εισπράξεων ανά ημέρα και την εύρεση της μέγιστης συνολικής είσπραξης, δηλαδή του αθροίσματος των στηλών του πίνακα.**

**Προσέξτε τις διαφορές του δεύτερου και του τρίτου τμήματος. Και οι δύο χρησιμοποιούν δύο εμφωλευ-μένους βρόχους, ένα για τις γραμμές και ένα για τις στήλες αλλά σε διαφορετική σειρά.**

**ΠΡΟΓΡΑΜΜΑ Κινηματογράφοι**

**ΜΕΤΑΒΛΗΤΕΣ**

**ΑΚΕΡΑΙΕΣ: Εισπράξεις [ 10,7] ,i,j, Άθροισμα, Μέγιστο, Θέση**

**ΧΑΡΑΚΤΗΡΕΣ: Ονόματα[10]**

ΑΡΧΗ

**! Εισαγωγή δεδομένων και εκχώρηση τους σε δύο πίνακες**

**ΓΙΑ** i **ΑΠΟ** 1 **ΜΕΧΡΙ** 10

ΓΡΑΨΕ **'Δώσε το όνομα του ',**i**,' κινηματογράφου'**

ΔΙΑΒΑΣΕ **Ονόματα[**i**]**

ΓΙΑ **j** ΑΠΟ **1** ΜΕΧΡΙ **7**

ΓΡΑΨΕ **'Δώσε την 'j,'-η είσπραξη'**

ΔΙΑΒΑΣΕ **Εισπράξεις[i,j]**

ΤΕΛΟΣ\_ΕΠΑΝΑΛΗΨΗΣ

ΤΕΛΟΣ\_ΕΠΑΝΑΛΗΨΗΣ

**!Άθροισμα των στοιχείων του πίνακα Εισπράξεις ανά γραμμή**

**!και υπολογισμός του μέγιστου αθροίσματος**

**Μέγιστο 🡨 0**

**ΓΙΑ** i **ΑΠΟ** 1 **ΜΕΧΡΙ** 10

**Άθροισμα 🡨 0**

ΓΙΑ **j** ΑΠΟ **1** ΜΕΧΡΙ **7**

**136 / 163-164**

**Άθροισμα 🡨 Άθροισμα+Εισπράξεις [i,j]** ΤΕΛΟΣ\_ΕΠΑΝΑΛΗΨΗΣ

ΓΡΑΨΕ **'Συνολ. είσπραξη ',Ονόματα[i], '=' , Άθροισμα**

ΑΝ **Μέγιστο < Άθροισμα** ΤΟΤΕ

**Μέγιστο 🡨 Άθροισμα**

**θέση 🡨 i**

ΤΕΛΟΣ\_ΑΝ

ΤΕΛΟΣ ΕΠΑΝΑΛΗΨΗΣ

**ΓΡΑΨΕ 'Μέγ.συνολ.είσπραξη ',Μέγιστο, 'στον ',Ονόματα[Θέση]**

**!Άθροισμα των στοιχείων του πίνακα Εισπράξεις ανά στήλη**

**!και υπολογισμός του μέγιστου αθροίσματος**

**Μέγιστο 🡨 0**

**ΓΙΑ j ΑΠΟ 1 ΜΕΧΡΙ 7**

**Άθροισμα 🡨 0**

**ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 10**

**Άθροισμα 🡨 Άθροισμα + Εισπράξεις[i,j]**

**ΤΕΛΟΣ\_ΕΠΑΝΑΛΗΨΗΣ**

**ΓΡΑΨΕ 'Συνολ. είσπραξη',j, '-ης ημέρας = ', Άθροισμα**

**ΑΝ Μέγιστο < Άθροισμα ΤΟΤΕ**

**Μέγιστο 🡨 Άθροισμα**

**θέση 🡨 j**

**ΤΕΛΟΣ\_ΑΝ**

**ΤΕΛΟΣ\_ΕΠΑΝΑΛΗΨΗΣ**

**ΓΡΑΨΕ 'Μέγ.συν.είσπραξη ',Μέγιστο, 'την ',Θέση, '-η ημέρα'**

**ΤΕΛΟΣ ΠΡΟΓΡΑΜΜΑΤΟΣ**

**Προγραμματιστικό περιβάλλον Pascal**

**PROGRAM cinemas;**

**VAR**

**i,j,k,max,sum:INTEGER;**

**tickets:ARRAY[1..10,1..7] OF INTEGER;**

**name:ARRAY [1..10] OF STRING;**

**137 / 164**

**BEGIN**

**FOR i:=1 TO 10 do BEGIN**

**write ('Δώσε το όνομα : ');**

**readln (name[i]);**

**FOR j:=1 TO 7 DO**

**BEGIN**

**write ('Δώσε την ' j,' η είσπραξη : '); readln(tickets[i,j]);**

**END;**

**END;**

**max:=0;**

**FOR i:=1 TO 10 DO**

**BEGIN**

**sum:=0;**

**FOR j:=1 TO 7 DO**

**sum:=sum+tickets[i,j];**

**writeln('Σύνολο ', name[i],' = ', sum);**

**IF max<sum THEN**

**BEGIN**

**max:=sum;**

**k:=i;**

**END;**

**END;**

**writeln ('Μέγιστο ', max, ' στον ', name[k]);**

**max:=0;**

**FOR j:=1 TO 7 DO**

**BEGIN**

**sum:=0;**

**FOR i:=1 TO 10 DO**

**sum:=sum+tickets[i,j];**

**writeln('Σύνολο ',j'ημέρας =', sum);**

**IF max<sum THEN**

**BEGIN**

**max:=sum;**

**k:=j;**

**138 / 165**

**END;**

**END;**

**writeln ('Μέγιστο ', max, ' την ', k);**

**END.**

|  |  |
| --- | --- |
| **9.4** | **Τυπικές επεξεργασίες πινάκων** |

**Τα προγράμματα τα οποία χρησιμοποιούν πίνακες πολύ συχνά απαιτούν συγκεκριμένες επεξεργασίες στα στοιχεία του πίνακα. Οι τυπικές αυτές επεξεργασίες είναι:**

* **Υπολογισμός αθροισμάτων στοιχείων του πίνακα.**
* **Εύρεση του μέγιστου ή του ελάχιστου στοιχείου.**
* **Ταξινόμηση των στοιχείων του πίνακα.**
* **Αναζήτηση ενός στοιχείου του πίνακα.**
* **Συγχώνευση δύο πινάκων.**

**Μερικές από αυτές τις επεξεργασίες παρουσιάστηκαν ήδη στα παραδείγματα αυτού του κεφαλαίου. Για τις διαδικασίες της ταξινόμησης και της αναζήτησης έχουν παρουσιαστεί μερικοί αλγόριθμοι, οι οποίοι αναπτύ-χθηκαν στο αντίστοιχο κεφάλαιο των αλγορίθμων, όπου συζητήθηκαν τα πλεονεκτήματα και τα μειονεκτήματα τους.**

**Συνοπτικά έχουμε:**

**139 / 165-166**

**Υπολογισμός αθροισμάτων στοιχείων του πίνακα.**

**Πολύ συχνά απαιτείται ο υπολογισμός του αθροίσματος στοιχείων του πίνακα που έχουν κοινά χαρακτηριστικά για παράδειγμα βρίσκονται στην ίδια στήλη ή στην ίδια γραμμή. Τα παραδείγματα που παρουσιάστηκαν σε αυτό το κεφάλαιο απαιτούσαν τον υπολογισμό των αθροισμάτων των στοιχείων του πίνακα τόσο ανά γραμμή όσο και ανά στήλη.**

**140 / 166**

**Εύρεση του μέγιστου ή του ελάχιστου στοιχείου.**

**Η εύρεση του μέγιστου ή του ελάχιστου στοιχείου ενός πίνακα παρουσιάστηκε στα προηγούμενα παραδείγ-ματα. Αν ο πίνακας δεν είναι ταξινομημένος, τότε πρέπει να συγκριθούν τα στοιχεία ένα προς ένα, για να βρεθεί το μέγιστο ή το ελάχιστο. Αν ο πίνακας είναι ταξινομημένος, τότε προφανώς το μέγιστο και το ελάχιστο βρίσκονται στα δύο ακριανά στοιχεία του πίνακα.**

**Ταξινόμηση των στοιχείων του πίνακα.**

**Στο κεφάλαιο 3 αναφέρθηκε η μέθοδος ταξινόμησης της ευθείας ανταλλαγής, η οποία χρησιμοποιήθηκε και στο παράδειγμα 2.**

**Η μέθοδος αυτή είναι από τις απλούστερες αλλά δεν είναι η πιο αποδοτική. Υπάρχουν πολλές άλλες μέθοδοι ταξινόμησης καθώς και παραλλαγές αυτών.**

**Η επιλογή του καλύτερου αλγόριθμου εξαρτάται κυρίως από το πλήθος των στοιχείων του πίνακα και την αρχική τους διάταξη, αν δηλαδή ο πίνακας είναι τελείως αταξινόμητος ή μερικώς ταξινομημένος.**

**Αναζήτηση ενός στοιχείου του πίνακα.**

**Δύο είναι οι πλέον διαδεδομένοι αλγόριθμοι ανα-ζήτησης:**

1. **Η σειριακή αναζήτηση**
2. **Η δυαδική αναζήτηση**

**Η σειριακή μέθοδος αναζήτησης είναι η πιο απλή, αλλά και η λιγότερη αποτελεσματική μέθοδος. Χρησιμοποι-είται όμως υποχρεωτικά για πίνακες που δεν είναι ταξινομημένοι. Αντίθετα η δυαδική αναζήτηση χρησι-μοποιείται μόνο σε ταξινομημένους πίνακες και είναι σαφώς αποδοτικότερη από τη σειριακή μέθοδο.**

**Συγχώνευση δύο πινάκων.**

**Η συγχώνευση είναι μία από τις βασικές λειτουργίες σε πίνακες.**

**Σκοπός της είναι η δημιουργία από τα στοιχεία δυο (ή περισσότερων) ταξινομημένων πινάκων ενός άλλου, που είναι και αυτός ταξινομημένος.**

**141 / 166**

**• Ανακεφαλαίωση**

**Στο κεφάλαιο αυτό παρουσιάστηκαν οι πίνακες και πώς η ΓΛΩΣΣΑ χειρίζεται τους πίνακες. Ο πίνακας είναι μία ομάδα μεταβλητών ιδίου τύπου που αναφέρονται με ένα κοινό όνομα και αποθηκεύονται σε διαδοχικές θέσεις στη μνήμη. Για την πρόσβαση σε ένα ατομικό στοιχείο του πίνακα πρέπει να γραφεί το όνομα του πίνακα ακολουθούμενο από ένα δείκτη (μεταβλητή ή σταθερά). Οι πίνακες μπορεί να είναι μονοδιάστατοι, δισδιάστατοι ή γενικότερα πολυδιάστατοι. Ο αριθμός των δεικτών καθορίζει τη διάσταση του πίνακα. Η επεξεργασία πινάκων γίνεται συνήθως με τη χρήση εντολών ΓΙΑ.**

**Οι τυπικές επεξεργασίες που γίνονται σε πίνακες περιλαμβάνουν την αναζήτηση, την ταξινόμηση και τη συγχώνευση πινάκων. Για αυτές τις επεξεργασίες έχουν αναπτυχθεί αρκετοί αλγόριθμοι και η μελέτη τους αποτελεί έναν από τους σημαντικούς τομείς της αλγοριθμικής.**

**142 / 167**

**Λέξεις κλειδιά**

**Μεταβλητή με δείκτη, Πίνακας, Στοιχείο πίνακα, Αναζήτηση, Ταξινόμηση, Συγχώνευση**

**• Ερωτήσεις Θέματα για συζήτηση**

**1. Τι ονομάζεται πίνακας;**

**2. Για ποιο λόγο χρησιμοποιούνται οι πίνακες;**

**3. Τι μπορεί να είναι οι δείκτες των πινάκων;**

**4. Ποια η διαφορά του πίνακα και του στοιχείου ενός πίνακα;**

**5. Πού ορίζεται η διάσταση του πίνακα;**

**6. Τι είδους δομή είναι ο πίνακας;**

**7. Ποιοι πίνακες ονομάζονται μονοδιάστατοι;**

**8. Δώσε ένα παράδειγμα ενός τρισδιάστατου πίνακα.**

**9. Πού αποθηκεύονται τα στοιχεία ενός πίνακα;**

**10. Ποια τα μειονεκτήματα της χρήσης πινάκων;**

**143 / 167**

**11. Ποιες οι τυπικές επεξεργασίες πίνακα;**

**12. Ποιοι είναι οι πιο διαδεδομένοι αλγόριθμοι αναζήτησης; Ποιες οι διαφορές τους;**

**• Βιβλιογραφία**

**1. Κ. Αντωνακόπουλος, Turbo Pascal 6.0 Θεωρία και Πράξη, Εκδόσεις Νέων Τεχνολογιών, Αθήνα, 1993.**

**2. QuickBASIC: Η εργαλειοθήκη του προγραμματιστή, Κλειδάριθμος, Αθήνα, 1991.**

**3. Χρ. Κοίλιας, Η QuickBasic και οι εφαρμογές της, Εκδόσεις Νέων Τεχνολογιών, Αθήνα, 1992.**

**4. R. Shackelford, Introduction to Computing and Algorithms, Addison-Wesley, USA,1998.**

**5. S. Leestma-L.Nyhoff, Turbo Pascal, Programming and Solving, McMillan, NewYork,1990.**

**6. N. Wirth, Systematic Programming: An introduction, Prentice Hall, 1973.**

**Διαδίκτυο**

[**http://qbasic.com/**](http://qbasic.com/)

**Περιέχει εκπαιδευτικό οδηγό, κώδικα πολλών ασκήσεων και γενικές πληροφορίες για την Qbasic.**

**144 / 167**

**Επίσης στο διαδίκτυο παρουσιάζουν ενδιαφέρον ομάδες νέων (Usenet), σχετικές με τις γλώσσες προγραμματισμού.**

**Περιεχόμενα 2ου τόμου**

|  |  |  |
| --- | --- | --- |
| **6.** | **Εισαγωγή στον προγραμματισμό** | **5** |
|  | **6.1 Η έννοια του προγράμματος** | **8** |
|  | **6.2 Ιστορική αναδρομή** | **9** |
|  | **6.2.1 Γλώσσες μηχανής** | **10** |
|  | **6.2.2 Συμβολικές γλώσσες ή γλώσσες χαμηλού επιπέδου** | **11** |
|  | **6.2.3 Γλώσσες υψηλού επιπέδου** | **14** |
|  | **6.2.4 Γλώσσες 4ns γενιάς** | **29** |
|  | **6.3 Φυσικές και τεχνητές γλώσσες** | **34** |
|  | **6.4 Τεχνικές σχεδίασης προγραμμάτων** | **37** |
|  | **6.4.1 Ιεραρχική σχεδίαση προγράμματος** | **37** |
|  | **6.4.2 Τμηματικός προγραμματισμός** | **39** |
|  | **6.4.3 Δομημένος προγραμματισμός** | **40** |
|  | **6.5 Αντικειμενοστραφής προγραμματισμός** | **44** |
|  | **6.6 Παράλληλος προγραμματισμός** | **46** |
|  | **6.7 Προγραμματιστικά περιβάλλοντα** | **47** |
| **7.**  **145** | **Εισαγωγή στον προγραμματισμό** | **57** |
|  | **7.1 Το αλφάβητο της ΓΛΩΣΣΑΣ**  **146** | **61** |
|  | **7.2 Τύποι δεδομένων** | **62** |
|  | **7.3 Σταθερές** | **63** |
|  | **7.4 Μεταβλητές** | **66** |
|  | **7.5 Αριθμητικοί τελεστές** | **68** |
|  | **7.6 Συναρτήσεις** | **70** |
|  | **7.7 Αριθμητικές εκφράσεις** | **71** |
|  | **7.8 Εντολή εκχώρησης** | **73** |
|  | **7.9 Εντολές εισόδου-εξόδου** | **74** |
|  | **7.10 Δομή προγράμματος** | **76** |
| **8.** | **Επιλογή και επανάληψη** | **83** |
|  | **8.1 Εντολές Επιλογής** | **86** |
|  | **8.1.1 Εντολή ΑΝ** | **89** |
|  | **8.1.2 Εντολή ΕΠΙΛΕΞΕ** | **97** |
|  | **8.2 Εντολές επανάληψης** | **99** |
|  | **8.2.1 Εντολή ΟΣΟ...ΕΠΑΝΑΛΑΒΕ** | **99** |
|  | **8.2.2 Εντολή ΜΕΧΡΙΣ\_ΟΤΟΥ** | **102** |
|  | **8.2.3 Εντολή ΓΙΑ...ΑΠΟ...ΜΕΧΡΙ**  **147** | **107** |
| **9.** | **. Πίνακες** | **115** |
|  | **9.1. Μονοδιάστατοι πίνακες** | **118** |
|  | **9.2. Πότε πρέπει να χρησιμοποιούνται πίνακες** | **128** |
|  | **9.3. Πολυδιάστατοι πίνακες** | **129** |
|  | **9.4. Τυπικές επεξεργασίες πινάκων** | **139** |

**Βάσει του ν. 3966/2011 τα διδακτικά βιβλία του Δημοτικού, του Γυμνασίου, του Λυκείου, των ΕΠΑ.Λ. και των ΕΠΑ.Σ. τυπώνονται από το ΙΤΥΕ - ΔΙΟΦΑΝΤΟΣ και διανέμονται δωρεάν στα Δημόσια Σχολεία. Τα βιβλία μπορεί να διατίθενται προς πώληση, όταν φέρουν στη δεξιά κάτω γωνία του εμπροσθόφυλλου ένδειξη «ΔIΑΤΙΘΕΤΑΙ ΜΕ ΤΙΜΗ ΠΩΛΗΣΗΣ». Κάθε αντίτυπο που διατίθεται προς πώληση και δεν φέρει την παραπάνω ένδειξη θεωρείται κλεψίτυπο και ο παραβάτης διώκεται σύμφωνα με τις διατάξεις του άρθρου 7 του νόμου 1129 της 15/21 Μαρτίου 1946 (ΦΕΚ 1946, 108, Α').**

**Απαγορεύεται η αναπαραγωγή οποιουδήποτε τμήματος αυτού του βιβλίου, που καλύπτεται από δικαιώματα (copyright), ή η χρήση του σε οποιαδήποτε μορφή, χωρίς τη γραπτή άδεια του Υπουργείου Παιδείας και Θρησκευμάτων / IΤΥΕ - ΔΙΟΦΑΝΤΟΣ.**