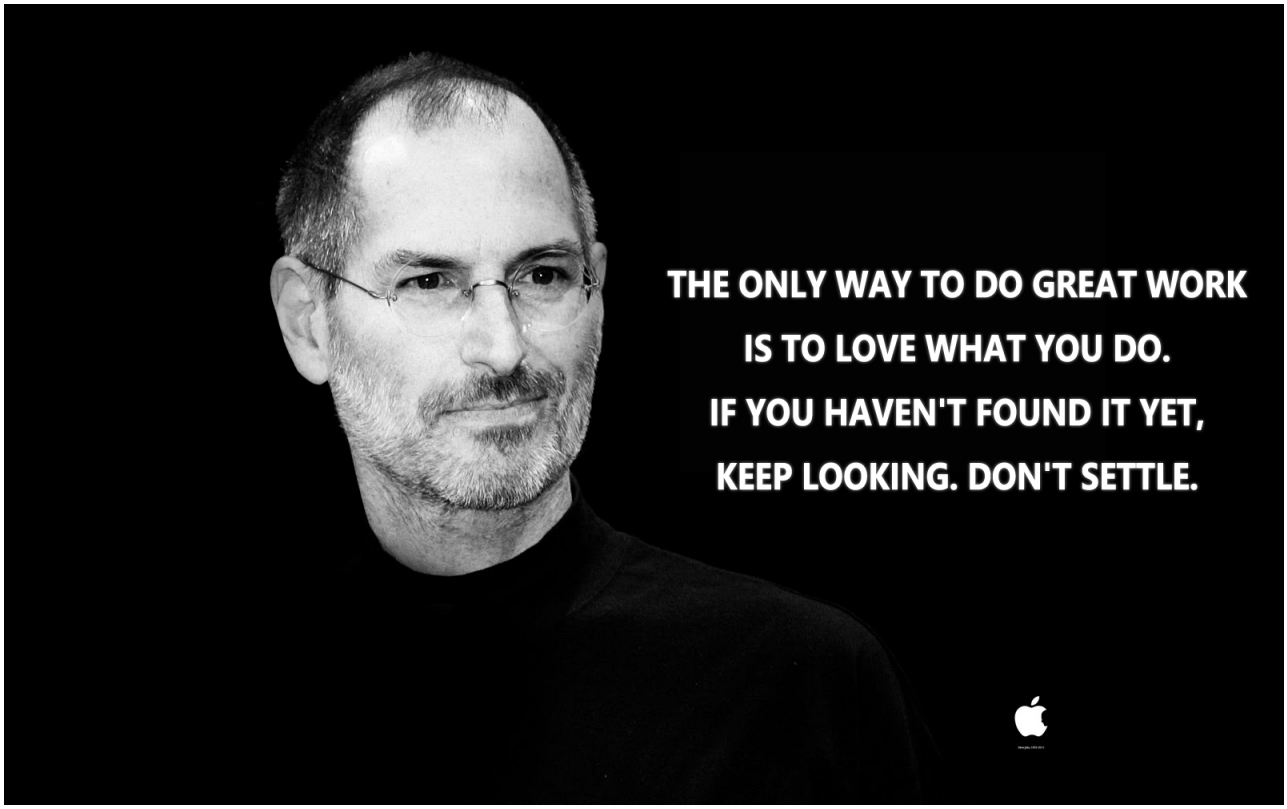
Επιστήμες Οικονομίας και Πληροφορικής

ΑΝΑΠΤΥΞΗ ΕΦΑΡΜΟΓΩΝ
σε
ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΟ ΠΕΡΙΒΑΛΛΟΝ



Επιμέλεια: Γιάννης Βελέντζας (gbelentzas@gmail.com)

Περιεχόμενα

1 Ανάλυση προβλήματος **5**
- 1.1 Η έννοια πρόβλημα . . . 5
- 1.2 Κατανόηση προβλήματος . . . 5
- 1.3 Δομή προβλήματος . . . 6
- 1.4 Καθορισμός απαιτήσεων . . . 8

2 Βασικές Εννοιες Αλγορίθμων **11**
- 2.1 Τι είναι αλγόριθμος . . . 11
- 2.2 Η Ψευδογλώσσα . . . 13
- 2.3 Δομή ακολουθίας . . . 18
- 2.4 Δομή επιλογής . . . 20
- 2.5 Δομή επανάληψης . . . 25

3 Δομές Δεδομένων και Αλγόριθμοι **33**
- 3.1 Δεδομένα . . . 33
- 3.2 Σπουδαιότητα αλγορίθμων . . . 35
- 3.3 Πίνακες . . . 37
- 3.4 Στοίβα . . . 39
- 3.5 Ουρά . . . 41
- 3.6 Αναζήτηση . . . 43
- 3.7 Ταξινόμηση . . . 46
- 3.8 Αλλες δομές δεδομένων . . . 48
  - 3.8.1 Λίστες . . . 48
  - 3.8.2 Δένδρα . . . 53
  - 3.8.3 Γράφοι . . . 58

4 Τεχνικές σχεδίασης αλγορίθμων **61**
- 4.1 Ανάλυση προβλημάτων . . . 61
- 4.2 Μέθοδος Διαίρει και Βασίλευε . . . 62

5 Ανάλυση αλγορίθμων **65**

6 Εισαγωγή στον Προγραμματισμό **67**
- 6.1 Η έννοια του προγράμματος . . . 67
- 6.2 Φυσικές και τεχνητές γλώσσες . . . 67
- 6.3 Τεχνικές σχεδίασης προγραμμάτων . . . 69
- 6.4 Αντικειμενοστραφής προγραμματισμός . . . 71

6.5 Προγραμματιστικά περιβάλλοντα 71

7 Βασικά στοιχεία προγραμματισμού 75
7.1 Το αλφάβητο της ΓΛΩΣΣΑΣ 75
7.2 Τύποι δεδομένων 75
7.3 Σταθερές 76
7.4 Μεταβλητές 77
7.5 Αριθμητικοί τελεστές 78
7.6 Συναρτήσεις 79
7.7 Αριθμητικές εκφράσεις 79
7.8 Εντολή εκχώρησης 80
7.9 Εντολές εισόδου-εξόδου 81
7.10 Δομή προγράμματος 82

8 Επιλογή και επανάληψη 85
8.1 Εντολές Επιλογής 85
8.2 Εντολές επανάληψης 86

9 Πίνακες 89
9.1 Μονοδιάστατοι πίνακες 89
9.2 Πότε πρέπει να χρησιμοποιούνται πίνακες 90
9.3 Τυπικές επεξεργασίες πινάκων 90

10 Υποπρογράμματα 93
10.1 Τμηματικός προγραμματισμός 93
10.2 Χαρακτηριστικά των υποπρογραμμάτων 93
10.3 Πλεονεκτήματα του τμηματικού προγραμματισμού 94
10.4 Παράμετροι 95
10.5 Διαδικασίες και συναρτήσεις 95
10.6 Πραγματικές και τυπικές παράμετροι 96
10.7 Εμβέλεια μεταβλητών-σταθερών. 98

11 Σύγχρονα προγραμματιστικά περιβάλλοντα 101

12 Σχεδίαση διεπαφής χρήστη 109

13 Εκσφαλμάτωση προγράμματος 111

Κεφάλαιο 1

Ανάλυση προβλήματος

1.1 Η έννοια πρόβλημα

Ερώτηση 1.1

Τι είναι πρόβλημα;

Το πρόβλημα αποτελεί έννοια που απαντάται σε όλες τις επιστήμες και τους κλάδους τους, αλλά παράλληλα και στην καθημερινή μας ζωή.

Πρόβλημα είναι μια κατάσταση η οποία χρήζει αντιμετώπισης, απαιτεί λύση, η δε λύση της δεν είναι γνωστή, ούτε προφανής.

1.2 Κατανόηση προβλήματος

Ερώτηση 1.2

Τι είναι η κατανόηση ενός προβλήματος; Από τι εξαρτάται η κατανόηση ενός προβλήματος;

Η οποιαδήποτε προσπάθεια αντιμετώπισης ενός προβλήματος είναι καταδικασμένη σε αποτυχία αν προηγουμένως δεν έχει γίνει απόλυτα κατανοητό το πρόβλημα που τίθεται.

Κατανόηση ενός προβλήματος είναι η πλήρης και σαφής αντίληψη, που παρέχει τη δυνατότητα γνώσης, διάκρισης και συσχετισμού δεδομένων.

Η κατανόηση ενός προβλήματος αποτελεί συνάρτηση δύο παραγόντων:

- τη σωστή διατύπωση εκ μέρους του δημιουργού του και
- τη σωστή ερμηνεία από τη μεριά εκείνου που καλείται να το αντιμετωπίσει.

Οποιοδήποτε μέσο μπορεί να χρησιμοποιηθεί για να αποδοθεί η διατύπωση ενός προβλήματος. Συνηθέστερο από όλα είναι ο λόγος, είτε ο προφορικός, είτε ο γραπτός.

Ερώτηση 1.3

Πότε μπορεί να υπάρξει ασάφεια της διατύπωσης ενός προβλήματος;

Οποιοδήποτε μέσο μπορεί να χρησιμοποιηθεί για να αποδοθεί η διατύπωση ενός προβλήματος. Συνηθέστερο από όλα είναι ο λόγος, είτε ο προφορικός, είτε ο γραπτός.

Ο λόγος σαν μέσο επικοινωνίας και συνεννόησης πρέπει να χαρακτηρίζεται από σαφήνεια. Αστοχη χρήση ορολογίας, λανθασμένη σύνταξη, είναι δύο στοιχεία που μπορούν να προκαλέσουν παρερμηνείες και παραπλανήσεις. Η παρερμηνεία είναι δυνατή ακόμα και σε περιπτώσεις όπου όλοι οι λεξικολογικοί και συντακτικοί κανόνες κρατούνται με ευλάβεια.

Ερώτηση 1.4

Από τι εξαρτάται η σωστή ερμηνεία ενός προβλήματος;

Η σωστή ερμηνεία ενός προβλήματος εξαρτάται από:

1. Τις γνώσεις
2. Την εμπειρία
3. Την ευφυΐα
4. Το ενδιαφέρον του λύτη

Ερώτηση 1.5

Τι σημαίνει η έννοια «χώρος» προβλήματος;

Το σύνολο των ερωτήσεων (τι, πως, ποιος, πότε, που) και απαντήσεων που σχετίζονται με το πρόβλημα.

1.3 Δομή προβλήματος

Ερώτηση 1.6

Να δώσετε τους ορισμούς: Δεδομένο, Πληροφορία, Επεξεργασία δεδομένων

Με τον όρο **δεδομένο** δηλώνεται οποιαδήποτε στοιχείο μπορεί να γίνει αντιληπτό από έναν τουλάχιστον παρατηρητή με μία από τις πέντε αισθήσεις.

Με τον όρο **πληροφορία** αναφέρεται οποιαδήποτε γνωσιακό στοιχείο προέρχεται από επεξεργασία δεδομένων.

Ο όρος **επεξεργασία δεδομένων** δηλώνει εκείνη τη διαδικασία κατά την οποία ένας ”μηχανισμός” δέχεται δεδομένα, τα επεξεργάζεται σύμφωνα με έναν προκαθορισμένο τρόπο και αποδίδει πληροφορίες.

Επί χιλιετίες ο ”μηχανισμός” επεξεργασίας των δεδομένων ήταν και εξακολουθεί να είναι ο ανθρώπινος εγκέφαλος. Στις μέρες μας, ένας άλλος ”μηχανισμός” επεξεργασίας δεδομένων είναι ο υπολογιστής.

Ερώτηση 1.7

Τι είναι η ανάλυση ενός προβλήματος;

Η κατανόηση του προβλήματος είναι βασική προϋπόθεση για να γίνει στη συνέχεια δυνατή η σωστή αποτύπωση της δομής του. Η καταγραφή της δομής ενός προβλήματος σημαίνει αυτόματα ότι έχει αρχίσει η διαδικασία ανάλυσης του προβλήματος σε άλλα απλούστερα. Με τη σειρά τους τα νέα προβλήματα μπορούν να αναλυθούν σε άλλα, ακόμη πιο απλά.

Η διαδικασία αυτή της ανάλυσης μπορεί να συνεχιστεί μέχρις ότου τα επιμέρους προβλήματα που προέκυψαν θεωρηθούν αρκετά απλά και η αντιμετώπισή τους χαρακτηριστεί ως δυνατή.

Η δυσκολία αντιμετώπισης των προβλημάτων ελαττώνεται όσο περισσότερο προχωράει η ανάλυση τους σε απλούστερα προβλήματα. Ο κατακερματισμός ενός προβλήματος σε άλλα απλούστερα είναι μια από τις διαδικασίες που ενεργοποιούν και αμβλύνουν τόσο τη σκέψη, αλλά κυρίως την αναλυτική ικανότητα .του ατόμου.

Ερώτηση 1.8

Τι ονομάζουμε δομή ενός προβλήματος;

Με τον όρο δομή ενός προβλήματος αναφερόμαστε στα συστατικά του μέρη, στα επιμέρους τμήματα που το αποτελούν καθώς επίσης και στον τρόπο που αυτά τα μέρη συνδέονται μεταξύ τους.

Ερώτηση 1.9

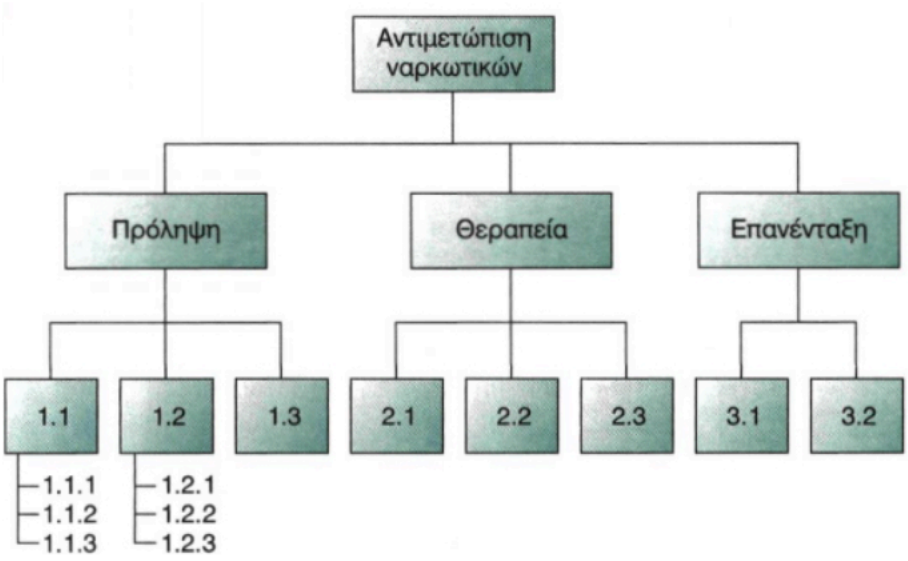
Πως μπορεί να γίνει η παρουσίαση της ανάλυσης ενός προβλήματος;

Η παρουσίαση μπορεί να γίνει με δύο τρόπους:

1. **Φραστικά**, δηλαδή περιγραφή με λόγια πώς και σε ποια υποπροβλήματα αναλύεται το πρόβλημα και τα υποπροβλήματα που προκύπτουν.

2. **Διαγραμματική αναπαράσταση**, δηλαδή με διάγραμμα σε σχήμα γενεαλογικού δέντρου, στο οποίο κάθε πρόβλημα έχει παιδιά τα υποπροβλήματα στα οποία αναλύεται. Σύμφωνα με αυτή:

   - το αρχικό πρόβλημα αναπαρίσταται από ένα ορθογώνιο παραλληλόγραμμο
   - κάθε ένα από τα απλούστερα προβλήματα στα οποία αναλύεται ένα οποιοδήποτε πρόβλημα, αναπαρίσταται επίσης από ένα ορθογώνιο παραλληλόγραμμο
   - τα παραλληλόγραμμα που αντιστοιχούν στα απλούστερα προβλήματα στα οποία αναλύεται ένα οποιοδήποτε πρόβλημα, σχηματίζονται ένα επίπεδο χαμηλότερα. Ετσι σε κάθε κατώτερο επίπεδο, δημιουργείται η γραφική αναπαράσταση των προβλημάτων στα οποία αναλύονται τα προβλήματα του αμέσως ψηλότερου επιπέδου.



Σχ. 1.1. Διαγραμματική αναπαράσταση προβλήματος "Αντιμετώπιση ναρκωτικών"

1.4 Καθορισμός απαιτήσεων

Ερώτηση 1.10

Πως χρησιμοποιείται η ανάλυση και η σύνθεση στην επίλυση ενός προβλήματος;

Η επίλυση ενός προβλήματος με τη βοήθεια της ανάλυσης και της σύνθεσης γίνεται σε τρεις φάσεις:

1. Το πρόβλημα που έχουμε να επιλύσουμε αναλύεται σε πιο απλά προβλήματα, τα οποία ονομάζουμε υποπροβλήματα.
2. Επίλυση καθενός από τα υποπροβλήματα.
3. Σύνθεση των λύσεων των υποπροβλημάτων, προκειμένου να επιλυθεί το αρχικό πρόβλημα.

Ερώτηση 1.11

Σε τι μπορούμε να χωρίσουμε τον «χώρο» ενός προβλήματος;

Ο χώρος ενός προβλήματος χωρίζεται σε δύο περιοχές, στα δεδομένα και τα ζητούμενα.

Ερώτηση 1.12

Τι ονομάζουμε ζητούμενο;

Ζητούμενο αυτό που ψάχνουμε, το σημείο στο οποίο πρέπει να φτάσουμε, για να βγούμε από την «προβληματική» κατάσταση στην οποία βρισκόμαστε.

Ερώτηση 1.13

Τι απαιτεί ο καθορισμός των απαιτήσεων;

Η σωστή επίλυση ενός προβλήματος προϋποθέτει τον *επακριβή προσδιορισμό των δεδομένων* που παρέχει το πρόβλημα. Απαιτεί επίσης την *λεπτομερειακή καταγραφή των ζητούμενων* που αναμένονται σαν αποτελέσματα της επίλυσης του προβλήματος.

Θα πρέπει να δοθεί μεγάλη προσοχή στην ανίχνευση των δεδομένων ενός προβλήματος. Επισημαίνεται πως δεν είναι πάντοτε εύκολο να διακρίνει κάποιος τα δεδομένα. Υπάρχουν πολλές περιπτώσεις προβλημάτων όπου τα δεδομένα θα πρέπει να "ανακαλυφθούν" μέσα στα λεγόμενα του προβλήματος.

Η διαδικασία αυτή απαιτεί προσοχή, συγκέντρωση και σκέψη. Μεθοδολογία προσδιορισμού των δεδομένων ενός προβλήματος δεν υπάρχει, ούτε και μεθοδολογία εντοπισμού και αποσαφήνισης των ζητούμενων ενός προβλήματος.

Το ίδιο προσεκτικά θα πρέπει να αποσαφηνιστούν και τα ζητούμενα του προβλήματος. Δεν είναι πάντοτε ιδιαίτερα κατανοητό τι ακριβώς ζητάει ένα πρόβλημα. Σε μια τέτοια περίπτωση θα πρέπει να θέτονται μια σειρά από ερωτήσεις με στόχο την διευκρίνηση πιθανών απορριών σχετικά με τα ζητούμενα, τον τρόπο παρουσίασης τους, το εύρος τους κ.λπ. Οι ερωτήσεις αυτές μπορούν να απευθύνονται είτε στο δημιουργό του προβλήματος, είτε στον ίδιο μας τον εαυτό αν εμείς καλούμαστε να αντιμετωπίσουμε το πρόβλημα.

Ερώτηση 1.14

Ποια είναι τα στάδια αντιμετώπισης ενός προβλήματος;

1. **Κατανόηση**, όπου απαιτείται η σωστή και πλήρης αποσαφήνιση των δεδομένων και των ζητούμενων του προβλήματος.
2. **Ανάλυση**, όπου το αρχικό πρόβλημα διασπάται σε άλλα επιμέρους απλούστερα προβλήματα.
3. **Επίλυση**, όπου υλοποιείται η λύση του προβλήματος, μέσω της λύσης των επιμέρους προβλημάτων.

Κεφάλαιο 2

Βασικές Εννοιες Αλγορίθμων

2.1 Τι είναι αλγόριθμος

Ερώτηση 2.1

Τι είναι αλγόριθμος;

Ο όρος αλγόριθμος, λοιπόν, χρησιμοποιείται για να δηλώσει μεθόδους που εφαρμόζονται για την επίλυση προβλημάτων. Ωστόσο, ένας πιο αυστηρός ορισμός της έννοιας αυτής είναι ο εξής:

Αλγόριθμος είναι μια πεπερασμένη σειρά ενεργειών, αυστηρά καθορισμένων και εκτελέσιμων σε πεπερασμένο χρόνο, που στοχεύουν στην επίλυση ενός προβλήματος.

Ερώτηση 2.2

Ποια κριτήρια πρέπει να ικανοποιεί ένας αλγόριθμος (ανάλυση εννοιών);

Κάθε αλγόριθμος απαραίτητα ικανοποιεί τα επόμενα κριτήρια.

1. **Είσοδος** (input). Καμία, μία ή περισσότερες τιμές δεδομένων πρέπει να δίνονται ως είσοδοι στον αλγόριθμο. Η περίπτωση που δεν δίνονται τιμές δεδομένων εμφανίζεται, όταν ο αλγόριθμος δημιουργεί και επεξεργάζεται κάποιες πρωτογενείς τιμές με τη βοήθεια συναρτήσεων παραγωγής τυχαίων αριθμών ή με τη βοήθεια άλλων απλών εντολών.

2. **Εξοδος** (output). Ο αλγόριθμος πρέπει να δημιουργεί τουλάχιστον μία τιμή δεδομένων ως αποτέλεσμα προς το χρήστη ή προς έναν άλλο αλγόριθμο.

3. **Καθοριστικότητα** (definiteness). Κάθε εντολή πρέπει να καθορίζεται χωρίς καμία αμφιβολία για τον τρόπο εκτέλεσης της. Λόγου χάριν, μία εντολή διαίρεσης πρέπει να θεωρεί και την περίπτωση, όπου ο διαιρέτης λαμβάνει μηδενική τιμή.

4. **Περατότητα** (finiteness). Ο αλγόριθμος να τελειώνει μετά από πεπερασμένα βήματα εκτέλεσης των εντολών του. Μία διαδικασία που δεν τελειώνει μετά από ένα συγκεκριμένο αριθμό βημάτων δεν αποτελεί αλγόριθμο, αλλά λέγεται απλά υπολογιστική διαδικασία (computational procedure).

5. **Αποτελεσματικότητα** (effectiveness). Κάθε μεμονωμένη εντολή του αλγορίθμου να είναι απλή. Αυτό σημαίνει ότι μία εντολή δεν αρκεί να έχει ορισθεί, αλλά πρέπει να είναι και εκτελέσιμη.

Ερώτηση 2.3

Η έννοια του αλγορίθμου συνδέεται αποκλειστικά και μόνο με προβλήματα της Πληροφορικής; Δικαιολογείστε την απάντησή σας.

Η έννοια του αλγόριθμου δεν συνδέεται αποκλειστικά και μόνο με προβλήματα της Πληροφορικής, όπως για παράδειγμα όταν θέλουμε να γευματίσουμε και επομένως πρέπει να εκτελέσουμε κάποιες ενέργειες.

Βέβαια, αυτή η αλληλουχία των ενεργειών δεν είναι η μοναδική για την επίτευξη του σκοπού, δηλαδή μπορούν να υπάρχουν διαφορετικοί αλγόριθμοι επίλυσης ενός προβλήματος.

Ερώτηση 2.4

Ποιοι οι τρόποι παράστασης ενός αλγορίθμου; (ανάλυση τρόπων)

Ενας αλγόριθμος μπορεί να παρασταθεί με έναν από τους ακόλουθους τέσσερις τρόπους:

1. με **ελεύθερο κείμενο** (free text), που αποτελεί τον πιο ανεπεξέργαστο και αδόμητο τρόπο παρουσίασης αλγορίθμου. Ετσι εγκυμονεί τον κίνδυνο ότι μπορεί εύκολα να οδηγήσει σε μη εκτελέσιμη παρουσίαση παραβιάζοντας το τελευταίο χαρακτηριστικό των αλγορίθμων, δηλαδή την αποτελεσματικότητα.

2. με **φυσική γλώσσα** (natural language) κατά βήματα. Στην περίπτωση αυτή χρειάζεται προσοχή, γιατί μπορεί να παραβιασθεί το τρίτο βασικό χαρακτηριστικό ενός αλγορίθμου, όπως προσδιορίσθηκε προηγουμένως, δηλαδή το κριτήριο του καθορισμού.

3. με **διαγραμματικές τεχνικές** (diagramming techniques), που συνιστούν ένα γραφικό τρόπο παρουσίασης του αλγορίθμου. Από τις διάφορες διαγραμματικές τεχνικές που έχουν επινοηθεί, η πιο παλιά και η πιο γνωστή ίσως, είναι το διάγραμμα ροής (flow chart). Ωστόσο η χρήση διαγραμμάτων ροής για την παρουσίαση αλγορίθμων δεν αποτελεί την καλύτερη λύση, γι' αυτό και εμφανίζονται όλο και σπανιότερα στη βιβλιογραφία και στην πράξη.

4. με **κωδικοποίηση** (coding), δηλαδή με ένα πρόγραμμα που όταν εκτελεσθεί θα δώσει τα ίδια αποτελέσματα με τον αλγόριθμο.

Ολοι οι αλγόριθμοι του βιβλίου είναι κωδικοποιημένοι σε μία υποθετική δομημένη ψευδογλώσσα, ωστόσο οι περισσότεροι από αυτούς μπορούν εύκολα σχετικά να προγραμματισθούν σε οποιαδήποτε γλώσσα προγραμματισμού.

Ερώτηση 2.5

Από τι αποτελείται ένα διάγραμμα ροής;

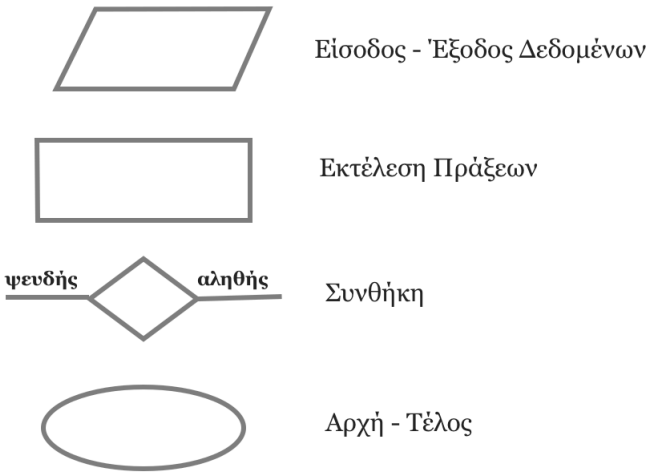
Ενα διάγραμμα ροής αποτελείται από ένα σύνολο γεωμετρικών σχημάτων, όπου το καθένα δηλώνει μία συγκεκριμένη ενέργεια ή λειτουργία. Τα γεωμετρικά σχήματα ενώνονται μεταξύ τους με βέλη, που δηλώνουν τη σειρά εκτέλεσης των ενεργειών αυτών.

Ερώτηση 2.6

Ποια είναι τα κυριότερα γεωμετρικά σχήματα που χρησιμοποιούμε στο διάγραμμα ροής και τι συμβολίζει το καθένα;

Τα κυριότερα γεωμετρικά σχήματα που χρησιμοποιούμε στο διάγραμμα ροής είναι:

- **Πλάγιο παραλληλόγραμμο**, που δηλώνει είσοδο ή έξοδο στοιχείων
- **Ορθογώνιο**, που δηλώνει την εκτέλεση μίας ή περισσοτέρων πράξεων
- **Ρόμβος**, που δηλώνει μία ερώτηση με δύο ή περισσότερες εξόδους
- **Ελλειψη**, που δηλώνει την αρχή και το τέλος του κάθε αλγορίθμου



2.2 Η Ψευδογλώσσα

Ερώτηση 2.7

Τι είναι η ψευδογλώσσα και τι εντολή αυτής;

Η ψευδογλώσσα είναι μια αλγοριθμική γλώσσα και περιλαμβάνει μια σειρά από εντολές που χρησιμοποιούνται για να είναι έυκολη η προσέγγιση ενός προβλήματος σε προγραμματιστικό περιβάλλον. Η ψευδογλώσσα μοιάζει πολύ με μια γλώσσα προγραμματισμού, ώστε να είναι εύκολη η υλοποίηση ενός οποιουδήποτε αλγορίθμου σε πρόγραμμα.

Ενας αλγόριθμος διατυπωμένος σε ψευδογλώσσα αρχίζει πάντα με τη λέξη **Αλγόριθμος** συνοδευόμενη με το όνομα του αλγορίθμου και τελειώνει με τη λέξη **Τέλος** συνοδευόμενη επίσης με το όνομα του αλγορίθμου.

Κάθε μία λέξη της χρησιμοποιούμενης ψευδογλώσσας, που προσδιορίζει μια σαφή ενέργεια, θα αποκαλείται στο εξής εντολή.

Ερώτηση 2.8

Από τι αποτελείται το αλφάβητο της ψευδογλώσσας;

- Κεφαλαία γράμματα ελληνικού αλφαβήτου
  ΑΒΓΔΕΖΗΘΙΚΛΜΝΞΟΠΡΣΤΥΦΧΨΩ
- Πεζά γράμματα ελληνικού αλφαβήτου
  αβγδεζηθικλμνξοπρστυφχψω
- Κεφαλαία γράμματα λατινικού αλφαβήτου
  ABCDEFGHIJKLMNOPQRSTUVWXYZ
- Πεζά γράμματα λατινικού αλφαβήτου
  abcdefghijklmnopqrstuvwxyz
- Αριθμητικά ψηφία
  0123456789
- Ειδικοί χαρακτήρες
  + - * / = < > ^ ( ) [ ] . , : ‘ ! & κενός χαρακτήρας

Ερώτηση 2.9

Ποια μεγέθη ονομάζονται σταθερές σε έναν αλγόριθμο;

Με τον όρο αυτό αναφερόμαστε σε προκαθορισμένες τιμές που παραμένουν αμετάβλητες σε όλη τη διάρκεια της εκτέλεσης ενός αλγορίθμου. Οι σταθερές διακρίνονται σε:

- αριθμητικές, π.χ. 123, +5, -1,25
- αλφαριθμητικές π.χ. “Τιμή”, “Κατάσταση αποτελεσμάτων”
- λογικές που είναι ακριβώς δύο, Αληθής και Ψευδής

Ερώτηση 2.10

Ποια μεγέθη ονομάζονται μεταβλητές σε έναν αλγόριθμο;

Μια μεταβλητή είναι ένα γλωσσικό αντικείμενο, που χρησιμοποιείται για να παραστήσει ένα στοιχείο δεδομένου. Στη μεταβλητή εκχωρείται μια τιμή, η οποία μπορεί να αλλάζει κατά τη διάρκεια εκτέλεσης του αλγορίθμου. Ανάλογα με το είδος της τιμής που μπορούν να λάβουν, οι μεταβλητές διακρίνονται σε αριθμητικές, αλφαριθμητικές και λογικές.

Προσοχή: Δεν αλλάζει ο τύπος μιας μεταβλητής κατά την εκτέλεση του αλγορίθμου.

Ερώτηση 2.11

Ποιοι είναι οι βασικοί τύποι δεδομένων;

1. **Αριθμητικοί**
   - **Ακέραιος τύπος**
     1, 39, 0, -80
   - **Πραγματικός τύπος**
     3.14, 5.7828, -12. 5, 0. 5
2. **Χαρακτήρες**
   "Κ", "Ελένη", "Τετάρτη" "Γ3"
3. **Λογικός τύπος**
   Αληθής και Ψευδής

Ερώτηση 2.12

Ποιοι είναι οι κανόνες ονοματολογίας μεταβλητών, σταθερών και αλγορίθμων;

Οι κανόνες που πρέπει να ισχύουν για την ονομασία είναι:

- Τα ονόματα πρέπει να περιέχουν μόνο, συνδυασμούς από γράμματα (πεζά ή κεφαλαία), αριθμούς και την κάτω παύλα (_).
- Ενα όνομα πρέπει να περιέχει τουλάχιστον ένα γράμμα και να αρχίζει από γραμμα.
- Δεν μπορούν να χρησιμοποιηθούν ως ονόματα οι δεσμευμένες λέξεις, δηλαδή λέξεις όπως το Διάβασε, Εμφάνισε, κ.α. οι οποίες έχουν συγκεκριμένη λειτουργία στον αλγόριθμο.
- Δεν επιτρέπονται τα κενά.

Ερώτηση 2.13

Δώστε παραδείγματα έγκυρων και μη ονομάτων.

Αποδεκτά ονόματα	**Μη αποδεκτά ονόματα**
Α100	100Α
Ονομα	_Τιμή
Τιμή	Μέση τιμή
Μέση_τιμή	Διάβασε
Μο1α	Τιμή$

Ερώτηση 2.14

Τι ονομάζουμε τελεστές;

Πρόκειται για τα γνωστά σύμβολα που χρησιμοποιούνται στις διάφορες πράξεις. Οι τελεστές διακρίνονται σε αριθμητικούς, συγκριτικούς και λογικούς.

- Οι αριθμητικοί τελεστές είναι: +, -, *, /, ^, div, mod
- Οι συγκριτικοί τελεστές είναι: <, <=, >, >=, =, <>
- Οι λογικοί τελεστές είναι: **όχι** (άρνηση), **και** (σύζευξη), **ή** (διάζευξη)

Ερώτηση 2.15

Ποια είναι η ιεραρχία (προτεραιότητα) των τελεστών;

Οι τελεστές εκτελούνται με την ακόλουθη προτεραιότητα:

- Αριθμητικοί τελεστές
- Συγκριτικοί τελεστές
- Λογικοί τελεστές

Ερώτηση 2.16

Ποια είναι η ιεραρχία των αριθμητικών πράξεων;

Οι αριθμητικές πράξεις εκτελούνται με την ακόλουθη ιεραρχία:

- Ύψωση σε δύναμη
- Διαίρεση, Πολ/σμός, div και mod.
- Πρόσθεση και αφαίρεση

Οι τελεστές που βρίσκονται στο ίδιο επίπεδο ιεραρχίας εκτελούνται με τη σειρά που θα βρεθούν από αριστερά προς τα δεξιά.

Οταν υπάρχουν παρενθέσεις, τότε η ιεραρχία αλλάζει και εκτελούνται πρώτα οι πράξεις μέσα στις παρενθέσεις.

Ερώτηση 2.17

Τι ονομάζουμε εκφράσεις;

Οι εκφράσεις διαμορφώνονται από τους τελεστέους (σταθερές, μεταβλητές και αριθμοί) και από τους τελεστές.

Η διεργασία αποτίμησης μιας έκφρασης συνίσταται στην απόδοση τιμών στις μεταβλητές και στην εκτέλεση των πράξεων.

Η τελική τιμή μιας έκφρασης εξαρτάται από την ιεραρχία των πράξεων και τη χρήση παρενθέσεων. Μια έκφραση μπορεί να αποτελείται από μια μόνο μεταβλητή ή σταθερά μέχρι μια πολύπλοκη μαθηματική παράσταση.

Το αποτέλεσμα του υπολογισμού μιας έκφρασης μπορεί να είναι:

- ένας αριθμός (αριθμητική έκφραση)
- ή μία λογική τιμή (λογική έκφραση ή λογική συνθήκη ή απλά συνθήκη), δηλαδή **αληθής** ή **ψευδής**.

Ερώτηση 2.18

Πότε χρησιμοποιούνται σχόλια και τι ρόλο έχουν;

Μπορούμε κατά τη συγγραφή ενός αλγορίθμου να χρησιμοποιήσουμε επεξηγηματικές εκφράσεις. Οι φράσεις αυτές καλούνται σχόλια και βοηθούν στην κατανόηση του κώδικα. Προκειμένου να διαχωρίζονται οι φράσεις αυτές στην αρχή προτάσσεται το σύμβολο !, για παράδειγμα !Σχόλια.

Ερώτηση 2.19

Ποιες είναι οι εντολές εισόδου;

Για την εισαγωγή δεδομένων από το πληκτρολόγιο χρησιμοποιείται η εντολή είσόδου **Διάβασε**. Η εκτέλεση της εντολής οδηγεί στην είσοδο τιμών από το πληκτρολόγιο (δηλαδή από τον χρήστη) και την εκχώρισή του στις μεταβλητές που αναφέρονται.

Η εντολή **Διάβασε** ακολουθείται πάντοτε από ένα ή περισσότερα ονόματα μεταβλητών. Αν υπάρχουν περισσότερες από μια μεταβλητές τότε αυτές χωρίζονται με κόμμα.

Διάβασε μεταβλητή1, μεταβλητή2, …

Πχ. Διάβασε α, β

Κατά την εκτέλεση του προγράμματος η εντολή Διάβασε διακόπτει την εκτέλεση 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.

Τα δεδομένα εισόδου (αν υπάρχουν) περιγράφονται στη δεύτερη γραμμή του αλγορίθμου εντός των συμβόλων//... //.

Χρησιμοποιούνται όταν η εκφώνηση λέει: «… δίνονται οι αριθμοί α και β» αντί του «… να διαβάζει δύο αριθμούς α και β».

Πχ. Δεδομένα //α, β θετικοί ακέραιοι//

Ερώτηση 2.20

Ποιες είναι οι εντολές εξόδου;

Η εντολή **Εμφάνισε** έχει ως αποτέλεσμα την εμφάνιση τιμών στην οθόνη. Εναλλακτικά, υπάρχει η εντολή **Εκτύπωσε** η οποία εκτυπώνει τα αποτελέσματα στον εκτυπωτή.

Η σύνταξή της είναι:

Εμφάνισε λίστα_στοιχείων

Η λίστα των στοιχείων μπορεί να περιέχει σταθερές τιμές και ονόματα μεταβλητών.

Τα αποτελέσματα εξόδου δίνονται στην προτελευταία γραμμή του αλγορίθμου εντός των συμβόλων // ... //. Αυτό γίνεται όταν ο αλγόριθμος δεν ζητάει ξεκάθαρα την εμφάνιση τιμών.

πχ. Αποτελέσματα //α, β//

Ερώτηση 2.21

Ποια είναι η δομή ενός αλγορίθμου;

Αλγόριθμος Ονομα
Εντολή 1
Εντολή 2
……
Εντολή ν
Τέλος Ονομα

Ερώτηση 2.22

Ποια η λειτουργία της εντολής εκχώρησης;

Η εντολή εκχώρησης εκχωρεί σε μια μεταβλητή την τιμή μιας άλλης ή το αποτέλεσμα του υπολογισμού μιας παράστασης. Συμβολίζεται με το← και η γενική της μορφή είναι:

Μεταβλητή ← Εκφραση

Στην εντολή αυτή χρησιμοποιείται το αριστερό βέλος, προκειμένου να δείχνει τη φορά της εκχώρησης. Ας σημειωθεί ότι δεν πρόκειται για εξίσωση. Επίσης οι διάφορες γλώσσες προγραμματισμού χρησιμοποιούν διάφορα σύμβολα για το σκοπό αυτό.

Για παράδειγμα:
Α← 1453
Μήνας← “Ιούνιος”
Εμβαδόν← Α*Β

Παρατηρήσεις

- Υπολογίζεται η τιμή της έκφρασης δεξιά απ’ το βέλος και εκχωρείται αυτή τη τιμή στη μεταβλητή αριστερά απ’ το βέλος.
- Η εκχώρηση δεν είναι εξίσωση.
- Η μεταβλητή και η έκφραση πρέπει να είναι του ιδίου τύπου.

2.3 Δομή ακολουθίας

Ερώτηση 2.23

Τι ονομάζουμε συνιστώσες ενός αλγορίθμου;

Είναι οι απαραίτητες εντολές ξεκινώντας από τις απλούστερες και προχωρώντας προς τις συνθετότερες.

Πιο συγκεκριμένα θα εξετασθούν περιπτώσεις σειριακών εντολών, αναθέσεων τιμών, επιλογής με βάση κριτήρια, διαδικασιών επανάληψης, ενεργειών πολλαπλών επιλογών καθώς και συνδυασμό εμφωλευμένων περιπτώσεων.

Για κάθε περίπτωση παρουσιάζονται σχετικά παραδείγματα με την εκφώνηση (σε φυσική γλώσσα), την παρουσίαση των βημάτων που πρέπει να ακολουθηθούν καθώς και το αντίστοιχο διάγραμμα ροής.

Ερώτηση 2.24

Σε ποιές βασικές κατηγορίες χωρίζουμε τις αλγοριθμικές συνιστώσες;

Οι βασικές κατηγορίες αλγοριθμικών συνιστώσων είναι οι εξής:

- Δομή ακολουθίας
- Δομή επιλογής
- Δομή επανάληψης

Ερώτηση 2.25

Τι ονομάζουμε δομή ακολουθίας;

Η δομή ακολουθίας αποτελείται από ενέργειες (εντολές) που θα εκτελεστούν μία φορά με προκαθορισμένη σειρά.

Ερώτηση 2.26

Πότε χρησιμοποιείται η δομή ακολουθίας;

Η ακολουθιακή δομή εντολών (σειριακών βημάτων) χρησιμοποιείται πρακτικά για την αντιμετώπιση απλών προβλημάτων, όπου είναι δεδομένη η σειρά εκτέλεσης ενός συνόλου ενεργειών.

Τα βήματα και οι ποσότητες που πρέπει να ακολουθηθούν είναι συγκεκριμένα και οι οδηγίες απόλυτα καθορισμένες και σαφείς.

2.4 Δομή επιλογής

Ερώτηση 2.27

Πότε χρησιμοποιούμε Δομή Επιλογής;

Στην πραγματικότητα πολύ λίγα προβλήματα μπορούν να επιλυθούν με τον τρόπο της σειριακής/ ακολουθιακής δομής ενεργειών.

Η πλέον συνηθισμένη περίπτωση είναι να λαμβάνονται κάποιες αποφάσεις με βάση κάποια δεδομένα κριτήρια (συνθήκες), που μπορεί να είναι διαφορετικό για κάθε διαφορετικό στιγμιότυπο (τιμές της συνθήκης) ενός προβλήματος.

Ερώτηση 2.28

Ποιά είναι η γενική μορφή της δομής επιλογής;

Γενικά, η διαδικασία της επιλογής περιλαμβάνει τον έλεγχο κάποιας συνθήκης που μπορεί να έχει δύο τιμές (Αληθής ή Ψευδής) και ακολουθεί η απόφαση εκτέλεσης κάποιας ενέργειας με βάση την τιμή της λογικής αυτής συνθήκης.

Ερώτηση 2.29

Τι ονομάζουμε λογική έκφραση;

Δομικό στοιχείο στις δομές επιλογής είναι οι λογικές εκφράσεις, δηλαδή εκφράσεις που το αποτέλεσμα τους είναι μία λογική τιμή (λογική έκφραση ή λογική συνθήκη ή απλά συνθήκη), δηλαδή αληθής ή ψευδής.

Για τη σύνταξη μιας λογικής έκφρασης ή συνθήκης χρησιμοποιούνται σταθερές, μεταβλητές, αριθμητικές παραστάσεις, συγκριτικοί και λογικοί τελεστές, καθώς και παρενθέσεις.

Ερώτηση 2.30

Τι παρατηρήσεις θα μπορούσαμε να κάνουμε για την σύγκριση των δεδομένων;

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

Ερώτηση 2.31

Ποιός είναι ο πίνακας των λογικών πράξεων;

Πρόταση Α	**Πρόταση Β**	Α **ή** Β	Α **και** Β	**όχι** (Α)	**όχι** (Β)
ΑΛΗΘΗΣ	ΑΛΗΘΗΣ	ΑΛΗΘΗΣ	ΑΛΗΘΗΣ	ΨΕΥΔΗΣ	ΨΕΥΔΗΣ
ΑΛΗΘΗΣ	ΨΕΥΔΗΣ	ΑΛΗΘΗΣ	ΨΕΥΔΗΣ	ΨΕΥΔΗΣ	ΑΛΗΘΗΣ
ΨΕΥΔΗΣ	ΑΛΗΘΗΣ	ΑΛΗΘΗΣ	ΨΕΥΔΗΣ	ΑΛΗΘΗΣ	ΨΕΥΔΗΣ
ΨΕΥΔΗΣ	ΨΕΥΔΗΣ	ΨΕΥΔΗΣ	ΨΕΥΔΗΣ	ΑΛΗΘΗΣ	ΑΛΗΘΗΣ

Ερώτηση 2.32

Ποιες είναι οι μορφές της δομής επιλογής;

Θα μπορούσαμε να χωρίσουμε τις δομές επιλογής στις εξής κατηγορίες:

- Δομής απλής επιλογής
- Δομή σύνθετης επιλογής
- Δομή πολλαπλής επιλογής

Ερώτηση 2.33

Πως συντάσσεται η δομή απλής επιλογής;

Αν συνθήκη **τότε**
    Εντολή ή εντολές
Τέλος_αν
Εναλλακτικά, θα μπορούσαμε να χρησιμοποιήσουμε την παρακάτω δομή:

Αν συνθήκη **τότε** εντολή

Αυτό συμβαίνει αν θα εκτελεστεί μόνο μια εντολή στην περίπτωση που ισχύει η συνθήκη. Τότε μπορούμε να γράψουμε την εντολή δίπλα στο τότε και να μην γράψουμε το **Τέλος_αν.** Σε κάθε περίπτωση εκτελείται στη συνέχεια η εντολή, που ακολουθεί.

Ερώτηση 2.34

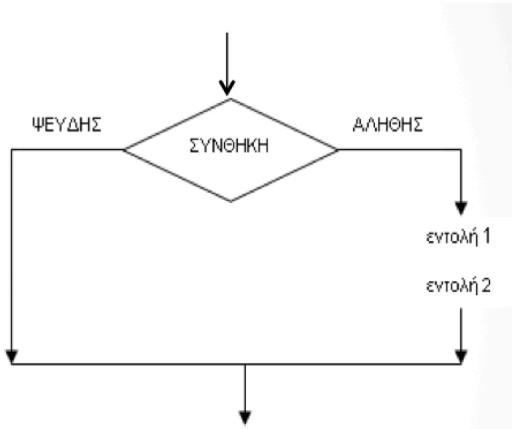
Πως λειτουργεί η δομή απλής επιλογής;

Η δομή της απλής επιλογής περιλαμβάνει μια συνθήκη και ένα σύνολο ενεργειών (εντολών).

- Αν η τιμή της συνθήκης είναι ΑΛΗΘΗΣ εκτελείται το σύνολο των ενεργειών (εντολών) που βρίσκεται μεταξύ των λέξεων Αν και Τέλος_αν.
- Διαφορετικά δεν εκτελείται τίποτα και συνεχίζει η εκτέλεση του αλγορίθμου με την εντολή που βρίσκεται μετά το Τέλος_αν.

Ερώτηση 2.35

Ποιο είναι το διάγραμμα ροής της δομής απλής επιλογής;



Ερώτηση 2.36

Πως συντάσσεται η δομή σύνθετης επιλογής;

Αν συνθήκη **τότε**
    Εντολή ή εντολές
Αλλιώς
    Εντολή ή εντολές
Τέλος_αν

Ερώτηση 2.37

Πως λειτουργεί η δομή σύνθετης επιλογής;

Η δομή της σύνθετης επιλογής περιλαμβάνει μια συνθήκη και 2 σύνολα ενεργειών (εντολών).

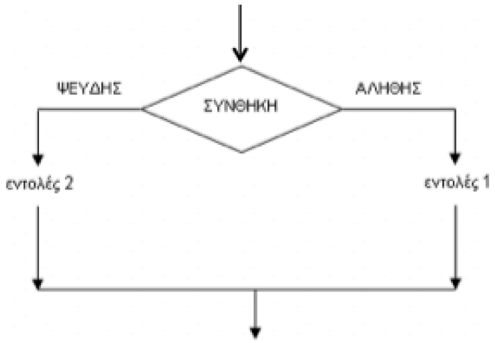
- Αν η τιμή της συνθήκης είναι ΑΛΗΘΗΣ εκτελείται το πρώτο σύνολο εντολών που βρίσκεται μεταξύ των λέξεων **Αν** και **Αλλιώς.**

- Αν η συνθήκη είναι ΨΕΥΔΗΣ εκτελείται το δεύτερο σύνολο εντολών που βρίσκεται μεταξύ των λέξεων **Αλλιώς** και **Τέλος_αν.**

Προσοχή: Θα εκτελεστεί μόνο το ένα σύνολο εντολών, ποτέ και τα δύο.

Ερώτηση 2.38

Ποιό είναι το διάγραμμα ροής της σύνθετης επιλογής;



Ερώτηση 2.39

Πως συντάσσεται η δομή πολλαπλής επιλογής **Αν...Αλλιώς_Αν**;

Οι δομές πολλαπλής επιλογής αποτελούνται από τις εξής εντολές:
Αν συνθήκη1 **τότε**
Εντολή Α ή εντολές Α
Αλλιώς_αν συνθήκη2 **τότε**
Εντολή Β ή εντολές Β
……
Αλλιώς
Εντολή Ν ή εντολές Ν
Τέλος_ Αν

Ερώτηση 2.40

Πως λειτουργεί η δομή πολλαπλής επιλογής **Αν...Αλλιώς_Αν**;

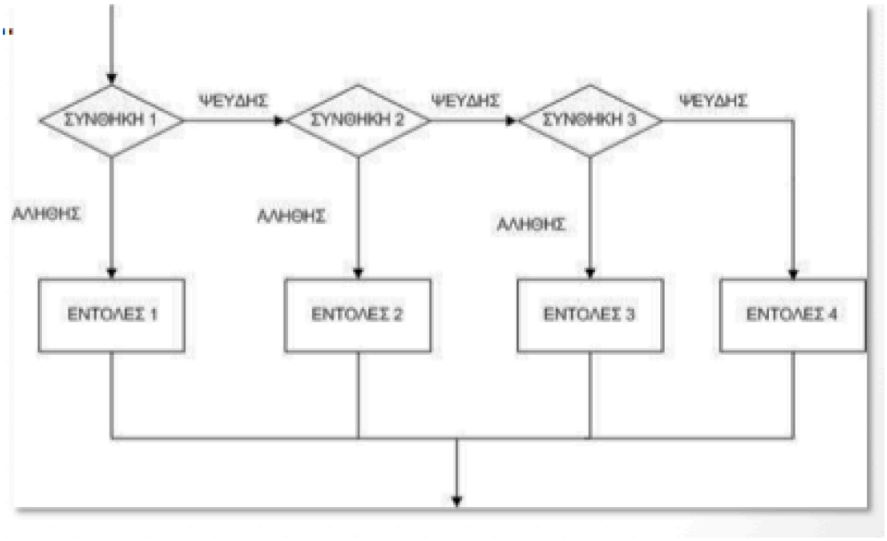
Η δομή πολλαπλής επιλογής περιλαμβάνει μια έκφραση και πολλά διακριτά σύνολα ενεργειών (εντολών).

- Αρχικά ελέγχεται η πρώτη συνθήκη.
- Αν είναι **ΑΛΗΘΗΣ** τότε εκτελείται η ομάδα *Εντολή Α ή εντολές Α*
- Αν είναι **ΨΕΥΔΗΣ** τότε ελέγχεται η δεύτερη συνθήκη κ.ο.κ.
- Αν όλες οι συνθήκες έχουν την τιμή **ΨΕΥΔΗΣ** τότε εκτελείται η *Εντολή Ν ή εντολές Ν*

Η χρήση του τελευταίου αλλιώς είναι προαιρετική.
Σε κάθε περίπτωση ο αλγόριθμος συνεχίζει με την επόμενη εντολή μετά το Τέλος_Αν.

Ερώτηση 2.41

Ποιο είναι το διάγραμμα ροής της πολλαπλής επιλογής **Αν...Αλλιώς_Αν**;



Ερώτηση 2.42

Πως συντάσσεται η δομή πολλαπλής επιλογής **Επίλεξε...Τέλος_Επιλογών**;

Επίλεξε έκφραση
Περίπτωση Λίστα Τιμών 1
Εντολή Α ή εντολές Α
Περίπτωση Λίστα Τιμών 2
Εντολή Β ή εντολές Β
.....

Περίπτωση Αλλιώς
Εντολή Ν ή εντολές Ν
Τέλος_ επιλογών

Ερώτηση 2.43

Πως λειτουργεί η δομή πολλαπλής επιλογής **Επίλεξε...Τέλος_Επιλογών**;

Υπολογίζεται η τιμή της εκφρασης και ανάλογα με την τιμή της έκφρασης εκτελούνται οι εντολές μετά την **Περίπτωση** που αντιστοιχεί στην τιμή της έκφρασης.

Οι τιμές που συνοδεύουν κάθε Περίπτωση μπορεί να είναι:

- μία ή περισσότερες διακριτές τιμές,
- περιοχή τιμών από - έως ή

- να υπακούουν σε μια συνθήκη

Αν η τιμή της έκφρασης δεν αντιστοιχεί σε καμία **Περίπτωση**, τότε εκτελούνται οι εντολές που ακολουθούν την **Περίπτωση Αλλιώς.**

Μετά την εκτέλεση μιας περίπτωσης, ο αλγόριθμος συνεχίζει με την εντολή, που ακολουθεί το **Τελος_Επιλογων.**

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

Ερώτηση 2.44

Τι ονομάζουμε εμφωλευμένες δομές επιλογής;

Η εμφωλευμένη επιλογή είναι συνδυασμός των δομών επιλογής. Ουσιαστικά πρόκειται για μια απλή, σύνθετη ή πολλαπλή επιλογή που περιέχει εσωτερικά, στην ομάδα εντολών της, μια ή περισσότερες άλλες επιλογές.

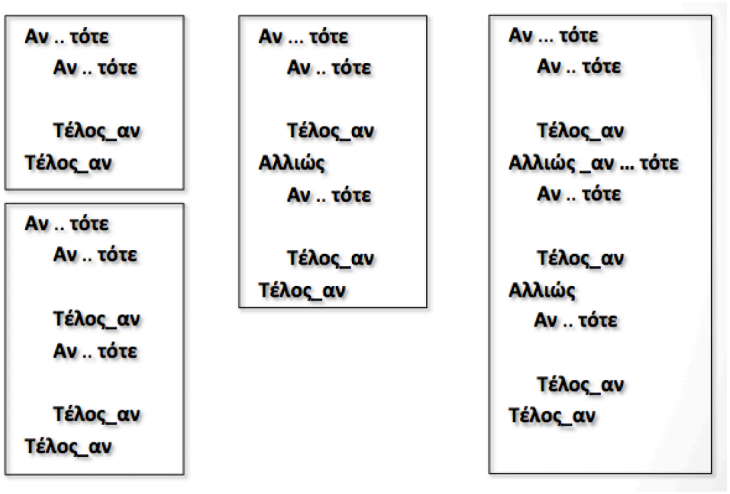
Η εμφωλευμένη επιλογή χρησιμεύει όταν έχουμε να εκτελέσουμε διαφορετικές διαδικασίες για τις διάφορες τιμές δύο ή περισσότερων μεταβλητών.

Οταν έχουμε να εκτελέσουμε διαφορετικές διαδικασίες για τις διάφορες τιμές μιας μεταβλητής είναι προτιμότερη η πολλαπλή επιλογή, η οποία απλοποιεί τον αλγόριθμο.

Στο σχολικό βιβλίο προτείνεται να αποφεύγεται η εμφωλευμένη επιλογή γιατί δεν διευκολύνει στην κατανόηση του αλγορίθμου.

Ερώτηση 2.45

Παραδείγματα εμφωλευμένων επιλογων



2.5 Δομή επανάληψης

Ερώτηση 2.46

Πότε χρησιμοποιείται η δομή επανάληψης;

Η λογική των επαναληπτικών διαδικασιών εφαρμόζεται στις περιπτώσεις, όπου μια ακολουθία εντολών πρέπει να εφαρμοσθεί σε ένα σύνολο περιπτώσεων, που έχουν κάτι κοινό.

Οι επαναληπτικές διαδικασίες μπορεί να έχουν διάφορες μορφές και συνήθως εμπεριέχουν και συνθήκες επιλογών.

Ερώτηση 2.47

Ποιες είναι οι βασικές μορφές των δομών επανάληψης;

Οι βασικές μορφές των δομών επανάληψης είναι:

- Οσο ... Επανάλαβε
- Αρχή_Επανάληψης ... Μέχρις_Οτου
- Για.. από... μέχρι

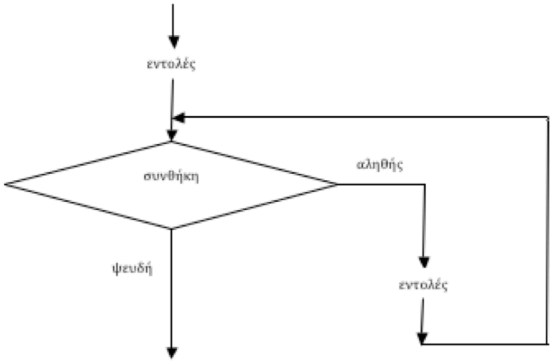
Ερώτηση 2.48

Πως συντάσσεται η δομή επανάληψης Οσο ... Επανάλαβε.

Η σύνταξη της εντολής είναι η εξής: **Οσο** συνθήκη **επανάλαβε**
εντολές
Τέλος_επανάληψης

Ερώτηση 2.49

Ποιο είναι το διάγραμμα ροής της δομής επανάληψης Αρχή_ επανάληψης...Μέχρις_ ότου;



Ερώτηση 2.50

Περιγράψτε τη δομή επανάληψης Οσο ... Επανάλαβε

Η δομή επανάληψης **Οσο...Επανάλαβε** αποτελεί τον πιο γενικό τύπο δομής επανάληψης. Η ομάδα εντολών στο εσωτερικό της επανάληψης εκτελείται όσο ισχύει η συνθήκη που υπάρχει στην αρχή της επανάληψης:

Αρχικά γίνεται έλεγχος της συνθήκης. Αν η συνθήκη ισχύει εκτελούμε τις εντολές που υπάρχουν μέσα στο Οσο…Επανάλαβε. Αν η συνθήκη δεν ισχύει, τότε η εκτέλεση του αλγορίθμου συνεχίζει μετά το Τέλος_επανάληψης.

Οταν εκτελούνται οι εντολές που υπάρχουν ανάμεσα στο Οσο…Επανάλαβε και στο Τέλος_επανάληψης επανερχόμαστε στη συνθήκη και ελέγχουμε εκ νέου αν ισχύει.

Ερώτηση 2.51

Τι παρατηρήσεις θα μπορούσαμε να κάνουμε για την δομή επανάληψης Οσο ... Επανάλαβε;

- Χρησιμοποιείται για γνωστό και για άγνωστο πλήθος επαναλήψεων.
- Ο έλεγχος της συνθήκης γίνεται στην αρχή.
- Η επανάληψη γίνεται όσο η συνθήκη είναι «αληθής».
- Υπάρχει περίπτωση να μην εκτελεστεί καμία επανάληψη (δηλ. να μην εκτελεστούν οι εντολές).
- Δεν επεξεργάζεται την τιμή «φρουρό»

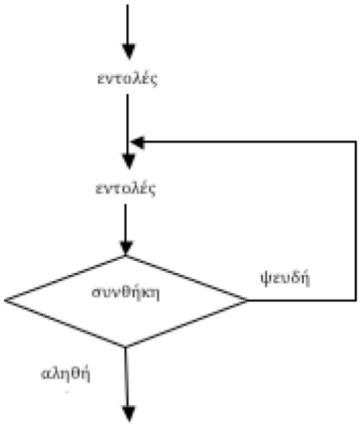
Ερώτηση 2.52

Πως συντάσσεται η δομή επανάληψης Αρχή_επανάληψης... Μέχρις_ότου

Αρχή_επανάληψης
Εντολές
Μέχρις_ότου συνθήκη

Ερώτηση 2.53

Ποιο είναι το διάγραμμα ροής της δομής επανάληψης Αρχή_ επανάληψης...Μέχρις_ ότου;



Ερώτηση 2.54

Περιγράψτε τη δομή επανάληψης Αρχή_επανάληψης ... Μέχρις_ότου

Η ομάδα εντολών στο εσωτερικό της επανάληψης εκτελείται μέχρις ότου η συνθήκη να γίνει αληθής, δηλαδή όσο η συνθήκη είναι ψευδής.

Αρχικά εκτελούνται οι εντολές και πηγαίνουμε στο Μέχρις_ότου.

Αν η συνθήκη δεν ισχύει επανερχόμαστε στο Αρχή_επανάληψης και εκτελούμε ξανά τις εντολές και ελέγχουμε εκ νέου τη συνθήκη. Αν η συνθήκη ισχύει τότε τερματίζει η επανάληψη.

Ερώτηση 2.55

Τι παρατηρήσεις θα μπορούσαμε να κάνουμε για την δομή επανάληψης Αρχή_επανάληψης... Μέχρις_ότου;

- Χρησιμοποιείται για γνωστό και για άγνωστο πλήθος επαναλήψεων.
- Ο έλεγχος της συνθήκης γίνεται στο τέλος.
- Η επανάληψη γίνεται όσο η συνθήκη είναι «ψευδής».
- Θα εκτελεστεί τουλάχιστον μία επανάληψη.
- Επεξεργάζεται την τιμή «φρουρό», αν δεν γίνει κάποιο τέχνασμα.

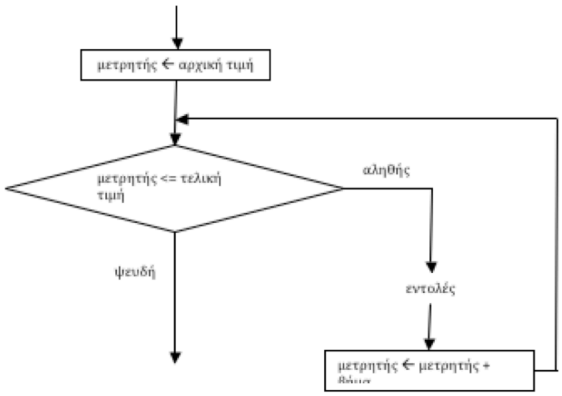
Ερώτηση 2.56

Πως συντάσσεται η δομή επανάληψης Για.. από ...μέχρι

Για μεταβλητή **από** αρχ.τιμή **μέχρι** τελ. τιμή **με_βήμα** βήμα μεταβολής
εντολές
Τέλος_επανάληψης

Ερώτηση 2.57

Ποιο είναι το διάγραμμα ροής της δομής επανάληψης Για.. από ...μέχρι



Ερώτηση 2.58

Περιγράψτε τη δομή επανάληψης Για.. από ...μέχρι

Η δομή επανάληψης **Για...από...μέχρι** εκτελείται για προκαθορισμένο αριθμό βημάτων. Στη θέση της μεταβλητής που ονομάζεται μετρητής, μπαίνει το όνομα μιας οποιαδήποτε μεταβλητής, πχ. i, j, x, y. Ο τρόπος λειτουργίας της εντολής Για…από…μέχρι είναι ο εξής:

- Αρχίζει η επανάληψη και η μεταβλητή παίρνει την αρχ. τιμή.
- Ελέγχεται η συνθήκη *μεταβλητή* $\leq$ *τελική_τιμή*, αν το βήμα είναι θετικό ή η συνθήκη *μεταβλητή* $\geq$ *τελική_τιμή*, αν το βήμα είναι θετικό.
- Αν η συνθήκη είναι αληθής, εκτελούνται οι εντολές που υπάρχουν μέσα στο Για…από…μέχρι και η μεταβλητή αυξάνεται ή μειώνεται ανάλογα με το βήμα μεταβολής.
- Επαναλαμβάνονται τα προηγούμενα βήματα μέχρι η συνθήκη να γίνει ψευδής.

Ερώτηση 2.59

Τι παρατηρήσεις θα μπορούσαμε να κάνουμε για την δομή επανάληψης Για.. από ...μέχρι;

- Χρησιμοποιείται μόνο για γνωστό πλήθος επαναλήψεων.
- Αν το βήμα μεταβολής είναι 1 τότε μπορεί να παραληφθεί.
- Αν το βήμα μεταβολής είναι θετικό η συνθήκη γίνεται *μεταβλητή*$\leq$ *τελική_τιμή.*
- Αν το βήμα μεταβολής είναι αρνητικό η συνθήκη γίνεται *μεταβλητή*$\geq$ *τελική_τιμή.*
- Το βήμα μεταβολής δεν πρέπει να είναι μηδέν γιατί υπάρχει κίνδυνος μη τερματισμού του αλγορίθμου.

- Η αρχική τιμή, η τελική τιμή και το βήμα μεταβολής μπορεί να είναι αρνητικοί αριθμοί.
- Η αρχική και η τελική τιμή μπορεί να είναι μηδέν.
- Μπορούμε να χρησιμοποιήσουμε μεταβλητές για αρχική τιμή, για τελική τιμή και για βήμα μεταβολής.

Ερώτηση 2.60

Να γραφεί ο αλγόριθμος του «πολλαπλασιασμού αλά ρωσικά» σε μορφή ψευδοκώδικα. Να εξηγήσετε τον τρόπο λειτουργίας του.

Αλγόριθμος Πολλαπλασιασμός_ αλά_ ρωσικά **Δεδομένα** // M1,M2 ακέραιοι //
P ← 0
Οσο M2 > 0 **επανάλαβε**
Αν M2 mod 2 = 1 **τότε** P ←P+M1
M1← M1*2
M2←M2 div 2 ![M2/2]
Τέλος_ επανάληψης
Αποτελέσματα //P το γινόμενο των M1,M2 // Τέλος Πολλαπλασιασμός_ αλα_ ρωσικά

Ερώτηση 2.61

Να γραφεί ο αλγόριθμος του «πολλαπλασιασμού αλά ρωσικά» με φυσική γλώσσα κατά βήματα.

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

Ερώτηση 2.62

Να εξηγήσετε τον τρόπο λειτουργίας του «πολλαπλασιασμού αλά ρωσικά»

Η πράξη του πολλαπλασιασμού δεν εκτελείται από τον υπολογιστή με τον τρόπο αυτό. Πιο συγκεκριμένα, ο χρησιμοποιούμενος τρόπος είναι ο λεγόμενος πολλαπλασιασμός αλά ρωσικά. Χωρίς βλάβη της γενικότητας θεωρούμε ότι οι ακέραιοι είναι θετικοί (μεγαλύτεροι του μηδενός), αλλά η μέθοδος μπορεί εύκολα να μετατραπεί, ώστε να περιγράψει και την περίπτωση των αρνητικών ακεραίων. Πως ακριβώς λειτουργεί η μέθοδος, θα φανεί με το επόμενο παράδειγμα, όπου περιγράφεται ο αλγόριθμος με ελεύθερο κείμενο.

Εστω, λοιπόν, ότι δίνονται δύο θετικοί ακέραιοι αριθμοί, οι αριθμοί 45 και 19. Οι αριθμοί γράφονται δίπλα-δίπλα και ο πρώτος διπλασιάζεται αγνοώντας το δεκαδικό μέρος, ενώ ο δεύτερος υποδιπλασιάζεται. Στο παρακάτω σχήμα παρουσιάζεται η επαναλαμβανόμενη διαδικασία, που συνεχίζεται μέχρις ότου στη δεύτερη στήλη να προκύψει μονάδα. Τελικώς, το γινόμενο ισούται με το άθροισμα των στοιχείων της πρώτης στήλης, όπου αντίστοιχα στη δεύτερη στήλη υπάρχει περιττός αριθμός. Για το παράδειγμα μας, τα στοιχεία αυτά παρουσιάζονται στην τρίτη στήλη.

M1	M2	P
45	19	45
90	9	90
180	4	
360	2	
720	1	720
Αθροισμα		855

Ερώτηση 2.63

Ποια η πρακτική σημασία του αλγορίθμου «πολλαπλασιασμού αλά ρώσικά»; Πότε γίνεται χρήση αυτού του τρόπου πολλαπλασιασμού δύο ακέραιων;

Η μέθοδος αυτή χρησιμοποιείται πρακτικά στους υπολογιστές, γιατί υλοποιείται πολύ πιο απλά απ' ότι ο γνωστός μας χειρωνακτικός τρόπος πολλαπλασιασμού. Πιο συγκεκριμένα, απαιτεί πολλαπλασιασμό επί δύο, διαίρεση δια δύο και πρόσθεση.

Σε αντίθεση η γνωστή μας διαδικασία πολλαπλασιασμού απαιτεί πολλαπλασιασμό με οποιοδήποτε ακέραιο και πρόσθεση.

Σε επίπεδο, λοιπόν, κυκλωμάτων υπολογιστή ο πολλαπλασιασμός επί δύο και η διαίρεση δια δύο μπορούν να υλοποιηθούν ταχύτατα με μία απλή εντολή ολίσθησης (shift), σε αντίθεση με τον πολλαπλασιασμό με οποιοδήποτε ακέραιο που θεωρείται πιο χρονοβόρα διαδικασία.

Το τελευταίο γεγονός είναι ο λόγος που ο πολλαπλασιασμός αλά ρωσικά είναι προτιμότερος απ' ότι ο χειρωνακτικός τρόπος πολλαπλασιασμού δύο ακεραίων.

Ερώτηση 2.64

Τι ονομάζουμε ολίσθηση προς τα αριστερά και τι ολίσθηση προς τα δεξιά;

Στα κυκλώματα του υπολογιστή τα δεδομένα αποθηκεύονται με δυαδική μορφή, δηλαδή 0 και 1, ανεξάρτητα από το πως τα ορίζει ο προγραμματιστής, όπως ακεραίους ή πραγματικούς σε δεκαδικό σύστημα, ή ακόμη χαρακτήρες κ.λπ. Ετσι ο αριθμός 17 του δεκαδικού συστήματος ισοδυναμεί με τον αριθμό 00010001 του δυαδικού συστήματος, ο οποίος μπορεί να αποθηκευθεί σε ένα byte.

0	0	0	1	0	0	0	0	1

Αν μετακινήσουμε τα ψηφία αυτά κατά μία θέση προς τα αριστερά, δηλαδή αν προσθέσουμε ένα 0 στο τέλος του αριθμού και αγνοήσουμε το αρχικό 0, τότε προκύπτει ο αριθμός 00100010 του δυαδικού συστήματος, που ισοδυναμεί με το αριθμό 34 του δεκαδικού συστήματος.

0	0	1	0	0	0	0	1	0

Επίσης, με παρόμοιο τρόπο, αν μετακινήσουμε τα ψηφία κατά μία θέση δεξιά, δηλαδή αποκόψουμε το τελευταίο ψηφίο 1 και θεωρήσουμε ένα ακόμη αρχικό 0, τότε προκύπτει ο αριθμός 00001000 του δυαδικού συστήματος, που ισοδυναμεί με τον αριθμό 8 του δεκαδικού συστήματος.

0	0	0	0	1	0	0	0	0

Αρα, η ολίσθηση προς τα αριστερά ισοδυναμεί με πολλαπλασιασμό επί δύο, ενώ η ολίσθηση προς τα δεξιά ισοδυναμεί με την ακέραια διαίρεση δια δύο.

Κεφάλαιο 3

Δομές Δεδομένων και Αλγόριθμοι

3.1 Δεδομένα

Ερώτηση 3.1

Τι ορίζουμε ως δεδομένα υπό το πρίσμα του πεδίου της Πληροφορικής; Πως συνδέονται με τις πληροφορίες;

Τα δεδομένα (data) είναι η αφαιρετική αναπαράσταση της πραγματικότητας και συνεπώς μία απλοποιημένη όψη της.

Για παράδειγμα, έστω ένα αρχείο μαθητών ενός σχολείου. Τα χρήσιμα δεδομένα που αποθηκεύονται είναι το ονοματεπώνυμο, η ηλικία, το φύλο, η τάξη, το τμήμα κλπ., όχι όμως το βάρος, το ύψος κλπ.

Τα δεδομένα, λοιπόν, είναι ακατέργαστα γεγονότα, και κάθε φορά η επιλογή τους εξαρτάται από τον τύπο του προβλήματος. Η συλλογή των ακατέργαστων δεδομένων και ο συσχετισμός τους δίνει ως αποτέλεσμα την πληροφορία (information).

Δεν είναι εύκολο να δοθεί επακριβής ορισμός της έννοιας της πληροφορίας, αλλά μπορεί να θεωρηθεί ότι ο αλγόριθμος είναι το μέσο για την παραγωγή πληροφορίας από τα δεδομένα. Με βάση τις δεδομένες πληροφορίες λαμβάνονται διάφορες αποφάσεις και γίνονται ενέργειες. Στη συνέχεια αυτές οι ενέργειες παράγουν νέα δεδομένα, νέες πληροφορίες, νέες αποφάσεις, νέες ενέργειες κοκ. Η μέτρηση, η κωδικοποίηση, η μετάδοση της πληροφορίας αποτελεί αντικείμενο μελέτης ενός ιδιαίτερου κλάδου, της Θεωρίας Πληροφοριών (Information Theory), που είναι ένα ιδιαίτερα σημαντικό πεδίο της Πληροφορικής.

Ερώτηση 3.2

Από ποιες σκοπιές μελετά τα δεδομένα η Πληροφορική;

Οπως η Πληροφορική ορίζεται ως επιστήμη σε συνάρτηση με την έννοια του αλγορίθμου, κατά τον ίδιο τρόπο η Πληροφορική ορίζεται και σε σχέση με την έννοια των δεδομένων. Ετσι, Πληροφορική θεωρείται η επιστήμη που μελετά τα δεδομένα από τις ακόλουθες σκοπιές:

- **Υλικού**. Το υλικό (hardware), δηλαδή η μηχανή, επιτρέπει στα δεδομένα ενός προγράμματος να αποθηκεύονται στην κύρια μνήμη και στις περιφερειακές συσκευές του υπολογιστή με διάφορες αναπαραστάσεις (representations). Τέτοιες μορφές είναι η δυαδική, ο κώδικας ASCII (βλ. παράρτημα), ο κώδικας EBCDIC, το συμπλήρωμα του 1 ή του 2 κ.λπ.

- **Γλωσσών προγραμματισμού.** Οι γλώσσες προγραμματισμού υψηλού επιπέδου (high level programming languages) επιτρέπουν τη χρήση διάφορων τύπων (types) μεταβλητών (variables) για να περιγράψουν ένα δεδομένο. Ο μεταφραστής κάθε γλώσσας φροντίζει για την αποδοτικότερη μορφή αποθήκευσης, από πλευράς υλικού, κάθε μεταβλητής στον υπολογιστή.
- **Δομών Δεδομένων.** Δομή δεδομένων (data structure) είναι ένα σύνολο δεδομένων μαζί με ένα σύνολο επιτρεπτών λειτουργιών επί αυτών. Για παράδειγμα, μία τέτοια δομή είναι η εγγραφή (record), που μπορεί να περιγράφει ένα είδος, ένα πρόσωπο κλπ. Η εγγραφή αποτελείται από τα πεδία (fields) που αποθηκεύουν χαρακτηριστικά (attributes) διαφορετικού τύπου, όπως για παράδειγμα ο κωδικός, η περιγραφή κλπ. Αλλη μορφή δομής δεδομένων είναι το αρχείο που αποτελείται από ένα σύνολο εγγραφών. Μία επιτρεπτή λειτουργία σε ένα αρχείο είναι η σειριακή προσπέλαση όλων των εγγραφών του.
- **Ανάλυσης Δεδομένων.** Τρόποι καταγραφής και αλληλοσυσχέτισης των δεδομένων μελετώνται έτσι ώστε να αναπαρασταθεί η γνώση για πραγματικά γεγονότα. Οι τεχνολογίες των Βάσεων Δεδομένων (Databases), της Μοντελοποίησης Δεδομένων (Data Modeling) και της Αναπαράστασης Γνώσης (Knowledge Representation) ανήκουν σε αυτή τη σκοπιά μελέτης των δεδομένων.

Ερώτηση 3.3

Τι γνωρίζετε για τις δομές δεδομένων δευτερεύουσας μνήμης;

- Σε μεγάλες πρακτικές εμπορικές/επιστημονικές εφαρμογές, το μέγεθος της κύριας μνήμης δεν επαρκεί για την αποθήκευση των δεδομένων. Στην περίπτωση αυτή χρησιμοποιούνται ειδικές δομές για την αποθήκευση των δεδομένων στη δευτερεύουσα μνήμη, δηλαδή κυρίως στο μαγνητικό δίσκο. Οι ειδικές αυτές δομές ονομάζονται αρχεία (files).
- Τα στοιχεία ενός αρχείου ονομάζονται εγγραφές (records), όπου κάθε εγγραφή αποτελείται από ένα ή περισσότερα πεδία (fields), που ταυτοποιούν την εγγραφή, και από άλλα πεδία που περιγράφουν διάφορα χαρακτηριστικά της εγγραφής. Για παράδειγμα, έστω η εγγραφή ενός μαθητή με πεδία: Αριθμός Μητρώου, Ονοματεπώνυμο, Ετος Γέννησης, Τάξη, Τμήμα.
- Το πεδίο Αριθμός Μητρώου ταυτοποιεί την εγγραφή και ονομάζεται πρωτεύον κλειδί (primary key) ή απλά κλειδί. Το πεδίο Ονοματεπώνυμο επίσης ταυτοποιεί την εγγραφή και γι' αυτό αποκαλείται δευτερεύον κλειδί (secondary keys), αν υπάρχει πρωτεύον κλειδί.
- Το πρόβλημα της αναζήτησης (searching) μίας εγγραφής με βάση την τιμή του πρωτεύοντος ή ενός δευτερεύοντος κλειδιού σε αρχεία είναι ιδιαίτερα ενδιαφέρον, αν ληφθεί υπ' όψη η μεγάλη ποικιλία των χαρακτηριστικών τόσο της δομής (για παράδειγμα, στατική ή δυναμική, τρόπος οργάνωσης, μέσο αποθήκευσης κ.λπ.), του τύπου των δεδομένων (για παράδειγμα, ακέραιοι, κείμενο, χαρτογραφικά δεδομένα, χρονοσειρές κ.λπ.), όσο και της αναζήτησης (δηλαδή, με βάση το πρωτεύον ή το δευτερεύον κλειδί κλπ.).

Ερώτηση 3.4

Ποια είναι η σημαντική διαφορά μεταξύ κύριας και δευτερεύουσας μνήμης;

Είναι γνωστό ότι μία σημαντική διαφορά μεταξύ κύριας μνήμης και μαγνητικού δίσκου είναι ότι στην περίπτωση του δίσκου, τα δεδομένα δεν χάνονται, αν διακοπεί η ηλεκτρική παροχή. Ετσι, τα δεδομένα των αρχείων διατηρούνται ακόμη και μετά τον τερματισμό ενός προγράμματος, κάτι που δεν συμβαίνει στην περίπτωση των δομών της κύριας μνήμης, όπως είναι οι πίνακες, όπου τα δεδομένα χάνονται όταν τελειώσει το πρόγραμμα.

3.2 Σπουδαιότητα αλγορίθμων

Ερώτηση 3.5

Τι ονομάζουμε δομή δεδομένων;

Τα δεδομένα ενός προβλήματος αποθηκεύονται στον υπολογιστή, είτε στην κύρια μνήμη του είτε στη δευτερεύουσα μνήμη του. Η αποθήκευση αυτή δεν γίνεται κατά ένα τυχαίο τρόπο αλλά συστηματικά, δηλαδή χρησιμοποιώντας μία δομή.

Δομή Δεδομένων είναι ένα σύνολο αποθηκευμένων δεδομένων που υφίστανται επεξεργασία από ένα σύνολο λειτουργιών.

Κάθε μορφή δομής δεδομένων αποτελείται από ένα σύνολο κόμβων(nodes).

Ερώτηση 3.6

Ποιες είναι οι βασικές λειτουργίες επί των δομών δεδομένων;

Κάθε μορφή δομής δεδομένων αποτελείται από ένα σύνολο κόμβων (nodes). Οι βασικές λειτουργίες (ή αλλιώς πράξεις) επί των δομών δεδομένων είναι οι ακόλουθες:

1. **Προσπέλαση**, πρόσβαση σε ένα κόμβο με σκοπό να εξετασθεί ή να τροποποιηθεί το περιεχόμενο του.
2. **Εισαγωγή**, δηλαδή η προσθήκη νέων κόμβων σε μία υπάρχουσα δομή.
3. **Διαγραφή**, που αποτελεί το αντίστροφο της εισαγωγής, δηλαδή ένας κόμβος αφαιρείται από μία δομή.
4. **Αναζήτηση**, κατά την οποία προσπελαύνονται οι κόμβοι μιας δομής, προκειμένου να εντοπιστούν ένας ή περισσότεροι που έχουν μια δεδομένη ιδιότητα.
5. **Ταξινόμηση**, όπου οι κόμβοι μιας δομής διατάσσονται κατά αύξουσα ή φθίνουσα σειρά.
6. **Αντιγραφή**, κατά την οποία όλοι οι κόμβοι ή μερικοί από τους κόμβους μίας δομής αντιγράφονται σε μία άλλη δομή.
7. **Συγχώνευση**, κατά την οποία δύο ή περισσότερες δομές συνενώνονται σε μία ενιαία δομή.
8. **Διαχωρισμός**, που αποτελεί την αντίστροφη πράξη της συγχώνευσης.

Ερώτηση 3.7

Χρησιμοποιούνται και οι οκτώ λειτουργίες σε κάποια δομή;

Στην πράξη σπάνια χρησιμοποιούνται και οι οκτώ λειτουργίες για κάποια δομή. Συνηθέστατα παρατηρείται το φαινόμενο μία δομή δεδομένων να είναι αποδοτικότερη από μία άλλη δομή με κριτήριο κάποια λειτουργία, για παράδειγμα την αναζήτηση, αλλά λιγότερο αποδοτική για κάποια άλλη λειτουργία, για παράδειγμα την εισαγωγή.

Αυτές οι παρατηρήσεις εξηγούν αφ' ενός την ύπαρξη διαφορετικών δομών, και αφ' ετέρου τη σπουδαιότητα της επιλογής της κατάλληλης δομής κάθε φορά.

Ερώτηση 3.8

Ποια είναι η εξάρτηση μεταξύ της δομής δεδομένων και του αλγορίθμου που επεξεργάζεται τη δομή;

Υπάρχει μεγάλη εξάρτηση μεταξύ της δομής δεδομένων και του αλγόριθμου που επεξεργάζεται τη δομή. Μάλιστα, το πρόγραμμα πρέπει να θεωρεί τη δομή δεδομένων και τον αλγόριθμο ως μία αδιάσπαστη ενότητα. Η παρατήρηση αυτή δικαιολογεί την εξίσωση που διατυπώθηκε το 1976 από τον Wirth, η οποία είναι:

Αλγόριθμοι + Δομές Δεδομένων = Προγράμματα

Ερώτηση 3.9

Σε ποιες κατηγορίες χωρίζονται οι δομές δεδομένων;

Οι δομές δεδομένων διακρίνονται σε δύο μεγάλες κατηγορίες:

- τις στατικές
- και τις δυναμικές.

Ερώτηση 3.10

Τι γνωρίζετε για τις δυναμικές δομές δεδομένων;

Οι δυναμικές δομές δεν αποθηκεύονται σε συνεχόμενες θέσεις μνήμης αλλά στηρίζονται στην τεχνική της λεγόμενης δυναμικής παραχώρησης μνήμης.

Με άλλα λόγια, οι δομές αυτές δεν έχουν σταθερό μέγεθος, αλλά ο αριθμός των κόμβων τους μεγαλώνει και μικραίνει καθώς στη δομή εισάγονται νέα δεδομένα ή διαγράφονται κάποια δεδομένα αντίστοιχα.

Ολες οι σύγχρονες γλώσσες προγραμματισμού προσφέρουν τη δυνατότητα δυναμικής παραχώρησης μνήμης.

Ερώτηση 3.11

Τι γνωρίζετε για τις στατικές δομές δεδομένων;

Με τον όρο στατική δομή δεδομένων εννοείται ότι το ακριβές μέγεθος της απαιτούμενης κύριας μνήμης καθορίζεται κατά τη στιγμή του προγραμματισμού τους, και κατά συνέπεια κατά τη στιγμή της μετάφρασης τους και όχι κατά τη στιγμή της εκτέλεσης τους προγράμματος.

Μία άλλη σημαντική διαφορά σε σχέση με τις δυναμικές δομές είναι ότι τα στοιχεία των στατικών δομών αποθηκεύονται σε συνεχόμενες θέσεις μνήμης.

3.3 Πίνακες

Ερώτηση 3.12

Τι ορίζουμε ως πίνακα;

Οι στατικές δομές υλοποιούνται με πίνακες που μας είναι γνωστοί από άλλα μαθήματα και υποστηρίζονται από κάθε γλώσσα προγραμματισμού.

Μπορούμε να ορίσουμε τον πίνακα ως μια δομή που περιέχει στοιχεία του ίδιου τύπου (δηλαδή ακέραιους, πραγματικούς κλπ.).

Ερώτηση 3.13

Πως γίνεται η δήλωση των στοιχείων ενός πίνακα;

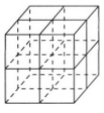
Η δήλωση των στοιχείων ενός πίνακα και η μέθοδος αναφοράς τους εξαρτάται από τη συγκεκριμένη γλώσσα υψηλού επιπέδου που χρησιμοποιείται.

Ομως, γενικά η αναφορά στα στοιχεία ενός πίνακα γίνεται με τη χρήση του συμβολικού ονόματος του πίνακα ακολουθούμενου από την τιμή ενός ή περισσότερων δεικτών σε παρένθεση ή αγκύλη.

Ερώτηση 3.14

Τι διαστάσεις μπορεί να έχει ένας πίνακας;

Ενας πίνακας μπορεί να είναι μονοδιάστατος, αλλά στη γενικότερη περίπτωση μπορεί να είναι δισδιάστατος, τρισδιάστατος και γενικά ν-διάστατος πίνακας.



Οσον αφορά στους δισδιάστατους πίνακες σημειώνεται ότι αν το μέγεθος των δύο διαστάσεων είναι ίσο, τότε ο πίνακας λέγεται τετραγωνικός και γενικά συμβολίζεται ως πίνακας NxN.

Μάλιστα μπορούμε να θεωρήσουμε το δισδιάστατο πίνακα ότι είναι ένας μονοδιάστατος πίνακας, όπου κάθε θέση του περιέχει ένα νέο μονοδιάστατο πίνακα.

Ερώτηση 3.15

Να γράψετε τον αλγόριθμο εύρεσης του μικρότερου στοιχείου ενός μονοδιάστατου πίνακα 100 στοιχείων.

```
Αλγόριθμος Ελαχ_ Πίνακα
    Δεδομένα // table//
     Min← table[1]
     Για i από 2 μέχρι 100
        Αν table[i]<min τότε min[ ]table[i]
     Τέλος_ επανάληψης
     Αποτελέσματα //min//
Τέλος Ελαχ_ Πίνακα
```

Ερώτηση 3.16

Να γράψετε τον αλγόριθμο εύρεσης αθροίσματος κατά γραμμή, κατά στήλη και συνολικά. των στοιχείων δισδιάστατου πίνακα table με m γραμμές n στήλες.

```
Αλγόριθμος Αθρ_ Πίνακα
    Δεδομένα //m, n, table//
    sum←0
    Για i από 1 μέχρι m
        row[i]←0
    Τέλος_ επανάληψης
    Για j από 1 μέχρι n
        col[j]←0
    Τέλος_ επανάληψης
    Για i από 1 μέχρι m
        Για j από 1 μέχρι n
            sum←sum + table[i, j]
            row[i]←row[i] + table[i, j]
            col[j] ←col[j] + table[i, j]
        Τέλος_ επανάληψης
     Τέλος_ επανάληψης
     Αποτελέσματα //sum, row, col//
Τέλος Αθρ_ Πίνακα
```

Παρατήρηση:

Ο διπλός εμφωλευμένος βρόχος που ακολουθεί τους δύο πρώτους απλούς βρόχους, είναι η καρδιά του αλγορίθμου, όπου γίνονται οι υπολογισμοί που ζητά η εκφώνηση του παραδείγματος. Γενικά σε εμφωλευμένους βρόχους, μία τιμή μεταβλητής του εξωτερικού βρόχου παραμένει σταθερή, όσο μεταβάλλεται η τιμή της μεταβλητής του εσωτερικού βρόχου. Πιο συγκεκριμένα, στον αλγόριθμο μας αρχικά το i λαμβάνει την τιμή 1 και το j διαδοχικά τις τιμές 1,2,...,n. Κατόπιν, το i λαμβάνει την τιμή 2, ενώ το j και πάλι λαμβάνει διαδοχικά τις τιμές 1,2,...,n. Η διαδικασία αυτή επαναλαμβάνεται μέχρι το i να λάβει την τιμή m.

3.4 Στοίβα

Ερώτηση 3.17

Ποια δομή δεδομένων ονομάζεται Στοίβα;

Στοίβα (stack), ονομάζεται μια δομή δεδομένων το σύνολο των στοιχείων της οποίας είναι διατεταγμένο με τέτοιο τρόπο, ώστε τα στοιχεία που βρίσκονται στην κορυφή της στοίβας λαμβάνονται πρώτα, ενώ αυτά που βρίσκονται στο βάθος της στοίβας λαμβάνονται τελευταία.

Η παραπάνω μέθοδος ονομάζεται Τελευταίο Μέσα, Πρώτο έξω ή LIFO (=Last In First Out).

Ερώτηση 3.18

Ποιες είναι οι κύριες λειτουργίες μιας Στοίβας;

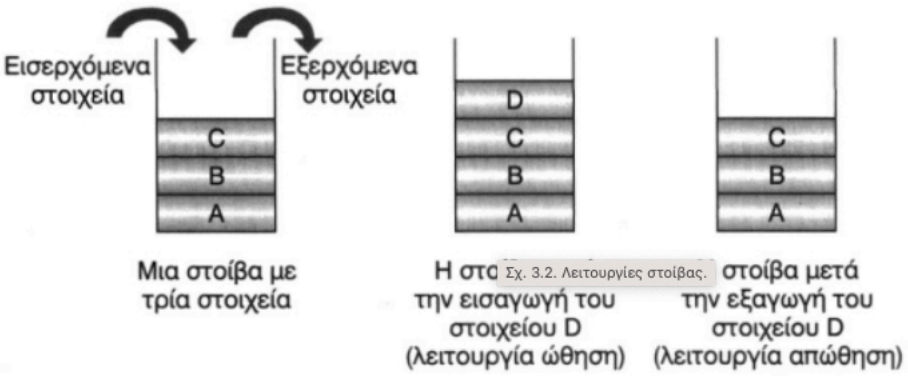
Οι κύριες λειτουργίες σε μια στοίβα είναι δύο:

- Η ώθηση (push) στοιχείου στην κορυφή της στοίβας.

  Στη διαδικασία της ώθησης ελέγχουμε αν η στοίβα είναι γεμάτη. Στην περίπτωση που προσπαθήσουμε να «προσθέσουμε» ένα στοιχείο σε μια ήδη γεμάτη στοίβα, έχουμε υπερχείλιση (overflow) της στοίβας.

- Η απώθηση (pop) στοιχείου από τη στοίβα.

  Στη διαδικασία της απώθησης ελέγχουμε αν υπάρχει ένα τουλάχιστον στοιχείο στη στοίβα. Στην περίπτωση που προσπαθήσουμε να «αφαιρέσουμε» ένα στοιχείο από μία κενή στοί-βα, έχουμε υποχείλιση (underflow) της στοίβας.



Σχ. 3.2. Λειτουργίες στοίβας.

Ερώτηση 3.19

Περιγράψτε την υλοποίηση Στοίβας με χρήση μονοδιάστατου πίνακα.

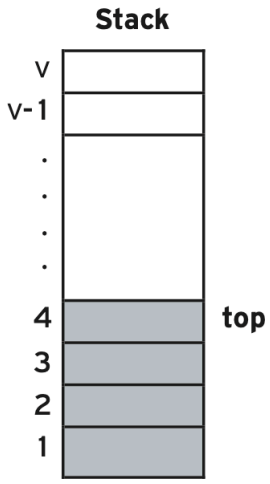
Μια στοίβα μπορεί να υλοποιηθεί πολύ εύκολα με τη βοήθεια ενός μονοδιάστατου πίνακα, όπως φαίνεται στο σχήμα.

- Χρησιμοποιούμε μια βοηθητική μεταβλητή (με όνομα συνήθως top), που δείχνει το στοιχείο που τοποθετήθηκε τελευταίο στην κορυφή της στοίβας.
- Η ώθηση ενός νέου στοιχείου στη στοίβα (εισαγωγή στοιχείου στον πίνακα) γίνεται πάντα στην κορυφή της. Συγκεκριμένα, η μεταβλητή top αυξάνεται κατά ένα:

  $$top \leftarrow top + 1$$

  και στη συνέχεια γίνεται η ώθηση του στοιχείου.
- Η απώθηση ενός στοιχείου από τη στοίβα (εξαγωγή από τον πίνακα) γίνεται πάντα από την κορυφή της στοίβας. Συγκεκριμένα, εξάγεται το στοιχείο που δείχνει η μεταβλητή top και στη συνέχεια η μεταβλητή top μειώνεται κατά ένα:

  $$top \leftarrow top - 1$$



Εικόνα 1. 1.
Υλοποίηση Στοίβας

Ερώτηση 3.20

Ποιες παρατηρήσεις μπορούμε να κάνουμε για τον δείκτη top στην υλοποίηση Στοίβας με χρήση μονοδιάστατου πίνακα;

- Σε μια κενή στοίβα/πίνακα θεωρούμε ότι η αρχική τιμή της μεταβλητής top είναι μηδέν ($top \leftarrow 0$). Η μεταβλητή top είναι η μεταβλητή που δείχνει τη θέση που τοποθετήθηκε το τελευταίο στοιχείο στη στοίβα/πίνακα (δηλ. δείχνει την κορυφή της στοίβας).
- Κατά την ώθηση ενός στοιχείου στη στοίβα (εισαγωγή ενός στοιχείου στον πίνακα), πρώτα αυξάνεται η τιμή της μεταβλητής top κατά ένα, δηλ. $top \leftarrow top + 1$, και στη συνέχεια γίνεται η ώθηση του στοιχείου στην κορυφή της στοίβας.
- Κατά την απώθηση ενός στοιχείου από τη στοίβα (εξαγωγή στοιχείου από τον πίνακα) μειώνεται η τιμή της μεταβλητής top κατά ένα, δηλ. $top \leftarrow top - 1$. Στην απώθηση δε διαγράφεται το στοιχείο, στην πραγματικότητα δε γίνεται καμία παρέμβαση στα περιεχόμενα του πίνακα. Απλώς ο δείκτης κορυφή δείχνει στην προηγούμενη θέση.

Ερώτηση 3.21

Να αναπτύξετε τμήμα προγράμματος σε ΓΛΩΣΣΑ, που πραγματοποιεί ώθηση στοιχείου στην κορυφή της στοίβας με χρήση μονοδιάστατου πίνακα Α, 10 θέσεων.

ΓΡΑΨΕ 'Δώσε στοιχείο για να εισαχθεί στη στοίβα Α'
ΔΙΑΒΑΣΕ στοιχείο
ΑΝ top<10 **ΤΟΤΕ**
  $top \leftarrow top + 1$
  $A[top] \leftarrow$ στοιχείο
ΑΛΛΙΩΣ
  ΓΡΑΨΕ 'Υπερχείλιση στοίβας'
ΤΕΛΟΣ_ΑΝ

Ερώτηση 3.22

Να αναπτύξετε τμήμα προγράμματος σε ΓΛΩΣΣΑ, που πραγματοποιεί απώθηση στοιχείου από στοίβας με χρήση μονοδιάστατου πίνακα Α, 10 θέσεων.

ΑΝ top>=1 **ΤΟΤΕ**
  ΓΡΑΨΕ $A[top]$
  $top \leftarrow top - 1$
ΑΛΛΙΩΣ
  ΓΡΑΨΕ 'Υποχείλιση στοίβας'
ΤΕΛΟΣ_ΑΝ

3.5 Ουρά

Ερώτηση 3.23

Ποια δομή δεδομένων ονομάζεται Ουρά;

Ουρά (Queue), ονομάζεται μια δομή δεδομένων το σύνολο των στοιχείων της οποίας είναι διατεταγμένο με τέτοιο τρόπο, ώστε τα στοιχεία που τοποθετήθηκαν πρώτα στην ουρά να λαμβάνονται επίσης πρώτα.

Η παραπάνω μέθοδος ονομάζεται Πρώτο Μέσα, Πρώτο έξω ή FIFO (=First In First Out).

Ερώτηση 3.24

Ποιες είναι οι κύριες λειτουργίες μιας Ουράς;

Οι κύριες λειτουργίες που εκτελούνται σε μια ουρά είναι δύο: 1. Η εισαγωγή (enqueue) στοιχείου στο πίσω άκρο της ουράς. 2. Η εξαγωγή (dequeue) στοιχείου από το εμπρός άκρο της ουράς.

Ερώτηση 3.25

Περιγράψτε την υλοποίηση Ουράς με χρήση μονοδιάστατου πίνακα.

Μια ουρά μπορεί να υλοποιηθεί με τη βοήθεια ενός μονοδιάστατου πίνακα, όπως φαίνεται στο σχήμα.

- Χρησιμοποιούμε δύο μεταβλητές, την front (ή εμπρός) που δείχνει τη θέση του 1ου στοιχείου της ουράς και την rear (ή πίσω) που δείχνει τη θέση του τελευταίου στοιχείου. Ως αρχικές τιμές των μεταβλητών rear και front θεωρούμε το μηδέν.
- Η εισαγωγή ενός νέου στοιχείου γίνεται από το πίσω άκρο της ουράς και η τιμή της μεταβλητής rear αλλάζει ως εξής:

$$rear \leftarrow rear + 1$$

  Κατά την εισαγωγή, πρώτα αυξάνουμε τον δείκτη rear κατά ένα και μετά εισάγουμε το στοιχείο στον πίνακα.
- Η εξαγωγή ενός στοιχείου γίνεται από το εμπρός άκρο της ουράς και η τιμή της μεταβλητής front αλλάζει ως εξής:

$$front \leftarrow front + 1$$

  Κατά την εξαγωγή ενός στοιχείου, αυξάνε- ται ο δείκτης front κατά ένα (δείχνει στην επόμενη θέση του πίνακα) χωρίς στην πραγματικότητα να γίνεται καμία παρέμβα- ση στα περιεχόμενα του πίνακα (χωρίς να διαγράφεται κάποιο στοιχείο).

Σχ. 3.4. Εισαγωγή και εξαγωγή από ουρά.

Ερώτηση 3.26

Να αναπτύξετε τμήμα προγράμματος σε ΓΛΩΣΣΑ που υλοποιεί την εισαγωγή στοιχείου σε ουρά, με χρήση μονοδιάστατου πίνακα Α, 10 θέσεων.

ΓΡΑΨΕ 'Δώσε στοιχείο για να εισαχθεί στη Ουρά Α'
ΔΙΑΒΑΣΕ στοιχείο
ΑΝ rear=10 **ΤΟΤΕ**
  ΓΡΑΨΕ 'Γεμάτη ουρά'
ΑΛΛΙΩΣ_ΑΝ (front=0 **ΚΑΙ** rear=0) **ΤΟΤΕ**
  $front \leftarrow 1$
  $rear \leftarrow 1$
  $A[rear] \leftarrow$ στοιχείο
ΑΛΛΙΩΣ
  $rear \leftarrow rear + 1$
  $A[rear] \leftarrow$ στοιχείο
ΤΕΛΟΣ_ΑΝ

Ερώτηση 3.27

Να αναπτύξετε τμήμα προγράμματος σε ΓΛΩΣΣΑ, που πραγματοποιεί εξαγωγή στοιχείου από ουρά με χρήση μονοδιάστατου πίνακα Α, 10 θέσεων.

ΑΝ (front=0 **ΚΑΙ** rear=0) **ΤΟΤΕ**
  ΓΡΑΨΕ 'Αδεια ουρά'
ΑΛΛΙΩΣ_ΑΝ (front=rear) **ΤΟΤΕ**
  ΓΡΑΨΕ 'Εξάγεται το στοιχείο ', A[front]
  $front \leftarrow 0$
  $rear \leftarrow 0$
ΑΛΛΙΩΣ
  ΓΡΑΨΕ 'Εξάγεται το στοιχείο ', A[front]
  $front \leftarrow front + 1$
ΤΕΛΟΣ_ΑΝ

3.6 Αναζήτηση

Ερώτηση 3.28

Γιατί υπάρχουν αρκετές μέθοδοι αναζήτησης σε πίνακα;

Υπάρχουν αρκετές μέθοδοι αναζήτησης σε πίνακα που εξαρτώνται κυρίως από το, αν ο πίνακας είναι ταξινομημένος ή όχι.

Μια άλλη παράμετρος είναι, αν ο πίνακας περιέχει στοιχεία που είναι όλα διάφορα μεταξύ τους ή όχι. Τα στοιχεία του πίνακα μπορεί να είναι αριθμητικά ή αλφαριθμητικά.

Ερώτηση 3.29

Να περιγραφεί η λειτουργία της σειριακής (γραμμική) αναζήτησης.

Η λειτουργία της αναζήτησης αποσκοπεί στην εύρεση κάποιου στοιχείου μέσα σε μια δομή.

Η σειριακή αναζήτηση σε πίνακα, ελέγχει όλα τα στοιχεία της δομής του πίνακα, ξεκινώντας από το πρώτο, συγκρίνοντας τα με τη τιμή που αναζητείται. Η αναζήτηση μπορεί να είναι επιτυχής, δηλαδή να εντοπιστεί η ζητούμενη τιμή στον πίνακα ή ανεπιτυχής δηλαδή να εξαντληθούν τα στοιχεία του πίνακα χωρίς να βρεθεί η συγκεκριμένη τιμή.

Ερώτηση 3.30

Ποιές παραλλαγές του αλγορίθμου της σειριακής αναζήτησης υπάρχουν;

- Να υπάρχει πολλές φορές η ζητούμενη τιμή και να αναζητείται μόνο το πρώτο
- Να υπάρχει πολλές φορές η ζητούμενη τιμή και να αναζητούνται όλα τα στοιχεία με αυτή την τιμή και όχι μόνο το πρώτο
- Να είναι ο πίνακας ταξινομημένος ώστε να αποφεύγονται περιττές επαναλήψεις.

Ερώτηση 3.31

Δώστε ένα παράδειγμα σειριακής αναζήτησης με πίνακα μη ταξινομημένο.

 Για παράδειγμα, στο παραπάνω σχήμα παρουσιάζεται ένας πίνακας που περιέχει εννέα αταξινόμητους ακεραίους.

Ετσι, για την επιτυχή αναζήτηση της τιμής 56 απαιτούνται 4 προσπελάσεις.

Αντίθετα, για την αναζήτηση της (ανύπαρκτης) τιμής 11 απαιτούνται 9 προσπελάσεις στον πίνακα, δηλαδή σάρωση ολόκληρου του πίνακα.

Ερώτηση 3.32

Δώστε ένα παράδειγμα σειριακής αναζήτησης με πίνακα ταξινομημένο.

Στο παραπάνω σχήμα παρουσιάζεται ένας πίνακας που περιέχει τα ίδια στοιχεία αλλά σε ταξινομημένη μορφή. Στον πίνακα αυτό η ανεπιτυχής αναζήτηση για την τιμή 11 τερματίζει μετά την τρίτη προσπάθεια και την ανάγνωση του αριθμού 12.

Ερώτηση 3.33

Να γραφεί ο αλγόριθμος της σειριακής αναζήτησης.

Η πιο απλή μορφή αναζήτησης στοιχείου σε πίνακα είναι η σειριακή ή γραμμική μέθοδος.

Ετσι για τον επόμενο αλγόριθμο αναζητείται η τιμή ΚΕΥ στο μη ταξινομημένο πίνακα TABLE, n θέσεων. Μετά την εκτέλεση του αλγορίθμου η μεταβλητή position επιστρέφει την τιμή 0, αν η αναζήτηση είναι ανεπιτυχής, ενώ αν η αναζήτηση είναι επιτυχής, τότε επιστρέφει τη θέση του στοιχείου στον πίνακα (δηλαδή, έναν αριθμό από 1 ως η).

```
Αλγόριθμος Σειριακή_ Αναζήτηση
    Δεδομένα //n, table, key //
    Done← ψευδής
    Position←0
    i←1
    Οσο (done=ψευδής) και (i<=n) επανάλαβε
        Αν table[i] = key τότε
            Done←αληθής
            Position← i
        Αλλιώς
```

```
            i←i+1
        Τέλος_ αν
    Τέλος_ επανάληψης
    Αποτελέσματα // done, position//
Τέλος Σειριακή_ Αναζήτηση
```

Οπως αναφέρθηκε, τα στοιχεία που περιέχονται στον πίνακα table δεν είναι ταξινομημένα.

Επίσης, ο προηγούμενος αλγόριθμος ισχύει για την περίπτωση όπου κάθε στοιχείο υπάρχει μία μόνο φορά στον πίνακα.

Αν κάποιο στοιχείο εμφανίζεται στον πίνακα περισσότερο από μία φορές, τότε ο αλγόριθμος πρέπει να τροποποιηθεί κατά το εξής: η μεταβλητή done είναι περιττή και η αναζήτηση συνεχίζεται μέχρι το τέλος του πίνακα ελέγχοντας με τη συνθήκη i<=n.

Εξάλλου, αν τα στοιχεία του πίνακα είναι ταξινομημένα, τότε ο αλγόριθμος πρέπει να σταματήσει, μόλις συναντήσει κάποιο στοιχείο που είναι μεγαλύτερο από το αναζητούμενο. (table[i]<=key, αν ο πίνακας είναι ταξινομημένος κατά αύξουσα σειρά).

Ερώτηση 3.34

Πότε δικαιολογείται η χρήση της σειριακής αναζήτησης;

Η σειριακή μέθοδος αναζήτησης είναι η πιο απλή, αλλά και η λιγότερη αποτελεσματική μέθοδος αναζήτησης. Ετσι, δικαιολογείται η χρήση της μόνο σε περιπτώσεις όπου:

1. Ο πίνακας είναι μη ταξινομημένος
2. Ο πίνακας είναι μικρού μεγέθους (για παράδειγμα, n<20)
3. Η αναζήτηση σε ένα συγκεκριμένο πίνακα γίνεται σπάνια

Ερώτηση 3.35

Να γράψετε τον αλγόριθμο της δυαδικής αναζήτησης.

Στη δυαδική αναζήτηση θα πρέπει ο πίνακας να είναι ταξινομημένος. Η βασική ιδέα είναι η εξής: Αναζητάμε το στοιχείο key στον πίνακα table. Πηγαίνουμε στο μέσο του πίνακα και αν το στοιχείο του είναι αυτό που ψάχνουμε , η αναζήτηση σταματά. Εάν δεν είναι αυτό, ψάχνουμε στη συνέχεια στο μισό του πίνακα όπου το key θα βρίσκεται. Ετσι διαμερίζεται ο πίνακας σε κομμάτια έως ότου να βρούμε το key ή να έχουμε εξαντλήσει την αναζήτηση και να μην υπάρχει το key. **Αλγόριθμος**

```
Δυαδική_ Αναζήτηση
        Δεδομένα //N, table, key //
        βρέθηκε ←ψευδής
        left ←1
        right ←N
        θεση ←0
        ΟΣΟ (left<=right) ΚΑΙ ( βρέθηκε =ΨΕΥΔΗΣ) ΕΠΑΝΑΛΑΒΕ
            mid ← (left+right) DIV 2
            ΑΝ table[mid]=key ΤΟΤΕ
```

```
            θεση ← mid
            βρέθηκε ←αληθής
        ΑΛΛΙΩΣ_ ΑΝ key>table[mid] ΤΟΤΕ
            left←mid+1
        ΑΛΛΙΩΣ_ ΑΝ key<A[mid] ΤΟΤΕ
            right ← mid-1
        ΤΕΛΟΣ_ ΑΝ
    ΤΕΛΟΣ_ ΕΠΑΝΑΛΗΨΗΣ
    ΑΝ βρέθηκε = ΑΛΗΘΗΣ ΤΟΤΕ
        ΓΡΑΨΕ 'Ο αριθμός ', key,' βρέθηκε στην ', θεση,'η θέση του πίνακα'
    ΑΛΛΙΩΣ
        ΓΡΑΨΕ 'Ο αριθμός ',key,' δεν βρέθηκε'
    ΤΕΛΟΣ_ ΑΝ
Τέλος Δυαδική_ Αναζήτηση
```

3.7 Ταξινόμηση

Ερώτηση 3.36

Να δοθεί ο τυπικός ορισμός της ταξινόμησης.

Δοθέντων των στοιχείων $\alpha_1, \alpha_2, \ldots, \alpha_n$ η ταξινόμηση συνίσταται στη μετάθεση (permutation) της θέσης των στοιχείων, ώστε να τοποθετηθούν σε μία σειρά $\alpha_{k1}, \alpha_{k2}, \ldots, \alpha_{kn}$ έτσι ώστε, δοθείσης μίας συνάρτησης διάταξης (ordering function), f, να ισχύει:

$$f(\alpha_{k1}) \leq f(\alpha_{k2}) \leq \cdots \leq f(\alpha_{kn})$$

Αξίζει να σημειωθεί ότι η προηγούμενη συνάρτηση διάταξης μπορεί να τροποποιηθεί, ώστε να καλύπτει και την περίπτωση που η ταξινόμηση γίνεται με φθίνουσα τάξη (descending sequence) μεγέθους.

Ερώτηση 3.37

Να δοθεί ο ορισμός της ταξινόμησης.

Η τακτοποίηση των κόμβων μίας δομής με μία ιδιαίτερη σειρά είναι μία πολύ σημαντική λειτουργία που ονομάζεται ταξινόμηση (sorting) ή διάταξη (ordering). Συνήθως η σειρά αυτή είναι η αύξουσα τάξη (ascending sequence) της τιμής των μεγεθών προς ταξινόμηση.

Ερώτηση 3.38

Ποιος είναι ο σκοπός της ταξινόμησης;

Σκοπός της ταξινόμησης είναι να διευκολυνθεί στη συνέχεια η αναζήτηση των στοιχείων του ταξινομημένου πίνακα.

Η χρησιμότητα της ταξινόμησης αποδεικνύεται στην πράξη σε αναρίθμητες περιπτώσεις αναζήτησης αριθμητικών ή αλφαβητικών δεδομένων, όπως σε βιβλιοθηκονομικά συστήματα, λεξικά, τηλεφωνικούς καταλόγους, καταλόγους φόρου εισοδήματος και γενικά παντού όπου γίνεται αναζήτηση αποθηκευμένων αντικειμένων

Ερώτηση 3.39

Ποια μέθοδο ονομάζουμε ταξινόμηση ευθείας ανταλλαγής;

Η μέθοδος της ταξινόμησης ευθείας ανταλλαγής βασίζεται στην αρχή της σύγκρισης και ανταλλαγής ζευγών γειτονικών στοιχείων, μέχρις ότου διαταχθούν όλα τα στοιχεία.

Σύμφωνα με τη μέθοδο αυτή κάθε φορά γίνονται διαδοχικές προσπελάσεις στον πίνακα και μετακινείται το μικρότερο κλειδί της ακολουθίας προς το αριστερό άκρο του πίνακα.

Αν ο πίνακας θεωρηθεί σε κατακόρυφη θέση αντί σε οριζόντια και οι ακέραιοι θεωρηθούν - επιστρατεύοντας αρκετή φαντασία - ως φυσαλίδες σε μία δεξαμενή νερού με βάρη σύμφωνα με την τιμή τους, τότε κάθε προσπέλαση στον πίνακα έχει ως αποτέλεσμα την άνοδο της φυσαλίδας στο κατάλληλο επίπεδο βάρους. Η μέθοδος είναι γνωστή ως ταξινόμηση φυσαλίδας.

Ερώτηση 3.40

Να γραφεί ο αλγόριθμος της φυσαλίδας.

Αλγόριθμος Φυσσαλίδα
Δεδομένα // table, n//
Για i **από** 2 **μέχρι** n
Για j **από** n **μέχρι** i **με_ βήμα** -1
Αν table[j – 1] > table[j] **τότε**
αντιμετάθεσε table[j-1], table[j]
Τέλος_ αν
Τέλο_ επανάληψης
Τέλος_ επανάληψης
Αποτελέσματα // table//
Τέλος Φυσαλίδα

Σημειώνεται ότι η εντολή "**αντιμετάθεσε** table[j-1], table[j]" ανταλλάσσει το περιεχόμενο δύο θέσεων με τη βοήθεια μίας βοηθητικής θέσης. Εναλλακτικά αυτό μπορεί να γίνει με τις εξής τρεις εντολές:

temp ← table[j – 1]
table[j – 1]←table[j]
table[j]← temp

Ερώτηση 3.41

Ο αλγόριθμος της φυσαλίδας είναι ο μοναδικός αλγόριθμος ταξινόμησης;

Για την ταξινόμηση δεδομένων έχουν εκπονηθεί πάρα πολλοί αλγόριθμοι. Αλλοι σχετικά απλοί αλγόριθμοι είναι:

- η ταξινόμηση με επιλογή
- η ταξινόμηση με παρεμβολή.
- και η "γρήγορη ταξινόμηση" (Quicksort) που είναι πιο γρήγορος αλγόριθμος ταξινόμησης.

Η ταξινόμηση φυσαλίδας είναι ο πιο απλός και ταυτόχρονα ο πιο αργός αλγόριθμος ταξινόμησης.

Ερώτηση 3.42

Να γραφεί ο αλγόριθμος ταξινόμησης με επιλογή.

Η ταξινόμηση με επιλογή (selection sort), αποτελεί βασικό τρόπο ταξινόμησης, που υλοποιείται σε ένα μονοδιάστατο πίνακα σε τρία βήματα:

1. Επιλογή του ελάχιστου στοιχείου
2. Ανταλλαγή του ελάχιστου με το πρώτο στοιχείο
3. Επανάληψη των βημάτων 1 και 2 για τα υπόλοιπα στοιχεία του πίνακα

```
Αλγόριθμος Selection_Sort
        Δεδομένα // table, n //
        Για i από 1 μέχρι n-1
                k ← i
                x ← table[i]
                Για j από i+1 μέχρι n
                        Αν x > table[j] Τότε
                                k ← j
                                x ←table[j]
                    Τέλος_ αν
                 Τέλος_ επανάληψης
                table[k] ← table[i]
                table[i] ← x
         Τέλος_ επανάληψης
Τέλος Selection_Sort
```

3.8 Αλλες δομές δεδομένων

3.8.1 Λίστες

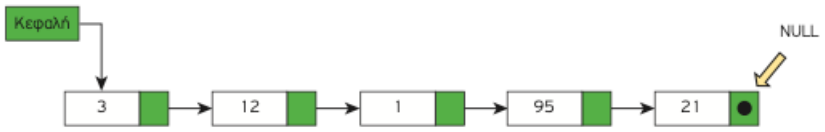
Ερώτηση 3.43

Τι ονομάζεται συνδεδεμένη λίστα;

Μια (απλά) συνδεδεμένη λίστα (linked list) είναι ένα σύνολο κόμβων διατεταγμένων γραμμικά (ο ένας μετά τον άλλο). Κάθε κόμβος περιέχει εκτός από τα δεδομένα του και έναν δείκτη που δείχνει προς τον επόμενο κόμβο. Ο δείκτης του τελευταίου κόμβου δε δείχνει σε κάποιον κόμβο (δείκτης στο κενό). Για να το δηλώσουμε αυτό λέμε ότι το πεδίο δείκτη του τελευταίου κόμβου έχει την τιμή NULL.

Το πεδίο Δεδομένα μπορεί να περιέχει μία ή περισσότερες αλφαριθμητικές ή αριθμητικές πληροφορίες. Ο δείκτης (pointer) είναι ένας ιδιαίτερος τύπος δεδομένων που προσφέρεται από τις περισσότερες σύγχρονες γλώσσες προγραμματισμού.

Ο δείκτης δε λαμβάνει αριθμητικές τιμές όπως ακέραιες, πραγματικές κ.ά., αλλά οι τιμές του είναι διευθύνσεις στην κύρια μνήμη και χρησιμοποιείται ακριβώς για τη σύνδεση των διαφόρων στοιχείων μιας δομής, που είναι αποθηκευμένα σε μη συνεχόμενες θέσεις μνήμης.



Εικόνα 1.3.3. Μία απλά συνδεδεμένη λίστα με 5 κόμβους.

Ερώτηση 3.44

Πώς γίνεται η προσπέλαση σε μια συνδεδεμένη λίστα;

Για να προσπελάσουμε τους κόμβους της λίστας χρειάζεται να γνωρίζουμε τη διεύθυνση (θέση στη μνήμη) του πρώτου κόμβου της λίστας. Η διεύθυνση αυτή αποθηκεύεται σε μία ειδική μεταβλητή που την ονομάζουμε συνήθως Κεφαλή (Head).

Οι κόμβοι μιας (απλά) συνδεδεμένης λίστας είναι διατεταγμένοι σε μια συγκεκριμένη σειρά, χωρίς αυτό να σημαίνει ότι αποθηκεύονται σε συνεχόμενες θέσεις στη μνήμη. Αντίθετα, είναι διασκορπισμένοι σε όλη τη μνήμη και η σύνδεση μεταξύ τους γίνεται μέσω των δεικτών. έχουμε άμεση πρόσβαση μόνο στον πρώτο κόμβο της λίστας. Επομένως, για να εντοπίσουμε κάποιον από τους ενδιάμεσους κόμβους, πρέπει να ξεκινήσουμε από τον πρώτο κόμβο της λίστας και να ακολουθήσουμε τους δείκτες με τη σειρά, μέχρι να φτάσουμε στον επιθυμητό κόμβο.

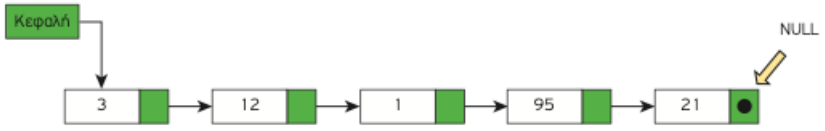
Ερώτηση 3.45

Πώς προσθέτω κόμβο σε μια συνδεδεμένη λίστα;

Εισαγάγουμε ένα νέο κόμβο στη λίστα μας, κάνοντας την κατάλληλη διευθέτηση των δεικτών. Οι απαιτούμενες ενέργειες για την εισαγωγή (παρεμβολή) του νέου κόμβου είναι :

- ο δείκτης του κόμβου μετά τον οποίο θα γίνει η παρεμβολή (N2), να δείχνει τον νέο κόμβο (NEO) και
- ο δείκτης του νέου κόμβου (NEO), να δείχνει αυτό που έδειχνε ο προηγούμενος (στο N3) (δηλαδή, να πάρει την τιμή που είχε πριν την εισαγωγή ο δείκτης του N2).

Με αυτόν τον τρόπο, οι κόμβοι της λίστας διατηρούν τη λογική τους σειρά, αλλά οι φυσικές θέσεις στη μνήμη μπορεί να είναι τελείως διαφορετικές.

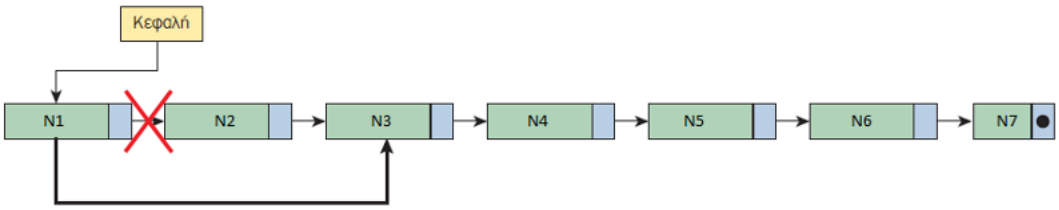


Εικόνα 1.3.3. Μία απλά συνδεδεμένη λίστα με 5 κόμβους.

Ερώτηση 3.46

Πώς διαγράφω κόμβο από μια συνδεδεμένη λίστα;

Για τη διαγραφή ενός κόμβου αρκεί ν' αλλάξει τιμή ο δείκτης του προηγούμενου κόμβου και να δείχνει πλέον στον επόμενο αυτού που διαγράφεται. Ο κόμβος που διαγράφηκε (ο δεύτερος) αποτελεί «άχρηστο δεδομένο» και ο χώρος μνήμης που καταλάμβανε, παραχωρείται για άλλη χρήση.

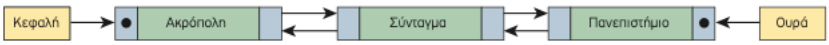


Ερώτηση 3.47

Τι είναι η διπλά συνδεδεμένη λίστα (doubly linked list); Ποια είναι τα πλεονεκτήματα και τα μειονεκτήματα της σε σχέση μη μια απλά συνδεδεμένη λίστα;

Μια λίστα μπορεί να είναι απλά συνδεδεμένη, δηλαδή να μπορούμε να κινηθούμε προς μία μόνο κατεύθυνση, ξεκινώντας από τον πρώτο κόμβο και μετακινούμενοι προς τον τελευταίο, ή να είναι διπλά συνδεδεμέ νη (doubly linked list), δηλαδή, να μπορούμε να τη διατρέξουμε και προς τις δύο κατευθύνσεις. Η χρήση του δεύτερου δείκτη προσφέρει τη δυνατότητα ξεκινώντας από οποιοδήποτε κόμβο της λίστας να μπορούμε να διαβάσουμε τη λίστα και προς τις δυο κατευθύνσεις.

Σε μια διπλά συνδεδεμένη λίστα διευκολύνεται η ταξινόμηση και η αναζήτηση, ωστόσο, αυξάνεται η πολυπλοκότητα στη διαχείριση των κόμβων, καθώς απαιτείται επιπλέον χώρος για τον δεύτερο δείκτη (επιπρόσθετη μνήμη για κάθε κόμβο).



Εικόνα 1.3.8. Οι σταθμοί του μετρό - Μια διπλά συνδεδεμένη λίστα

Ερώτηση 3.48

Γιατί χρησιμοποιούνται οι συνδεδεμένες λίστες στην υλοποίηση της στοίβας και της ουράς

Οι συνδεδεμένες λίστες αξιοποιούνται για την υλοποίηση της στοίβας και της ουράς, λόγω της δυνατότητάς αυξομείωσης του μεγέθους τους. Δηλαδή, ως δυναμική δομή δεδομένων, το μέγεθός της μπορεί να μεταβάλλεται καθώς εισέρχονται νέοι κόμβοι στη λίστα ή διαγράφονται κάποιοι άλλοι.

Ερώτηση 3.49

Πώς μια στοίβα μπορεί να υλοποιηθεί με μία απλά συνδεδεμένη λίστα;

Οι κόμβοι (στοιχεία) εισέρχονται από την αρχή της απλά συνδεδεμένης λίστας και εξέρχονται πάλι από την αρχή της λίστας τότε ο κόμβος (στοιχείο) που εισήλθε τελευταίος εξέρχεται και πρώτος (LIFO). Επομένως η στοίβα μπορεί να υλοποιηθεί με μία απλά συνδεδεμένη λίστα αφού μπορεί να γίνει εισαγωγή κόμβου (στοιχείου) στην αρχή της λίστας και διαγραφή κόμβου (στοιχείου) από την αρχή της λίστας .

Ερώτηση 3.50

Πότε θα χρησιμοποιήσουμε λίστα και πότε πίνακα για την υλοποίηση μιας στοίβας;

Καλύτερη είναι η υλοποίηση στοίβας με συνδεδεμένη λίστα εκτός αν δεν γνωρίζουμε εκ των προτέρων τον μέγιστο αριθμό στοιχείων και θέλουμε μια εύκολη και γρήγορη λύση, οπότε επιλέγουμε την υλοποίηση με πίνακα.

Ερώτηση 3.51

Πώς μια ουρά μπορεί να υλοποιηθεί με μία διπλά συνδεδεμένη λίστα.

Στην ουρά έχουμε τις εξής δύο κύριες λειτουργίες. Εισαγωγή στοιχείου στο πίσω άκρο της ουράς και εξαγωγή στοιχείου από το εμπρός άκρο της ουράς. Η ουρά μπορεί να υλοποιηθεί με μία διπλά συνδεδεμένη λίστα, αφού μπορούμε να κάνουμε εισαγωγή κόμβου στο τέλος της διπλά συνδεδεμένης λίστας. Επίσης μπορούμε να κάνουμε διαγραφή κόμβου στην αρχή της διπλά συνδεδεμένης λίστας (εξαγωγή στοιχείου από το εμπρός άκρο της ουράς).

Ερώτηση 3.52

Ποιες οι διαφορές ανάμεσα σε λίστες και πίνακες;

Παρατηρώντας όμως πιο προσεκτικά, είναι εύκολο να εντοπίσει σημαντικές διαφορές μεταξύ λιστών και πινάκων, οι οποίες παρατίθενται παρακάτω:

- Ο πίνακας θεωρείται μια δομή τυχαίας προσπέλασης, σε αντίθεση με μια λίστα που είναι στην ουσία μια δομή ακολουθιακής ή σειριακής προσπέλασης. Για να φθάσουμε, δηλαδή, σ' έναν κόμβο μιας λίστας πρέπει να περάσουμε από όλους τους προηγούμενους ξεκινώντας από τον πρώτο.
- Ο πίνακας έχει σταθερό μέγεθος, το οποίο δηλώνεται εξαρχής κατά την υλοποίηση. Αυτό γίνεται, διότι ο πίνακας είναι στατική δομή δεδομένων σε αντίθεση με τη λίστα που είναι δυναμική δομή και το μέγεθός της μπορεί να μεταβάλλεται καθώς εισέρχονται νέοι κόμβοι στη λίστα ή διαγράφονται κάποιοι άλλοι.
- Οι κόμβοι της λίστας αποθηκεύονται σε μη συνεχόμενες θέσεις μνήμης σε αντιδιαστολή με τους πίνακες, όπου τα στοιχεία αποθηκεύονται σε συνεχόμενες θέσεις μνήμης.

Ερώτηση 3.53

Ποια είναι τα πλεονεκτήματα των λιστών (έναντι των πινάκων);

Στα πλεονεκτήματα των λιστών (έναντι των πινάκων) συγκαταλέγονται τα εξής:

- Το δυναμικό τους μέγεθος,
- η ευκολία εισαγωγής και διαγραφής από οποιοδήποτε μέρος της λίστας, καθώς και
- η μη αναγκαιότητα δήλωσης του μεγέθους τους.

Ερώτηση 3.54

Ποια τα μειονεκτήματα των λιστών (έναντι των πινάκων);

Στα μειονεκτήματα των λιστών (έναντι των πινάκων) περιλαμβάνονται τα εξής:

- Η τυχαία πρόσβαση στη λίστα δεν επιτρέπεται. Είναι αδύνατο να φτάσετε στον n-οστό κόμβο μιας απλά συνδεδεμένης λίστας χωρίς πρώτα να περάσετε από όλους τους κόμβους διαδοχικά μέχρι τον συγκεκριμένο κόμβο ξεκινώντας από τον πρώτο κόμβο. Εναλλακτικά, στην περίπτωση της διπλά συνδεμένης λίστας μπορείτε να ξεκινήσετε και από τον τελευταίο κόμβο. Επομένως, δεν μπορούμε να πραγματοποιήσουμε με αποτελεσματικό τρόπο δυαδική αναζήτηση σε συνδεδεμένες λίστες.
- Οι συνδεδεμένες λίστες έχουν πολύ μεγαλύτερη επιβάρυνση από τους πίνακες, αφού οι συνδεδεμένοι κόμβοι της λίστας είναι δυναμικά κατανεμημένοι (οι οποίοι είναι λιγότερο αποτελεσματικοί στη χρήση της μνήμης) και κάθε κόμβος στη λίστα πρέπει, επιπλέον, να αποθηκεύσει έναν πρόσθετο δείκτη που θα δείχνει στον επόμενο κόμβο. Στην περίπτωση των διπλά συνδεδεμένων λιστών χρειαζόμαστε επιπλέον έναν δεύτερο δείκτη που θα δείχνει στον προηγούμενο κόμβο.

Ερώτηση 3.55

Ποιες είναι οι βασικές πράξεις των συνδεδεμένων λιστών;

Οι βασικές πράξεις των συνδεδεμένων λιστών είναι οι παρακάτω:

- Εισαγωγή κόμβου στη λίστα (εισαγωγή κόμβου στην αρχή, στο τέλος της λίστας ή ενδιάμεσα).
- Διαγραφή κόμβου από τη λίστα (διαγραφή από την αρχή, το τέλος της λίστας ή ενδιάμεσα).
- έλεγχος για το αν η λίστα είναι κενή.
- Αναζήτηση κόμβου για την εύρεση συγκεκριμένου στοιχείου.
- Διάσχιση της λίστας και προσπέλαση των στοιχείων της (π.χ. εκτύπωση των δεδομένων που περιέχονται σε όλους τους κόμβους της λίστας).

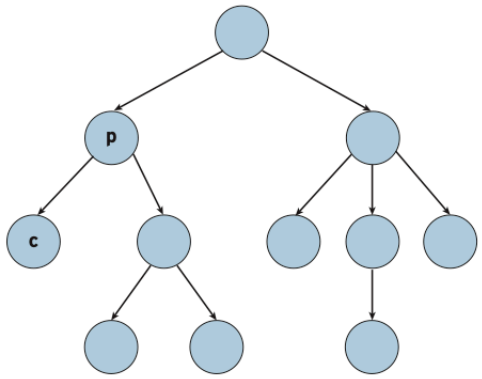
3.8.2 Δένδρα

Ερώτηση 3.56

Τι ονομάζεται δέντρο;

ένα δένδρο (tree) είναι μία δομή που αποτελείται από ένα σύνολο κόμβων και ένα σύνολο ακμών μεταξύ των κόμβων με βάση τους εξής κανόνες:

- Υπάρχει ένας ξεχωριστός κόμβος που ονομάζεται ρίζα. Αυτός είναι ένας κόμβος χωρίς γονέα και βρίσκεται στην κορυφή του δένδρου.
- Για κάθε κόμβο c, εκτός από τη ρίζα, υπάρχει μόνο μια ακμή που καταλήγει στον κόμβο αυτόν ξεκινώντας από κάποιον άλλον κόμβο p. Ο κόμβος p ονομάζεται γονέας του c και ο κόμβος c παιδί του p. ένας κόμβος μπορεί να έχει κανένα, ένα ή περισσότερα παιδιά. όλοι οι κόμβοι, εκτός από έναν, έχουν ακριβώς έναν γονέα.
- Για κάθε κόμβο υπάρχει μία μοναδική διαδρομή, δηλαδή, μια ακολουθία διαδοχικών ακμών, που ξεκινάει από τη ρίζα και τερματίζει σε αυτόν τον κόμβο.
- Δένδρο θεωρούμε και το κενό δένδρο, δηλαδή το δένδρο που δεν έχει ούτε κόμβους, ούτε ακμές. Το κενό δένδρο είναι το μόνο δένδρο χωρίς ρίζα.

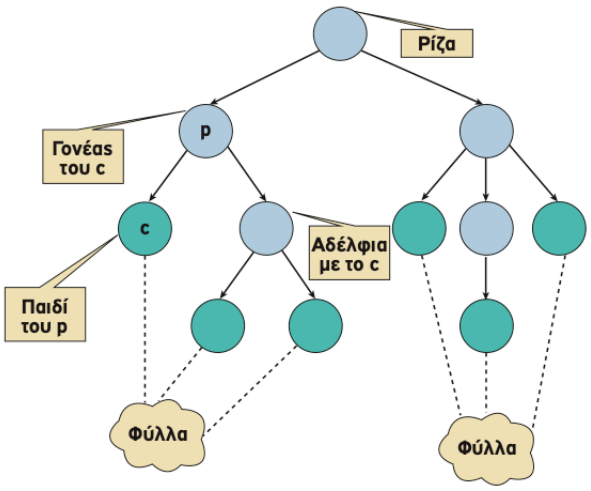


Εικόνα 1.3.1Ο. Δομή ενός δένδρου

Ερώτηση 3.57

Σε ένα δέντρο ποιοι κόμβοι ονομάζονται φύλλα και ποιοι αδέλφια;

Κόμβοι με τον ίδιο γονέα ονομάζονται «αδέλφια». Οι κόμβοι χωρίς παιδιά ονομάζονται «φύλλα».



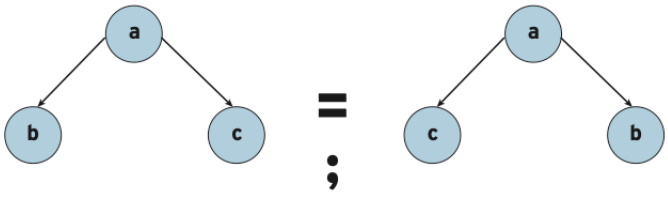
Εικόνα 1.3.11. Σχέσεις μεταξύ των κόμβων ενός δένδρου

Ερώτηση 3.58

Ποια δέντρα ονομάζονται διατεταγμένα;

Τα δέντρα στα οποία για κάθε κόμβο του υπάρχει μία γραμμική σχέση μεταξύ των παιδιών του κόμβου αυτού, αναφερόμαστε σε ένα διατεταγμένο δένδρο

Για παράδειγμα, αν θέλουμε να μοντελοποιήσουμε την ιεραρχική σχέση των μελών μιας οικογένειας και μας ενδιαφέρει να οργανώσουμε τα αδέλφια σύμφωνα με την ηλικία τους, τότε τα αδέλφια που θα έχουν γεννηθεί νωρίτερα θα τοποθετηθούν στην δενδρική δομή πιο αριστερά σε σχέση με αυτά που θα έχουν γεννηθεί αργότερα.



Εικόνα 1.3.15. Πότε δύο δένδρα είναι ίδια

Ερώτηση 3.59

Σε ποιους τομείς της επιστήμης χρησιμοποιούνται τα δέντρα;

Τα δένδρα είναι μία μη-γραμμική ευέλικτη δομή δεδομένων που χρησιμοποιούνται σε πολλούς τομείς της επιστήμης των υπολογιστών, συμπεριλαμβανομένων των λειτουργικών συστημάτων, των γραφικών, των συστημάτων βάσεων δεδομένων, των παιχνιδιών, της τεχνητής νοημοσύνης και της δικτύωσης υπολογιστών.

Ερώτηση 3.60

Τι ονομάζεται δένδρο απόφασης; Δώστε παράδειγμα.

Τα δένδρα απόφασης, όπως πολύ εύκολα μπορείτε να συμπεράνετε από την Εικόνα, είναι δένδρα στα οποία

- κάθε κόμβος αντιπροσωπεύει ένα χαρακτηριστικό (ιδιότητα),
- κάθε ακμή αντιπροσωπεύει μια απόφαση (κανόνα) και
- κάθε φύλλο αντιπροσωπεύει ένα αποτέλεσμα.

Στους αλγορίθμους μηχανικής μάθησης (machine learning) τα δένδρα απόφασης έχουν πρωτεύοντα ρόλο.

Ενα απλό πρόβλημα λήψης απόφασης μπορεί να είναι αυτό της καθαριότητας του φοιτητικού σπιτιού σας.

Εικόνα 1.3.19. Δένδρο Απόφασης

Ερώτηση 3.61

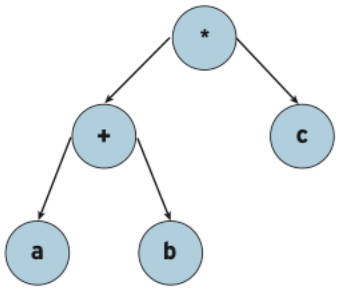
Τι ονομάζεται δένδρο του παιχνιδιού (game tree);

Στα παιχνίδια ο υπολογιστής χρησιμοποιεί ένα ειδικό δένδρο, που ονομάζεται δένδρο του παιχνιδιού (game tree), το οποίο μοντελοποιεί όλες τις πιθανές κινήσεις των παικτών για να νικήσει. Κάθε κόμβος στο δένδρο, αντιπροσωπεύει μία συγκεκριμένη κατάσταση παιχνιδιού και περιέχει πληροφορίες σχετικά με το ποιος παίκτης έχει τη μεγαλύτερη πιθανότητα να κερδίσει από οποιαδήποτε πιθανή κίνηση.

Ερώτηση 3.62

Πώς χρησιμοποιούνται τα δέντρα στον υπολογισμό αριθμητικών εκφράσεων.

Για τον υπολογισμό αριθμητικών εκφράσεων στα φύλλα του δέντρου μπαίνουν οι τελεστέοι, ενώ στους εσωτερικούς κόμβους οι τελεστές.



Εικόνα 1.3.20. (a+b)*c

Ερώτηση 3.63

Για ποιους λόγους θεωρούνται τα δένδρα ισχυρές δομές.

Υπάρχουν δύο λόγοι για τους οποίους τα δένδρα είναι τόσο ισχυρά. Ο πρώτος λόγος αναφέρεται στη δυναμικότητα των δένδρων. Είναι πολύ εύκολο να προσθέσετε, να αφαιρέσετε ή να αναζητήσετε ένα στοιχείο σε ένα δένδρο, όπως θα δούμε στη συνέχεια. Ο δεύτερος βασικός λόγος είναι ότι η δομή των δένδρων μεταφέρει πληροφορίες.

Ερώτηση 3.64

Δώστε μερικά παραδείγματα χρήσης δέντρων.

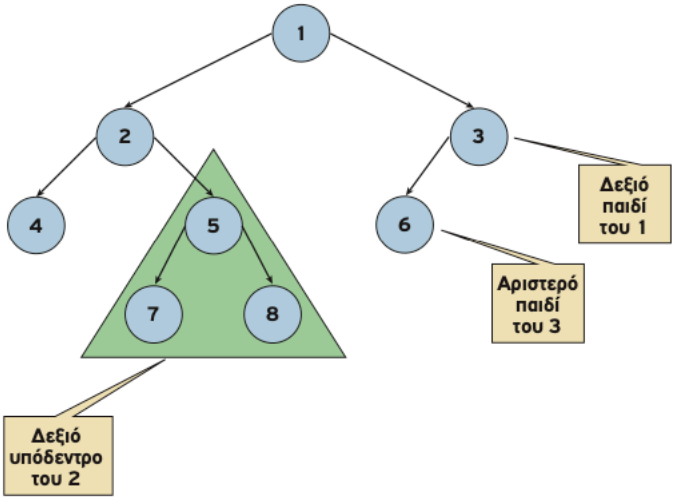
Παραδείγματα χρήσης δέντρων είναι:

- το σύστημα αρχείων του υπολογιστή
- το οικογενειακό δένδρο
- δομή ενός οργανισμού
- πίνακας περιεχομένων ενός βιβλίου
- μέρη που απαρτίζουν την μηχανή ενός αυτοκινήτου

Ερώτηση 3.65

Τι ονομάζεται δυαδικό δέντρο;

Ενα δυαδικό δένδρο (binary tree) είναι ένα διατεταγμένο δένδρο, στο οποίο κάθε κόμβος έχει το πολύ δύο παιδιά, το αριστερό και το δεξί παιδί. Μπορούμε, συνεπώς, να μιλάμε για αριστερό και δεξιό υποδένδρο ενός κόμβου. Στο δυαδικό δένδρο της εικόνας, ο κόμβος 3 έχει ως αριστερό υποδένδρο, το δένδρο με μοναδικό κόμβο το 6 και ως δεξιό υποδένδρο, το κενό δένδρο. Προφανώς, αν ανταλλάξουμε το αριστερό με το δεξιό υποδένδρο ενός κόμβου παίρνουμε ένα διαφορετικό δένδρο.

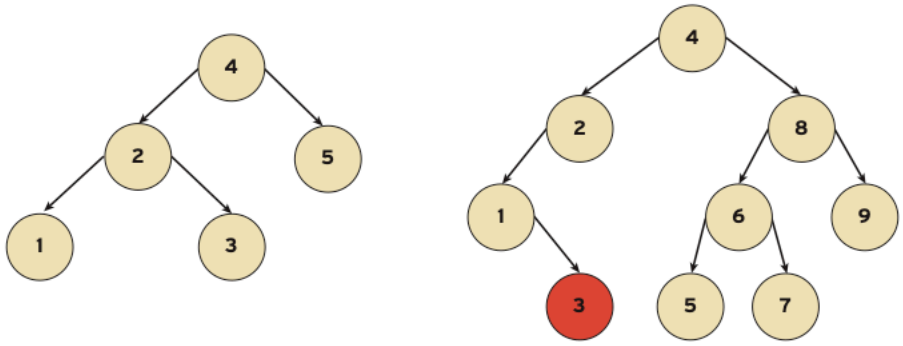


Εικόνα 1.3.21. Δυαδικό δένδρο

Ερώτηση 3.66

Τι ονομάζεται δυαδικό δέντρο αναζήτησης;

Ενα δυαδικό δένδρο αναζήτησης (binary search tree) είναι ένα δυαδικό δένδρο, όπου για κάθε κόμβο u, όλοι οι κόμβοι του αριστερού υποδένδρου έχουν τιμές μικρότερες της τιμής του κόμβου u και όλοι οι κόμβοι του δεξιού υποδένδρου έχουν τιμές μεγαλύτερες (ή ίσες) της τιμής του κόμβου u.



Εικόνα 1.3.23. α: Δυαδικό δένδρο αναζήτησης **β: Μη δυαδικό δένδρο αναζήτησης**

Ερώτηση 3.67

Ποιο ότι είναι το πλεονέκτημα των δυαδικών δένδρων αναζήτησης;

Το πλεονέκτημα βρίσκεται στο ίδιο το όνομα και συγκεκριμένα στη λέξη «αναζήτηση». Η αναζήτηση για μια συγκεκριμένη τιμή γίνεται ταχύτερα χάρη στον τρόπο αποθήκευσης των τιμών.

Τα δυαδικά δένδρα αναζήτησης συνδυάζουν τα πλεονεκτήματα των λιστών, όσον αφορά τις πράξεις της εισαγωγής και της διαγραφής, αλλά και τα πλεονεκτήματα των ταξινομημένων πινάκων, όσον αφορά την πράξη της αναζήτησης.

3.8.3 Γράφοι

Ερώτηση 3.68

Τι ονομάζεται γράφος;

Ενας γράφος (graph) είναι μία δομή που αποτελείται :

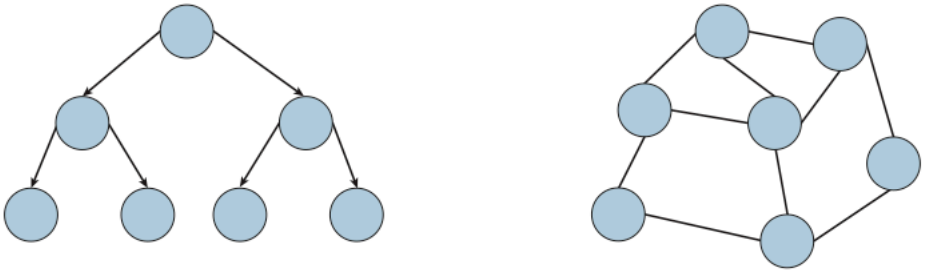
- από ένα σύνολο κόμβων (ή σημείων ή κορυφών)
- και ένα σύνολο γραμμών (ή ακμών ή τόξων) που ενώνουν μερικούς ή όλους τους κόμβους.

Ερώτηση 3.69

Τι είναι αυτό που κάνει ένα δένδρο διαφορετικό από έναν γράφο;

ένα δένδρο μπορεί μόνο να ρέει προς μία κατεύθυνση, από τον κόμβο ρίζας σε κόμβους φύλλων ή κόμβους παιδιών. ένα δένδρο μπορεί να έχει μόνο μονόδρομες συνδέσεις - ένας κόμβος παιδιού μπορεί να έχει μόνο έναν γονέα και ένα δένδρο δεν μπορεί να έχει βρόχους ή κυκλικούς δεσμούς

Με τους γράφους, όλοι αυτοί οι περιορισμοί δεν υπάρχουν. Οι γράφοι δεν έχουν την έννοια ενός κόμβου «ρίζας». Οι κόμβοι μπορούν να συνδεθούν με οποιονδήποτε τρόπο. Για παράδειγμα, ένας κόμβος μπορεί να συνδεθεί με άλλους πέντε (εικόνα β) Οι γράφοι, επίσης, δεν έχουν «μονοκατευθυντική» ροή - αντ' αυτού, μπορεί να έχουν κατεύθυνση ή να μην έχουν καμιά κατεύθυνση.



Εικόνα 1.3.27.

α: Δένδρο **β: Γράφος**

Ερώτηση 3.70

Ποια η συσχέτιση του γράφου με τις λίστες και τα δέντρα;

Ο γράφος αποτελεί την πιο γενική δομή δεδομένων, με την έννοια ότι όλες οι προηγούμενες δομές που παρουσιάστηκαν μπορούν να θεωρηθούν περιπτώσεις γράφων.

Ερώτηση 3.71

Ποιοι οι τύποι του γράφου;

Εάν όλες οι ακμές σε έναν γράφο έχουν κατεύθυνση, ο γράφος ονομάζεται **κατευθυνόμενος γράφος** (directed graph).

Εάν όλες οι ακμές σε έναν γράφο δεν έχουν κατεύθυνση, ο γράφος ονομάζεται **μη κατευθυνόμενος γράφος** (undirected graph) και η διαδρομή μεταξύ των δύο κόμβων είναι αμφίδρομη.

Εικόνα 1.3.28.

Ερώτηση 3.72

Τι τύπου γράφοι είναι τα κοινωνικά δίκτυα facebook, twitter και τι ο παγκόσμιος ιστός;

Το Facebook είναι μη κατευθυνόμενος γράφος επειδή η σχέση μεταξύ δύο χρηστών (κόμβοι), είναι αμφίδρομη. Δεν υπάρχει η έννοια κόμβου «προέλευσης» και «προορισμού» - αντ' αυτού, είσαι φίλος μου και είμαι δικός σου.

Το Twitter είναι κατευθυνόμενος γράφος επειδή μπορώ να σε ακολουθήσω, αλλά δεν απαιτείται να με ακολουθήσεις. Κάθε ακμή που δημιουργούμε αντιπροσωπεύει μια μονόδρομη σχέση.

- Οταν με ακολουθείς στο Twitter, δημιουργείται μια ακμή στον γράφο με τον λογαριασμό σου ως κόμβο προέλευσης και τον λογαριασμό μου ως τον κόμβο προορισμού.
- Οταν σε ακολουθώ στο Twitter, δημιουργώ μια δεύτερη ακμή, με τον λογαριασμό μου σαν κόμβο προέλευσης και τον δικό σου σαν κόμβο προορισμού.

Κεφάλαιο 4

Τεχνικές σχεδίασης αλγορίθμων

4.1 Ανάλυση προβλημάτων

Ερώτηση 4.1

Ποιος είναι ο λόγος ανάλυσης ενός προβλήματος;

Ο αλγόριθμος αποσκοπεί στην επίλυση ενός προβλήματος. Είναι πιθανόν ένα πρόβλημα να μην επιλύεται με μία μόνο λύση αλλά με περισσότερες. Γενικά, η λύση σε ένα πρόβλημα μπορεί να προέλθει από ποικίλες και διαφορετικές προσεγγίσεις, τεχνικές και μεθόδους. Ετσι, είναι απαραίτητο να γίνεται μία καλή ανάλυση του κάθε προβλήματος και να προτείνεται συγκεκριμένη μεθοδολογία και ακολουθία βημάτων. Βασικός στόχος μας είναι η πρόταση έξυπνων και αποδοτικών λύσεων.

Ερώτηση 4.2

Ποια στάδια περιλαμβάνει η ανάλυση ενός προβλήματος;

Η ανάλυση ενός προβλήματος σε ένα σύγχρονο υπολογιστικό περιβάλλον περιλαμβάνει:

- την καταγραφή της υπάρχουσας πληροφορίας για το πρόβλημα,
- την αναγνώριση των ιδιαιτεροτήτων του προβλήματος,
- την αποτύπωση των συνθηκών και προϋποθέσεων υλοποίησής του
- την πρόταση επίλυσης με χρήση κάποιας μεθόδου, και
- την τελική επίλυση με χρήση υπολογιστικών συστημάτων.

Ερώτηση 4.3

Κατά την ανάλυση ενός προβλήματος, σε ποια ερωτήματα θα πρέπει να δοθούν απαντήσεις;

- Ποια είναι τα δεδομένα και το μέγεθος του προβλήματος,

- Ποιες είναι οι συνθήκες που πρέπει να πληρούνται για την επίλυση του προβλήματος,
- Ποια είναι η πλέον αποδοτική μέθοδος επίλυσής τους (σχεδίαση αλγορίθμου),
- Πώς θα καταγραφεί η λύση σε ένα πρόβλημα (π.χ. σε ψευδογλώσσα), και
- Ποιος είναι ο τρόπος υλοποίησης στο συγκεκριμένο υπολογιστικό σύστημα (π.χ. επιλογή γλώσσας προγραμματισμού)

Ερώτηση 4.4

Να αναφέρετε τους λόγους για τους οποίους οι μέθοδοι ανάλυσης και επίλυσης των προβλημάτων παρουσιάζουν ιδιαίτερο ενδιαφέρον.

Γενικότερα, οι μέθοδοι ανάλυσης και επίλυσης των προβλημάτων παρουσιάζουν ιδιαίτερο ενδιαφέρον για τους εξής λόγους:

- παρέχουν ένα γενικό πρότυπο κατάλληλο για την επίλυση προβλημάτων ευρείας κλίμακας,
- μπορούν να αναπαρασταθούν με κοινές δομές δεδομένων και ελέγχου (που υποστηρίζονται από τις περισσότερες σύγχρονες γλώσσες προγραμματισμού),
- παρέχουν τη δυνατότητα καταγραφής των χρονικών και "χωρικών" απαιτήσεων της μεθόδου επίλυσης, έτσι ώστε να μπορεί να γίνει επακριβής εκτίμηση των αποτελεσμάτων.

4.2 Μέθοδος Διαίρει και Βασίλευε

Ερώτηση 4.5

Να αναφέρετε τυποποιημένες κατηγορίες τεχνικών για την επίλυση ενός συγκεκριμένου προβλήματος,

Υπάρχουν αρκετές τυποποιημένες κατηγορίες τεχνικών, όμως στα πλαίσια του βιβλίου αυτού θα εξετάσουμε μερικές μόνο από αυτές:

- Μέθοδος Διαίρει και Βασίλευε
- Μέθοδος Δυναμικού προγραμματισμού
- Απληστη μέθοδος

Ερώτηση 4.6

Τι γνωρίζεται για την μέθοδο «Διαίρει και Βασίλευε» (divide and conquer)

Η «Διαίρει και Βασίλευε» (divide and conquer) αποτελεί μια μέθοδο σχεδίασης αλγορίθμων στην οποία εντάσσονται οι τεχνικές που υποδιαιρούν ένα πρόβλημα σε μικρότερα υποπροβλήματα, που έχουν την ίδια τυποποίηση με το αρχικό πρόβλημα, αλλά είναι μικρότερα σε μέγεθος. Με όμοιο τρόπο, τα υποπροβλήματα αυτά μπορούν να διαιρεθούν σε ακόμη μικρότερα υποπροβλήματα κ.ο.κ. Ετσι η επίλυση ενός προβλήματος έγκειται στη σταδιακή επίλυση των όσο το δυνατόν μικρότερων υποπροβλημάτων, ώστε τελικά να προκύψει η συνολική λύση του αρχικού ευρύτερου προβλήματος. Η προσέγγιση αυτή ονομάζεται «από πάνω προς τα κάτω» (top-down).

Ερώτηση 4.7

Ποια τα βήματα που ακολοθούμε στη μέθοδο σχεδίασης αλγορίθμων «Διαίρει και Βασίλευε»

Η μέθοδος σχεδίασης αλγορίθμων «Διαίρει και Βασίλευε» μπορεί να αποδοθεί με τα επόμενα βήματα:

- Δίνεται για επίλυση ένα στιγμιότυπο ενός προβλήματος.
- Το στιγμιότυπο του προβλήματος υποδιαιρείται σε υπο-στιγμιότυπα του ίδιου προβλήματος.
- Δίνεται ανεξάρτητη λύση σε κάθε ένα υπο-στιγμιότυπο.
- Συνδυάζονται όλες οι μερικές λύσεις που βρέθηκαν για τα υπο-στιγμιότυπα, έτσι ώστε να δοθεί η συνολική λύση του προβλήματος.

Ερώτηση 4.8

Να δώσετε παραδείγματα στα οποία εφαρμόζεται η μέθοδος «Διαίρει και Βασίλευε»

- Μάντεψε τον αριθμό: Σκεφθείτε έναν αριθμό από το 1 έως το 100 και ζητήστε από τον συμμαθητή ή τη συμμαθήτριά σας (παίκτης-1) να μαντέψει τον αριθμό αυτόν με όσο το δυνατόν λιγότερες προσπάθειες.
- Αναζήτηση ονόματος σε τηλεφωνικό κατάλογο
- «Δυαδική αναζήτηση», η οποία εφαρμόζεται μόνο στην περίπτωση ταξινομημένου συνόλου στοιχείων (π.χ. οι αριθμοί από το 1 έως το 100, τα στοιχεία ενός ταξινομημένου μονοδιάστατου πίνακα κ.ά.)
- Η μέθοδος της διχοτόμησης (ή του Bolzano): χρησιμοποιείται για την εύρεση μιας ρίζας της εξίσωσης f(x)=0 στο διάστημα [a, b]

Ερώτηση 4.9

Ποιος είναι ο μέγιστος αριθμός συγκρίσεων (επαναλήψεων) στη μέθοδο «Διαίρει και Βασίλευε»

Στο πλαίσιο του μαθήματος η υλοποίηση της μεθόδου «Διαίρει και Βασίλευε» γίνεται με την επαναληπτική προσέγγιση (με διαδοχικές επαναλήψεις). Ο μέγιστος αριθμός των συγκρίσεων (επαναλήψεων) που απαιτούνται για την εύρεση ενός στοιχείου σε ένα σύνολο «n» ταξινομημένων στοιχείων, συμπεριλαμβανομένης και της περίπτωσης μη ύπαρξης του στοιχείου, δίνεται από το ακέραιο μέρος του $[log_2(n)+1]$ (με στρογγυλοποίηση προς τα κάτω), η απόδειξη του οποίου υπερβαίνει τα όρια της διδακτέας ύλης του μαθήματος. Επομένως, για την εύρεση του μέγιστου πλήθους των επαναλήψεων θεωρείται γνωστό το $log_2(n)$.

Για παράδειγμα, σε ένα σύνολο 100 ταξινομημένων στοιχείων (n=100), ο μέγιστος αριθμός συγκρίσεων (επαναλήψεων) είναι: $[log_2(100) + 1] = [6, 643856 + 1] = [7, 643856] = 7$.

Κεφάλαιο 5

Ανάλυση αλγορίθμων

Είναι εκτός ύλης για το σχολικό έτος 2022-2023.

Κεφάλαιο 6

Εισαγωγή στον Προγραμματισμό

6.1 Η έννοια του προγράμματος

Ερώτηση 6.1

Ποια είναι τα στάδια για την επίλυση ενός προβλήματος με τον υπολογιστή;

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

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

Ερώτηση 6.2

Πως γίνεται η διατύπωση ενός 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 αλγόριθμοι και οι δομές δεδομένων είναι μια αδιάσπαστη ενότητα.

6.2 Φυσικές και τεχνητές γλώσσες

Ερώτηση 6.3

Τι χαρακτηρίζουμε ως φυσικές και τι ως 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, πού είναι τεχνητές γλώσσες, ακολουθούν τις βασικές έννοιες και αρχές της γλωσσολογίας, επιστήμη που μελετά τις φυσικές γλώσσες.

Ερώτηση 6.4

Από τι προσδιορίζεται μία γλώσσα (φυσική ή τεχνητή);

- **ΤΟ ΑΛΦΑΒΗΤΟ**

  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 (A-Z και ά-ζ) καθώς και τα ψηφία και όλα τα σημεία στίξης που χρησιμοποιούνται.

- **ΤΟ ΛΕΞΙΛΟΓΙΟ**

  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.

- **Η ΓΡΑΜΜΑΤΙΚΗ**

  Η Γραμματική αποτελείται από το τυπικό ή τυπολογικό και το συντακτικό. 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 καθορίζει τη νομιμότητα της διάταξης και της σύνδεσης των λέξεων της γλώσσας για τη δημιουργία προτάσεων.

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

- **Η ΣΗΜΑΣΙΟΛΟΓΙΑ**

  Η σημασιολογία είναι το σύνολο των κανόνων που καθορίζει το νόημα των λέξεων και κατά επέκταση των εκφράσεων και προτάσεων που χρησιμοποιούνται σε μία γλώσσα.

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

Ερώτηση 6.5

Σε τι διαφέρουν οι φυσικές από τις τεχνητές γλώσσες;

Μία βασική διαφορά μεταξύ φυσικών και τεχνητών γλωσσών είναι η δυνατότητα εξέλιξης τους. Οι φυσικές γλώσσες εξελίσσονται συνεχώς, νέες λέξεις δημιουργούνται, 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 βελτιώνονται 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 προγραμματισμού αλλάζουν:

- σε επίπεδο διαλέκτου (για παράδειγμα GW-Basic και QBasic) ή
- σε επίπεδο επέκτασης (για παράδειγμα Basic και Visual Basic).

6.3 Τεχνικές σχεδίασης προγραμμάτων

Ερώτηση 6.6

Τι ονομάζουμε ιεραρχική σχεδίαση προγράμματος;

Η τεχνική της ιεραρχικής σχεδίασης και επίλυσης ή η διαδικασία σχεδίασης ”από επάνω προς τα κάτω” όπως 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 απλούστερα υποπροβλήματα, τα οποία να είναι εύκολο να επιλυθούν οδηγώντας στην επίλυση του αρχικού προβλήματος.

Η ιεραρχική σχεδίαση ή ιεραρχικός προγραμματισμός χρησιμοποιεί τη στρατηγική της συνεχούς διαίρεσης του προβλήματος σε υποπροβλήματα.

Για την υποβοήθηση της ιεραρχικής σχεδίασης χρησιμοποιούνται διάφορες διαγραμματικές τεχνικές.

Ερώτηση 6.7

Τι είναι ο τμηματικός προγραμματισμός; Ποια είναι τα πλεονεκτήματα του;

Η ιεραρχική σχεδίαση προγράμματος υλοποιείται με τον τμηματικό προγραμματισμό. Μετά την ανάλυση του προβλήματος σε αντίστοιχα υποπροβλήματα, κάθε υποπρόβλημα αποτελεί **ανεξάρτητη ενότητα**, 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.

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

Ερώτηση 6.8

Τι ξέρετε για την εντολή $GOTO$;

Στην ιστορία του προγραμματισμού καμία άλλη εντολή δεν συζητήθηκε τόσο πολύ όσο η εντολή $GOTO$ (πήγαινε). Η εντολή $GOTO$ έχει ως αποτέλεσμα την αλλαγή της ροής του προγράμματος, της διακλάδωσης σε μία άλλη εντολή του προγράμματος εκτός από την επόμενη.

Η εντολή αυτή χώρισε τους προγραμματιστές σε δύο αντιμαχόμενες 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 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 $GOTO$.

Ολες 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.

Ερώτηση 6.9

Τι είναι ο δομημένος προγραμματισμός; Που στηρίζεται ο δομημένος προγραμματισμός;

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. Ολα 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.

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

Ερώτηση 6.10

Ποια είναι τα πλεονεκτήματα του δομημένου προγραμματισμού;

1. Δημιουργία απλούστερων προγραμμάτων.
2. Αμεση 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.

6.4 Αντικειμενοστραφής προγραμματισμός

Ερώτηση 6.11

Ποια είναι πρωτεύοντα δομικά στοιχεία ενός προγράμματος στην αντικειμενοστραφή σχεδίαση;

Η αντικειμενοστραφής σχεδίαση εκλαμβάνει ως πρωτεύοντα δομικά στοιχεία ενός προγράμματος τα δεδομένα, από τα οποία δημιουργούνται με κατάλληλη μορφοποίηση τα αντικείμενα (objects).

Ερώτηση 6.12

Ποια τα πλεονεκτήματα της αντικειμενοστραφούς σχεδίασης;

Η αντικειμενοστραφής σχεδίαση αποδείχθηκε ότι επιφέρει καλύτερα αποτελέσματα, αφού τα προγράμματα που δημιουργούνται είναι περισσότερο ευέλικτα και επαναχρησιμοποιήσιμα.

Ερώτηση 6.13

Ο αντικειμενοστραφής προγραμματισμός χρησιμοποιεί τεχνικές του κλασσικού διαδικασιακού προγραμματισμού;

Ο αντικειμενοστραφής προγραμματισμός εκτός από τον τρόπο που χειρίζεται τα δεδομένα, συνεχίζει να χρησιμοποιεί την ιεραρχική σχεδίαση, τον τμηματικό προγραμματισμό και να ακολουθεί τις αρχές του δομημένου προγραμματισμού.

6.5 Προγραμματιστικά περιβάλλοντα

Ερώτηση 6.14

Ποια η διαφορά λογικών και συντακτικών λαθών;

Τα *λογικά* λάθη που είναι τα πλέον σοβαρά και δύσκολα στη διόρθωση 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.

Τα λογικά λάθη εμφανίζονται μόνο στην εκτέλεση, ενώ τα συντακτικά λάθη στο στάδιο της μεταγλώττισης.

Ερώτηση 6.15

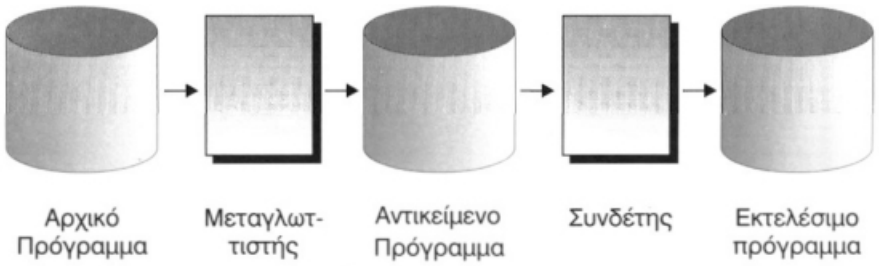
Ποια η διαδικασία για την μετάφραση και εκτέλεση ενός προγράμματος;

Το αρχικό πρόγραμμα λέγεται πηγαίο πρόγραμμα, ενώ το πρόγραμμα που παράγεται από το μεταγλωττιστή λέγεται αντικείμενο πρόγραμμα Το αντικείμενο πρόγραμμα είναι μεν σε μορφή κατανοητή από τον υπολογιστή, αλλά συνήθως δεν είναι σε θέση να εκτελεστεί.

Χρειάζεται να συμπληρωθεί και να συνδεθεί με άλλα τμήματα προγράμματος απαραίτητα για την εκτέλεση του, τμήματα που είτε τα γράφει ο προγραμματιστής είτε βρίσκονται στις βιβλιοθήκες της γλώσσας.

Το πρόγραμμα που επιτρέπει αυτή τη σύνδεση ονομάζεται συνδέτης - φορτωτής. Το αποτέλεσμα του συνδέτη είναι η παραγωγή του εκτελέσιμου προγράμματος, το οποίο είναι το τελικό πρόγραμμα που εκτελείται από τον υπολογιστή. Για το λόγο αυτό η συνολική διαδικασία αποκαλείται μεταγλώττιση και σύνδεση.

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



Ερώτηση 6.16

Τι είναι ο μεταγλωττιστής και τι ο διερμηνευτής; Ποιες οι διαφορές μεταγλωττιστή και διερμηνευτή;

Κάθε πρόγραμμα που γράφτηκε σε οποιαδήποτε γλώσσα προγραμματισμού, πρέπει να μετατραπεί σε μορφή αναγνωρίσιμη και εκτελέσιμη από τον υπολογιστή, δηλαδή σε εντολές γλώσσας μηχανής.

Η μετατροπή αυτή επιτυγχάνεται με τη χρήση ειδικών μεταφραστικών προγραμμάτων. Υπάρχουν δύο μεγάλες κατηγορίες τέτοιων προγραμμάτων,

- οι μεταγλωττιστές και

- οι διερμηνευτές.

Ο μεταγλωττιστής δέχεται στην είσοδο ένα πρόγραμμα γραμμένο σε μια γλώσσα υψηλού επιπέδου και παράγει 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.

Η δημιουργία του εκτελέσιμου προγράμματος γίνεται μόνο στην περίπτωση, που το αρχικό πρόγραμμα δεν περιέχει λάθη.

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 υποβάλλεται πάλι 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. Ομως η εκτέλεση του προγράμματος 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 δημιουργίας του προγράμματος και μεταγλωττιστής για την τελική έκδοση και εκμετάλλευση του προγράμματος.

Ερώτηση 6.17

Τι ονομάζεται συντάκτης προγραμμάτων;

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

Ερώτηση 6.18

Ποια προγράμματα και εργαλεία περιέχει ένα προγραμματιστικό περιβάλλον;

Για τη δημιουργία, τη μετάφραση και την εκτέλεση ενός προγράμματος απαιτούνται τουλάχιστον τρία προγράμματα:

- ο συντάκτης,
- ο μεταγλωττιστής και
- ο συνδέτης.

Τα σύγχρονα προγραμματιστικά περιβάλλοντα παρέχουν αυτά τα προγράμματα με ενιαίο τρόπο.

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

Κεφάλαιο 7

Βασικά στοιχεία προγραμματισμού

7.1 Το αλφάβητο της ΓΛΩΣΣΑΣ

Ερώτηση 7.1

Ποιο είναι το αλφάβητο της γλώσσας;

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

- **Γράμματα**
  - Κεφαλαία ελληνικού αλφαβήτου (Α-Ω)
  - Πεζά ελληνικού αλφαβήτου (α-ω)
  - Κεφαλαία λατινικού αλφαβήτου (A-Z)
  - Πεζά λατινικού αλφαβήτου (a-z)
- **Ψηφία**
  0-9
- **Ειδικοί χαρακτήρες**
  + - * / = () . , ’ ! & κενός_ χαρακτήρας ^

7.2 Τύποι δεδομένων

Ερώτηση 7.2

Να αναφέρετε τους τύπους δεδομένων της γλώσσας.

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.

1. **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.

2. **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.

3. **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'. 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.

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

Ερώτηση 7.3

Γιατί υπάρχουν διαφορετικοί τύποι δεδομένων;

Στην πραγματικότητα τα δεδομένα καταχωρούνται στη μνήμη του υπολογιστή καταλαμβάνοντας συγκεκριμένο αριθμό θέσεων (bytes). Ανάλογα με τον τύπο του δεδομένου και το διατιθέμενο αριθμό bytes ποικίλει και το εύρος τιμών που μπορούν να λάβουν.

Ετσι στον υπολογιστή διαθέτουμε 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) οε 4 ή 8 bytes.

7.3 Σταθερές

Ερώτηση 7.4

Τι ονομάζουμε 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.

Ερώτηση 7.5

Πως γίνεται η αντιστοίχιση σταθερών τιμών στη ΓΛΩΣΣΑ;

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

Αποδίδει ονόματα σε σταθερές τιμές. Κάθε ένα από αυτά τα ονόματα μπορεί να χρησιμοποιηθεί οπουδήποτε 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.

ΣΤΑΘΕΡΕΣ
Ονομα-1 = σταθερή-τιμή-1
Ονομα-2 = σταθερή-τιμή-2
.....
Ονομα-ν = σταθερή-τιμή-ν

Ερώτηση 7.6

Να δώσετε 3 παραδείγματα δήλωσης σταθερών.

ΣΤΑΘΕΡΕΣ
ΠΙ = 3.14159
ΦΠΑ = 0.19
ΟΝΟΜΑ = 'ΚΩΣΤΑΣ'

Ερώτηση 7.7

Τι κανόνες ισχύουν για τα ονόματα που χρησιμοποιούμε στην ΓΛΩΣΣΑ;

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, A-Z), 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, Ονομα, Τιμή, Τυπική Απόκλιση, Α100, ΦΠΑ, μέγιστο, Υπολογισμός Ταχύτητας.

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

7.4 Μεταβλητές

Ερώτηση 7.8

Τι ονομάζουμε μεταβλητή;

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

Οι μεταβλητές που χρησιμοποιούνται σε ένα πρόγραμμα, 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 τη 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 μεταβλητής και ως περιεχόμενο εσωτερικά, την τιμή που έχει εκείνη τη συγκεκριμένη στιγμή η μεταβλητή.

Ερώτηση 7.9

Που και πως δηλώνουμε τις μεταβλητές; (Να δώσετε παραδείγματα)

Η ΓΛΩΣΣΑ επιτρέπει τη χρήση μεταβλητών των τεσσάρων 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. Συνιστάται τα ονόματα των μεταβλητών και των σταθερών να ανάγουν στο περιεχόμενο τους.

ΜΕΤΑΒΛΗΤΕΣ
τύπος -1: Λίστα μεταβλητών 1
τύπος -2: Λίστα μεταβλητών 2
.....
τύπος -ν: Λίστα μεταβλητών ν

Ακολουθούν παραδείγματα:

ΜΕΤΑΒΛΗΤΕΣ
ΠΡΑΓΜΑΤΙΚΕΣ: Εμβαδόν, Α
ΑΚΕΡΑΙΕΣ: ΤΙΜΗ, Ν
ΧΑΡΑΚΤΗΡΕΣ: Ονομα
ΛΟΓΙΚΕΣ: Ελεγχος, flag

Αλλάζει η τιμή μιας μεταβλητής κατά τη διάρκεια εκτέλεσης ενός προγράμματος; Αλλάζει ο τύπος μιας μεταβλητής κατά τη διάρκεια εκτέλεσης ενός προγράμματος;

Ερώτηση 7.10

Ποιες ιδιότητες ισχύουν για τις μεταβλητές;

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

7.5 Αριθμητικοί τελεστές

Ερώτηση 7.11

Να αναφέρετε τους αριθμητικούς 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.

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.

Οι τελεστές και οι αντίστοιχες πράξεις είναι:

Τελεστής	Πράξη
+	Πρόσθεση
-	Αφαίρεση
*	Πολλαπλασιασμός
/	Διαίρεση
^	Ύψωση σε δύναμη
div	Ακέραια διαίρεση
mod	Υπόλοιπο ακέραιας διαίρεσης

7.6 Συναρτήσεις

Ερώτηση 7.12

Να αναφέρετε τις συναρτήσεις της ΓΛΩΣΣΑΣ

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

1. **ΗΜ(Χ)** Υπολογισμός ημίτονου
2. **ΣΥΝ(Χ)** Υπολογισμός συνημίτονου
3. **ΕΦ(Χ)** Υπολογισμός εφαπτομένης
4. **Τ_Ρ(Χ)** Υπολογισμός τετραγωνικής ρίζας
5. **ΛΟΓ(Χ**) Υπολογισμός φυσικού λογαρίθμου
6. **Ε(Χ)** Υπολογισμός του ex
7. **Α_Μ(Χ)** Ακέραιο μέρος του Χ
8. **Α_Τ(Χ)** Απόλυτη τιμή του Χ

7.7 Αριθμητικές εκφράσεις

Ερώτηση 7.13

Τι γνωρίζετε για τις αριθμητικές εκφράσεις;

Οταν 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.

Ερώτηση 7.14

Ποια η σειρά εκτέλεσης των πράξεων;

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

1. Υψωση σε δύναμη
2. Πολλαπλασιασμός και διαίρεση
3. Πρόσθεση και αφαίρεση

Οταν 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. (Πάντα πρέπει να χρησιμοποιούνται ζεύγη παρενθέσεων. Διαφορετικός αριθμός αριστερών από δεξιές παρενθέσεις στην ίδια έκφραση είναι ένα από τα πιο συνηθισμένα λάθη.)

Η πράξη που πρέπει να προηγηθεί περικλείεται σε ένα ζεύγος παρενθέσεων, οπότε και εκτελείται πρώτη. Για παράδειγμα, 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.

7.8 Εντολή εκχώρησης

Ερώτηση 7.15

Τι ονομάζουμε εντολή εκχώρησης; Πως λειτουργεί μια εντολή εκχώρησης;

Η εντολή εκχώρησης χρησιμοποιείται για την απόδοση τιμών στις μεταβλητές κατά τη διάρκεια εκτέλεσης του προγράμματος.

Σύνταξη:

όνομα_ μεταβλητής ← έκφραση

Παραδείγματα:
Α← 132
Μήνας ← 'Ιανουάριος'
Αθροισμα← x+y

Η λειτουργία μιας εντολής εκχώρησης είναι οι εξής: Υπολογίζεται η τιμή της έκφρασης στη δεξιά πλευρά και εκχωρείται η τιμή αυτή στη μεταβλητή, που αναφέρεται στην αριστερή πλευρά.

Προσοχή: Σε μια εντολή εκχώρησης η μεταβλητή και η έκφραση πρέπει να είναι του ιδίου τύπου.

Ερώτηση 7.16

Είναι σωστό να θεωρήσουμε μια εντολή εκχώρησης ως εξίσωση; Να δικαιολογήσετε την απάντησή σας.

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.

7.9 Εντολές εισόδου-εξόδου

Ερώτηση 7.17

Ποιος ο σκοπός των εντολών εισόδου και εξόδου; Ποιες είναι οι εντολές εισόδου και 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 προγράμματος 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**.

Ερώτηση 7.18

Τι γνωρίζετε για την εντολή ΔΙΑΒΑΣΕ;

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 προγράμματος 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

ΔΙΑΒΑΣΕ λίστα-μεταβλητών

Παραδείγματα

ΔΙΑΒΑΣΕ Ποσότητα, Τιμή

Λειτουργία

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

Ερώτηση 7.19

Τι γνωρίζετε για την εντολή ΓΡΑΨΕ;

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.

Σύνταξη

ΓΡΑΨΕ Λίστα-στοιχείων

Παραδείγματα

ΓΡΑΨΕ 'Η τετραγωνική ρίζα του', Α, ' είναι:',ΡΙΖΑ

Οταν 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

Ερώτηση 7.20

Περιγράψτε τη δομή ενός προγράμματος

Οπως κάθε εντολή ακολουθεί αυστηρούς συντακτικούς κανόνες, έτσι και ολόκληρο το πρόγραμμα έχει αυστηρούς 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.

Στη συνέχεια ακολουθεί το τμήμα δήλωσης των σταθερών του προγράμματος, αν βέβαια το πρόγραμμα μας χρησιμοποιεί σταθερές.

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 ΑΡΧΗ Τέλος αν το πρόγραμμα χρησιμοποιεί διαδικασίες (βλ. κεφ. 10), 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.

Ερώτηση 7.21

Να δώσετε ένα παράδειγμα προγράμματος. Να γίνει το πρόγραμμα ΠΡΟΠΑΙΔΕΙΑ.

```
ΠΡΟΓΡΑΜΜΑ ΠΡΟΠΑΙΔΕΙΑ
    !Πρόγραμμα εκτύπωσης της προπαίδειας των αριθμών 1 έως 10
   ΜΕΤΑΒΛΗΤΕΣ
       ΑΚΕΡΑΙΕΣ:Α,Β,Γ
   !Α:Πολλαπλασιαστέος
   !Β:Πολαπλασιαστής
   !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
       ΤΕΛΟΣ_ ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Κεφάλαιο 8

Επιλογή και επανάληψη

8.1 Εντολές Επιλογής

Ερώτηση 8.1

Πως γίνεται η σύνταξη μιας λογικής έκφρασης; Ποιες είναι οι τιμές που μπορεί να πάρει μια 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.

Ερώτηση 8.2

Ποιοι είναι οι συγκριτικοί τελεστές;

Οι συγκρίσεις γίνονται σε δεδομένα αριθμητικά, αλφαριθμητικά και λογικά.

ΤΕΛΕΣΤΗΣ	ΕΛΕΓΧΟΜΕΝΗ ΣΧΕΣΗ	Παράδειγμα
=	Ισότητα	Αριθμός=0
< >	Ανισότητα	Ονομα1 < > 'Κώστας'
>	Μεγαλύτερο από	Τιμή>10000
>=	Μεγαλύτερο ή ίσο από	Χ+Υ>=(Α+Β)/Γ
<	Μικρότερο από	$Β^2$-4*Α*Γ<0
<=	Μικρότερο ή ίσο από	Βάρος <= 500

Ερώτηση 8.3

Πως γίνεται η σύγκριση δεδομένων;

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.

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

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

8.2 Εντολές επανάληψης

Ερώτηση 8.4

Τι ονομάζουμε δομή 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.

Ερώτηση 8.5

Ποιές είναι οι εντολές επανάληψης;

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.

Ερώτηση 8.6

Τι ονομάζουμε "τιμή φρουρός";

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".

Ερώτηση 8.7

Τι κανόνες ισχύουν στους εμφωλευμένους βρόχους;

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

1. Ο εσωτερικός βρόχος πρέπει να βρίσκεται ολόκληρος μέσα στον εξωτερικό. Ο βρόχος που ξεκινάει τελευταίος, πρέπει να ολοκληρώνεται πρώτος.

2. Η είσοδος σε κάθε βρόχο υποχρεωτικά γίνεται από την αρχή του.

3. Δεν μπορεί να χρησιμοποιηθεί η ίδια μεταβλητή ως μετρητής δύο ή περισσοτέρων βρόχων που ο ένας βρίσκεται στο εσωτερικό του άλλου.

Κεφάλαιο 9

Πίνακες

9.1 Μονοδιάστατοι πίνακες

Ερώτηση 9.1

Τι ονομάζεται πίνακας; Για ποιο λόγο χρησιμοποιούνται οι πίνακες;

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 ακολουθούμενο από ένα δείκτη

Οι πίνακες χρησιμοποιούνται όταν είναι απαραίτητη η διαχείριση ενός πλήθους δεδομένων ίδιου τύπου.

Ερώτηση 9.2

Τι μπορεί να είναι οι δείκτες των πινάκων;

Ο δείκτης είναι μια ακέραια έκφραση, σταθερή ή μεταβλητή που περικλείεται μέσα στα σύμβολα [και]. Η έκφραση αυτή μπορεί να έχει οποιαδήποτε όνομα. Είναι σύνηθες όμως να χρησιμοποιούνται ως δείκτες οι μεταβλητές i, j, k.

Ερώτηση 9.3

Ποια η διαφορά του πίνακα και του στοιχείου ενός πίνακα;

Με τη χρήση του πίνακα, όλα τα δεδομένα καταχωρούνται κάτω από το ίδιο όνομα μεταβλητής. Η ανάγνωση, η επεξεργασία και η εκτύπωση όλων των δεδομένων δεν μπορεί να γίνει χρησιμοποιώντας απλά το όνομα του πίνακα. Για κάθε αναφορά σε ένα στοιχείο του πίνακα απαιτείται η χρήση του ονόματος του πίνακα ακολουθούμενο από έναν ή περισσότερους δείκτες, ανάλογα τη διάσταση του πίνακα.

Για παράδειγμα για να εκτυπωθεί το πρώτο στοιχείο του πίνακα Α, η σωστή εντολή είναι ΓΡΑΨΕ Α[1]. Η ανάγνωση, η επεξεργασία και η εκτύπωση όλων των στοιχείων των πινάκων γίνεται πάντοτε από βρόχους, οι οποίοι επαναλαμβάνονται προκαθορισμένο αριθμό φορών, όσα είναι τα στοιχεία του πίνακα και υλοποιούνται καλύτερα στον προγραμματισμό με την εντολή επανάληψης ΓΙΑ.

Ερώτηση 9.4

Που ορίζεται η διάσταση ενός πίνακα; Δώστε ένα παράδειγμα ενός τρισδιάστατου πίνακα.

Η διάσταση του πίνακα ορίζεται από τον αριθμό των δεικτών του, που χρησιμοποιούνται για τον καθορισμό των στοιχείων του. Για παράδειγμα, ο πίνακα A[i] είναι μονοδιάστατος, ο πίνακας A[i, j] είναι δισδιάστατος και ο πίνακας A[i,j,k] τρισδιάστατος.

Ερώτηση 9.5

Τι είδους δομή είναι ένας πίνακας;

Ο πίνακας είναι μια στατική δομή δεδομένων, δηλαδή το ακριβές μέγεθος της απαιτούμενης δύναμης καθορίζεται από την αρχή και όχι κατά την στιγμή της εκτέλεσης. Επίσης τα στοιχεία του είναι όλα του ίδιου τύπου.

Ερώτηση 9.6

Που αποθηκεύονται τα στοιχεία ενός πίνακα;

Τα στοιχεία του πίνακα αποθηκεύονται σε διαδοχικές θέσεις στη μνήμη.

9.2 Πότε πρέπει να χρησιμοποιούνται πίνακες

Ερώτηση 9.7

Ποια τα μειονεκτήματα της χρήσης ενός πίνακα;

Υπάρχουν δύο μειονεκτήματα με τη χρήση πινάκων:

1. 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.
2. Οι πίνακες περιορίζουν τις δυνατότητες του προγράμματος. 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.

9.3 Τυπικές επεξεργασίες πινάκων

Ερώτηση 9.8

Ποιες είναι οι τυπικές επεξεργασίες ενός πίνακα.

1. **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.

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

   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.

3. **Ταξινόμηση των στοιχείων του πίνακα.**

   Στο κεφάλαιο 3 αναφέρθηκε η μέθοδος ταξινόμησης της ευθείας ανταλλαγής, η οποία 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.

4. **Αναζήτηση ενός στοιχείου του πίνακα.** Δύο είναι οι πλέον διαδεδομένοι αλγόριθμοι αναζήτησης:

   - Η σειριακή αναζήτηση
   - Η δυαδική αναζήτηση

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

5. **Συγχώνευση δύο πινάκων.**

   Η συγχώνευση είναι μία από τις βασικές λειτουργίες σε πίνακες. Σκοπός της είναι η δημιουργία από τα στοιχεία δύο (ή περισσότερων) ταξινομημένων πινάκων ενός άλλου, που είναι και αυτός ταξινομημένος.

Κεφάλαιο 10

Υποπρογράμματα

10.1 Τμηματικός προγραμματισμός

Ερώτηση 10.1

Τι είναι τμηματικός προγραμματισμός;

Τμηματικός προγραμματισμός ονομάζεται η τεχνική σχεδίασης και ανάπτυξης προγραμμάτων ως ένα σύνολο από απλούστερα τμήματα προγραμμάτων Η σωστή διαίρεση είναι μια πολύπλοκη διαδικασία αλλά έχει πολλά πλεονεκτήματα Τα τρία βασικά τμήματα που συναντώνται σε κάθε σχεδόν πρόβλημα είναι (δείτε και το παράδειγμα που ακολουθεί):

i. Εισαγωγή δεδομένων (καταχώρηση, έλεγχος)

ii. Επεξεργασία δεδομένων (πράξεις - υπολογισμοί)

iii. Εξοδος δεδομένων (εκτύπωση ή εμφάνιση)

Ερώτηση 10.2

Τι ονομάζουμε υποπρόγραμμα;

Οταν ένα τμήμα προγράμματος επιτελεί ένα αυτόνομο έργο και έχει γραφεί χωριστά από το υπόλοιπο πρόγραμμα, τότε αναφερόμαστε σε υποπρόγραμμα. (subprogram)

10.2 Χαρακτηριστικά των υποπρογραμμάτων

Ερώτηση 10.3

Ποιες είναι οι ιδιότητες που πρέπει να διακρίνουν τα υποπρογράμματα;

Τα υποπρογράμματα πρέπει να διακρίνονται από 3 βασικές αρχές :

i. Κάθε υποπρόγραμμα έχει μία είσοδο και μία έξοδο. Το υποπρόγραμμα δέχεται κάποιες τιμές-παραμέτρους, πραγματοποιεί κάποιες ενέργειες-υπολογισμούς και επιστρέφει μία ή περισσότερες τιμές κατά τον τερματισμό του

ii. Κάθε υποπρόγραμμα πρέπει να είναι όσο το δυνατόν ανεξάρτητο από τα άλλα. Να είναι διαφανής ο τρόπος που λειτουργεί. Σε ιδεατό επίπεδο δέχεται κάποιες τιμές - παραμέτρους και επιστρέφει αποτέλεσμα – ενέργειες

iii. Κάθε υποπρόγραμμα πρέπει να μην είναι πολύ μεγάλο. Αυτό είναι υποκειμενικό αλλά γενικά μπορούμε να πούμε πως κάθε πρόγραμμα πρέπει να επιτελεί συγκεκριμένη ενέργεια - υπολογισμό και να τερματίζεται (να επιστρέφει τον έλεγχο στο πρόγραμμα ή υποπρόγραμμα που το κάλεσε)

10.3 Πλεονεκτήματα του τμηματικού προγραμματισμού

Ερώτηση 10.4

Ποια τα πλεονεκτήματα του τμηματικού προγραμματισμού;

i. **Διευκολύνει την ανάπτυξη του αλγορίθμου και του αντιστοίχου προγράμματος.** Επιτρέπει την εξέταση και την επίλυση απλών προβλημάτων και όχι στην αντιμετώπιση του συνολικού προβλήματος. Με τη σταδιακή επίλυση των υποπροβλημάτων και τη δημιουργία των αντιστοίχων υποπρογραμμάτων τελικά επιλύεται το συνολικό πρόβλημα

ii. **Διευκολύνει την κατανόηση και διόρθωση του προγράμματος.** Ο χωρισμός του προγράμματος σε μικρότερα αυτοτελή τμήματα επιτρέπει τη γρήγορη διόρθωση ενός συγκεκριμένου τμήματος του χωρίς οι αλλαγές αυτές να επηρεάσουν όλο το υπόλοιπο πρόγραμμα. Επίσης διευκολύνει οποιονδήποτε χρειαστεί να διαβάσει και να κατανοήσει τον τρόπο που λειτουργεί το πρόγραμμα

iii. **Απαιτεί λιγότερο χρόνο και προσπάθεια στη συγγραφή του προγράμματος.** Πολύ συχνά χρειάζεται η ίδια λειτουργία σε διαφορετικά σημεία ενός προγράμματος. Από τη στιγμή που ένα υποπρόγραμμα έχει γραφεί, μπορεί το ίδιο να καλείται από πολλά σημεία του προγράμματος. Ετσι μειώνονται το μέγεθος του προγράμματος, ο χρόνος που απαιτείται για τη συγγραφή του και οι πιθανότητες λάθους, ενώ ταυτόχρονα το πρόγραμμα γίνεται πιο εύληπτο και κατανοητό

iv. **Επεκτείνει τις δυνατότητες των γλωσσών προγραμματισμού.** Ενα υποπρόγραμμα που έχει γραφεί μπορεί να χρησιμοποιηθεί πολύ εύκολα και σε άλλα προγράμματα. Από τη στιγμή που έχει δημιουργηθεί, η χρήση του δεν διαφέρει από τη χρήση των ενσωματωμένων συναρτήσεων που παρέχει η γλώσσα προγραμματισμού, όπως για τον υπολογισμό του ημίτονου ή του συνημίτονου ή την εντολή με την οποία εκτελεί μία συγκεκριμένη διαδικασία, για παράδειγμα γράφει στην οθόνη (εντολή ΓΡΑΨΕ). Αν λοιπόν χρειάζεται συχνά κάποια λειτουργία που δεν υποστηρίζεται απευθείας από τη γλώσσα, όπως για παράδειγμα η εύρεση του μικρότερου δύο αριθμών, τότε μπορεί να γραφεί το αντίστοιχο υποπρόγραμμα. Η συγγραφή πολλών υποπρογραμμάτων και η δημιουργία βιβλιοθηκών με αυτά, ουσιαστικά επεκτείνουν την ίδια τη γλώσσα προγραμματισμού

10.4 Παράμετροι

Ερώτηση 10.5

Πως γίνεται η ενεργοποίηση των υποπρογραμμάτων;

Τα υποπρογράμματα ενεργοποιούνται από κάποιο άλλο πρόγραμμα ή υποπρόγραμμα για να εκτελέσουν συγκεκριμένες λειτουργίες.

Κάθε υποπρόγραμμα για να ενεργοποιηθεί καλείται, όπως λέγεται, από ένα άλλο υποπρόγραμμα ή το αρχικό πρόγραμμα, το οποίο ονομάζεται κύριο πρόγραμμα.

Ερώτηση 10.6

Τι είναι οι παραμέτροι;

Το υποπρόγραμμα είναι αυτόνομο και ανεξάρτητο τμήμα προγράμματος, αλλά συχνά πρέπει να επικοινωνεί με το υπόλοιπο πρόγραμμα. Συνήθως δέχεται τιμές από το τμήμα προγράμματος που το καλεί και μετά την εκτέλεση επιστρέφει σε αυτό νέες τιμές, αποτελέσματα.

Οι τιμές αυτές που περνούν από το ένα υποπρόγραμμα στο άλλο λέγονται **παράμετροι**.

Οι παράμετροι λοιπόν είναι σαν τις κοινές μεταβλητές ενός προγράμματος με μία ουσιώδη διαφορά, χρησιμοποιούνται για να περνούν τιμές στα υποπρογράμματα.

10.5 Διαδικασίες και συναρτήσεις

Ερώτηση 10.7

Ποια ήδη υποπρογραμμάτων γνωρίζετε; Τι κάνει το καθένα;

Υπάρχουν δυο είδη υποπρογραμμάτων, οι διαδικασίες και οι συναρτήσεις.

Διαδικασία: τύπος υποπρογράμματος που μπορεί να εκτελέσει όλες τις λειτουργίες ενός προγράμματος δηλαδή να διαβάζει δεδομένα, να πραγματοποιεί υπολογισμούς, να τυπώνει μεταβλητές και μηνύματα κ.λ.π.

Συνάρτηση: τύπος υποπρογράμματος δέχεται μια ομάδα τιμών, τις επεξεργάζεται, υπολογίζει και επιστρέφει μια μόνο τιμή

Ερώτηση 10.8

Ποια η δομή της συνάρτησης; Πως καλείται από το κυρίως πρόγραμμα;

ΣΥΝΑΡΤΗΣΗ όνομα (λίστα παραμέτρων):τύπος συνάρτησης
Τμήμα δηλώσεων
ΑΡΧΗ
…….
Ονομα← έκφραση
…….
ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ

Κάθε συνάρτηση εκτελείται, όπως ακριβώς εκτελούνται οι ενσωματωμένες συναρτήσεις της γλώσσας. Απλώς, αναφέρεται το όνομα της σε μια έκφραση ή σε μια εντολή και επιστρέφεται η τιμή της.

Ερώτηση 10.9

Ποια η δομή της διαδικασίας; Πως καλείται από το κυρίως πρόγραμμα;

ΔΙΑΔΙΚΑΣΙΑ όνομα (λίστα παραμέτρων)
Τμήμα δηλώσεων
ΑΡΧΗ
εντολές
ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ

Κάθε διαδικασία εκτελείται όταν καλείται από το κύριο πρόγραμμα ή άλλη διαδικασία. Η κλήση σε διαδικασία γίνεται με την εντολή **ΚΑΛΕΣΕ**, που ακολουθείται από το όνομα της διαδικασίας συνοδευόμενο μέσα σε παρενθέσεις με τη λίστα παραμέτρων.

Η εκτέλεση του προγράμματος διακόπτεται και εκτελούνται οι εντολές της διαδικασίας που καλείται. Μετά το τέλος της διαδικασίας η εκτέλεση του προγράμματος συνεχίζεται από την εντολή που ακολουθεί. Η λίστα των παραμέτρων ορίζει τις τιμές που περνούν στη διαδικασία και τις τιμές που αυτή επιστρέφει. Η λίστα παραμέτρων δεν είναι υποχρεωτική.

Η γενική μορφή της **ΚΑΛΕΣΕ** είναι:
ΚΑΛΕΣΕ όνομα-διαδικασίας (λίστα παραμέτρων)

Ερώτηση 10.10

Ποια η βασική διαφορά των διαδικασιών και συναρτήσεων;

Οι διαδικασίες μπορούν να εκτελέσουν οποιαδήποτε λειτουργία από αυτές που μπορεί να εκτελέσει ένα πρόγραμμα: να εισάγουν δεδομένα, να εκτελέσουν υπολογισμούς, να μεταβάλλουν τις τιμές των μεταβλητών και να τυπώσουν αποτελέσματα. Με τη χρήση των παραμέτρων αυτές τις τιμές μπορούν να τις μεταφέρουν και στα άλλα υποπρογράμματα ή στο κύριο πρόγραμμα.

Αντίθετα η λειτουργία των συναρτήσεων είναι πιο περιορισμένη. Οι συναρτήσεις υπολογίζουν μόνο μια τιμή, αριθμητική, χαρακτήρα ή λογική και μόνο σε αυτήν επιστρέφουν στο υποπρόγραμμα που την κάλεσε. Οι συναρτήσεις μοιάζουν με τις συναρτήσεις των μαθηματικών και η χρήση τους είναι όμοια με τη χρήση των ενσωματωμένων συναρτήσεων που υποστηρίζει η ΓΛΩΣΣΑ.

10.6 Πραγματικές και τυπικές παράμετροι

Ερώτηση 10.11

Τι γνωρίζετε για τις πραγματικές – τυπικές παραμέτρους;

Οι παράμετροι που βρίσκονται στο πρόγραμμα ονομάζονται *πραγματικές* παράμετροι.

Οι παράμετροι που βρίσκονται στο υποπρόγραμμα ονομάζονται *τυπικές* παράμετροι.

Ολες οι μεταβλητές είναι γνωστές, έχουν ισχύ όπως λέγεται, μόνο για το τμήμα προγράμματος στο οποίο έχουν δηλωθεί, ισχύουν δηλαδή τοπικά για το συγκεκριμένο υποπρόγραμμα ή κυρίως πρόγραμμα.

Τα ονόματα των τυπικών και των πραγματικών παραμέτρων μπορούν να είναι οποιαδήποτε. Αφού είναι ονόματα μεταβλητών σε διαφορετικά τμήματα προγράμματος, είναι υποχρεωτικά διαφορετικές μεταβλητές, άσχετα αν έχουν το ίδιο όνομα.

Η λίστα των τυπικών παραμέτρων καθορίζει τις παραμέτρους στη δήλωση του υποπρογράμματος. Η λίστα των πραγματικών παραμέτρων καθορίζει τις παραμέτρους στην κλήση του υποπρογράμματος.

Μερικές γλώσσες προγραμματισμού ονομάζουν ορίσματα τις τυπικές παραμέτρους και απλά παραμέτρους τις πραγματικές παραμέτρους.

Ερώτηση 10.12

Ποιους κανόνες πρέπει να ακολουθούν οι λίστες των παραμέτρων;

i. Ο αριθμός των πραγματικών και των τυπικών παραμέτρων πρέπει να είναι ίδιος.

ii. Κάθε πραγματική παράμετρος αντιστοιχεί στην τυπική παράμετρο που βρίσκεται στην αντίστοιχη θέση. Για παράδειγμα η πρώτη της λίστας των τυπικών παραμέτρων στην πρώτη της λίστας των πραγματικών παραμέτρων κοκ.

iii. Η τυπική παράμετρος και η αντίστοιχη της πραγματική πρέπει να είναι του ιδίου τύπου

Ερώτηση 10.13

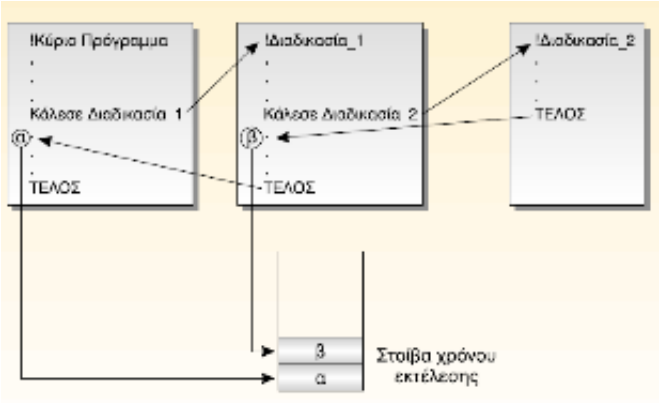
Ποια η χρήση της στοίβας στην κλήση διαδικασιών;

Η έννοια της στοίβας είναι πολύ χρήσιμη στο ίδιο το λογισμικό των γλωσσών προγραμματισμού. Οταν μία διαδικασία ή συνάρτηση καλείται από το κύριο πρόγραμμα, τότε η αμέσως επόμενη διεύθυνση του κύριου προγράμματος, που ονομάζεται **διεύθυνση επιστροφής** αποθηκεύεται από το μεταφραστή σε μία στοίβα που ονομάζεται **στοίβα χρόνου εκτέλεσης**.

Μετά την εκτέλεση της διαδικασίας ή της συνάρτησης η διεύθυνση επιστροφής απωθείται από τη στοίβα και έτσι ο έλεγχος του προγράμματος μεταφέρεται και πάλι στο κύριο πρόγραμμα. Η τεχνική αυτή εφαρμόζεται και γενικότερα, δηλαδή οποτεδήποτε μία διαδικασία ή συνάρτηση καλεί μία διαδικασία ή συνάρτηση.

Για παράδειγμα, έστω ότι μία διαδικασία Α καλεί τη διαδικασία Β, που με τη σειρά της καλεί τη διαδικασία Γ κοκ. Στην περίπτωση αυτή οι διευθύνσεις επιστροφής εμφανίζονται στη στοίβα με σειρά Γ, Β, Α. Μετά την εκτέλεση κάθε διαδικασίας, η διεύθυνση επιστροφής απωθείται από τη στοίβα και ο έλεγχος μεταβιβάζεται στη διεύθυνση αυτή.

Το παράδειγμα αυτό δείχνει μία από τις πολλές χρησιμότητες της LIFo ιδιότητας της στοίβας.



Σχ. 10.3. Χρήση στοίβας από το μεταφραστή για το χειρισμό κλήσεων διαδικασιών και επιστροφών από αυτές.

10.7 Εμβέλεια μεταβλητών-σταθερών.

Ερώτηση 10.14

Ποιές είναι οι τοπικές μεταβλητές;

Κάθε κύριο πρόγραμμα όπως και κάθε υποπρόγραμμα περιλαμβάνει τις δικές του μεταβλητές και σταθερές.

Οι μεταβλητές αυτές στη ΓΛΩΣΣΑ είναι γνωστές στο αντίστοιχο υποπρόγραμμα που δηλώνονται και μόνο σε αυτό. Ολες οι μεταβλητές (και οι σταθερές) είναι *τοπικές* στο συγκεκριμένο τμήμα προγράμματος.

Ερώτηση 10.15

Τι ονομάζουμε εμβέλεια των μεταβλητών και των σταθερών;

Πολλές γλώσσες προγραμματισμού επιτρέπουν τη χρήση των μεταβλητών και των σταθερών, όχι μόνο στο τμήμα προγράμματος που δηλώνονται, αλλά και σε άλλα ή ακόμη και σε όλα τα υπόλοιπα υποπρογράμματα. Αυτό που καθορίζει την περιοχή που ισχύουν οι μεταβλητές και οι σταθερές είναι η εμβέλεια των μεταβλητών της γλώσσας.

Ερώτηση 10.16

Τι σημαίνει ο όρος ”απεριόριστή εμβέλεια” μεταβλητών; Τι μειονέκτημα έχει;

Σύμφωνα με αυτή την αρχή όλες οι μεταβλητές και όλες οι σταθερές είναι γνωστές και μπορούν να χρησιμοποιούνται σε οποιοδήποτε τμήμα του προγράμματος, άσχετα που δηλώθηκαν. Ολες οι μεταβλητές είναι **καθολικές**.

Η απεριόριστη εμβέλεια καταστρατηγεί την αρχή της αυτονομίας των υποπρογραμμάτων, δημιουργεί πολλά προβλήματα και τελικά είναι αδύνατη για μεγάλα προγράμματα με πολλά υποπρογράμματα, αφού ο καθένας που γράφει κάποιο υποπρόγραμμα πρέπει να γνωρίζει τα ονόματα όλων των μεταβλητών που χρησιμοποιούνται στα υπόλοιπα υποπρογράμματα.

Ερώτηση 10.17

Τι σημαίνει ο όρος ”περιορισμένη εμβέλεια” μεταβλητών; Τι πλεονέκτημα έχει;

Η περιορισμένη εμβέλεια υποχρεώνει όλες τις μεταβλητές που χρησιμοποιούνται σε ένα τμήμα προγράμματος, να δηλώνονται σε αυτό το τμήμα. Ολες οι μεταβλητές είναι τοπικές, ισχύουν δηλαδή για το υποπρόγραμμα στο οποίο δηλώθηκαν. Στη ΓΛΩΣΣΑ έχουμε περιορισμένη εμβέλεια.

Τα πλεονεκτήματα της περιορισμένης εμβέλειας είναι η απόλυτη αυτονομία όλων των υποπρογραμμάτων και η δυνατότητα να χρησιμοποιείται οποιοδήποτε όνομα, χωρίς να ενδιαφέρει αν το ίδιο χρησιμοποιείται σε άλλο υποπρόγραμμα.

Ερώτηση 10.18

Τι σημαίνει ο όρος ”μερικώς περιορισμένη εμβέλεια” μεταβλητών; Τι πλεονέκτημα και τι μειονέκτημα έχει;

Σύμφωνα με αυτή την αρχή άλλες μεταβλητές είναι τοπικές και άλλες καθολικές. Κάθε γλώσσα προγραμματισμού έχει τους δικούς της κανόνες και μηχανισμούς για τον τρόπο και τις προϋποθέσεις που ορίζονται οι μεταβλητές ως τοπικές ή καθολικές.

Η μερικώς περιορισμένη εμβέλεια προσφέρει μερικά πλεονεκτήματα στον πεπειραμένο προγραμματιστή, αλλά για τον αρχάριο περιπλέκει το πρόγραμμα δυσκολεύοντας την ανάπτυξή του.

Κεφάλαιο 11

Σύγχρονα προγραμματιστικά περιβάλλοντα

Ερώτηση 11.1

Τι ονομάζεται Αντικειμενοστραφής προγραμματισμός και τι τα Αντικείμενα;

Αντικειμενοστραφής προγραμματισμός (object-oriented programming) ή αντικειμενοστραφής σχεδίαση είναι μια μεθοδολογία ανάπτυξης εφαρμογών η οποία στηρίζεται σε αυτόνομες προγραμματιστικές οντότητες με δική τους ταυτότητα και συμπεριφορά.

Οι οντότητες αυτές καλούνται αντικείμενα (objects), αντιστοιχούν σε φυσικές οντότητες ή έννοιες του φυσικού μας κόσμου, και δομούνται με βάση δεδομένα (ιδιότητες) που προσδιορίζουν την υπόστασή τους και ενέργειες (κανόνες συμπεριφοράς) που εφαρμόζονται πάνω στα δεδομένα.

Σε μια εφαρμογή, ένα αντικείμενο είναι ο ομαδοποιημένος συνδυασμός δεδομένων και κώδικα, τα οποία έχουμε τη δυνατότητα να χειριστούμε ενιαία.

Ερώτηση 11.2

Ποια είναι πρωτεύοντα δομικά στοιχεία ενός προγράμματος στην αντικειμενοστραφή σχεδίαση;

Η αντικειμενοστραφής σχεδίαση λοιπόν εκλαμβάνει σαν πρωτεύοντα δομικά στοιχεία ενός προγράμματος τα δεδομένα, από τα οποία δημιουργούνται με κατάλληλη μορφοποίηση τα αντικείμενα.

Αυτή η σχεδίαση αποδείχθηκε ότι επιφέρει καλύτερα αποτελέσματα, αφού τα προγράμματα που δημιουργούνται είναι περισσότερο ευέλικτα, επαναχρησιμοποιήσιμα και περισσότερο φιλικά.

Ερώτηση 11.3

Ποια τα πλεονεκτήματα της αντικειμενοστραφούς σχεδίασης;

Η αντικειμενοστραφής σχεδίαση αποδείχθηκε ότι επιφέρει καλύτερα αποτελέσματα, αφού τα προγράμματα που δημιουργούνται είναι περισσότερο ευέλικτα και επαναχρησιμοποιήσιμα.

Ερώτηση 11.4

Ο αντικειμενοστραφής προγραμματισμός χρησιμοποιεί τεχνικές του κλασσικού διαδικασιακού προγραμματισμού;

Ο αντικειμενοστραφής προγραμματισμός εκτός από τον τρόπο που χειρίζεται τα δεδομένα, συνεχίζει να χρησιμοποιεί την ιεραρχική σχεδίαση, τον τμηματικό προγραμματισμό και να ακολουθεί τις αρχές του δομημένου προγραμματισμού.

Ερώτηση 11.5

Τι ονομάζουμε ιδιότητες και τι μεθόδους στα αντικείμενα;

Τα δεδομένα αποτελούν τα χαρακτηριστικά ενός αντικειμένου και αναφέρονται ως ιδιότητες (properties) ενώ οι ενέργειες καθορίζουν τη συμπεριφορά του. Οι ενέργειες στον αντικειμενοστραφή προγραμματισμό αναφέρονται και ως μέθοδοι (methods).

Ερώτηση 11.6

Ποια είναι η βασική αντίληψη στην αντικειμενοστραφή προσέγγιση για την επίλυση προβλημάτων;

Σύμφωνα με την αντικειμενοστραφή θεωρία ανάπτυξης εφαρμογών, η προσέγγιση κάθε προβλήματος πρέπει να γίνεται με φυσική ερμηνεία και να μη στηρίζεται σε πολύπλοκα τεχνικά ζητήματα.

Η επίλυση ενός προβλήματος επιτυγχάνεται με τη σύνθεση ικανοτήτων (ο τρόπος υλοποίησης των οποίων μας είναι άγνωστος) που διαθέτουν διαφορετικές ανεξάρτητες οντότητες, οι οποίες αλληλεπιδρούν για τον σκοπό αυτό.

Οι θεμελιώδεις αρχές του αντικειμενοστραφούς προγραμματισμού πηγάζουν από τον καθημερινό μας φυσικό κόσμο, καθώς είναι πολύ κοντά στον τρόπο που σκεφτόμαστε για να επιλύσουμε προβλήματα της καθημερινότητάς μας. Στηρίζονται στο γεγονός ότι για να μπορέσει κάποιος να κατανοήσει άγνωστες σε αυτόν έννοιες, θα πρέπει να καθοδηγηθεί μέσω της προσομοίωσης των άγνωστων αυτών εννοιών αντιστοιχίζοντας αυτές σε πρακτικές γνώσεις και εικόνες από το περιβάλλον του, τις οποίες γνωρίζει και μπορεί πολύ εύκολα να χειριστεί.

Ερώτηση 11.7

Ποια είναι τα βήματα στη μεθοδολογία επίλυσης προβλημάτων στην αντικειμενοστραφή προσέγγιση;

Για να αναλύσουμε το πρόβλημα το οποίο θέλουμε να επιλύσουμε, πρέπει να αναγνωρίσουμε και να καταγράψουμε τα βασικά συστατικά στοιχεία της διαδικασίας επίλυσής του που είναι:

- τα **αντικείμενα** που συμμετέχουν με βάση τον ρόλο τους στο συγκεκριμένο σενάριο,
- οι **ιδιότητες** κάθε αντικειμένου, δηλ. τα σχετικά με το συγκεκριμένο πρόβλημα χαρακτηριστικά του, και
- οι **υπηρεσίες** που προσφέρει ή οι **ενέργειες** που υλοποιεί κάθε αντικείμενο (μέθοδοι) προς αξιοποίηση από άλλες, ώστε να αναπτυχθούν οι απαραίτητες **συνεργασίες** μεταξύ των αντικειμένων για την επίλυση του προβλήματος.

Ερώτηση 11.8

Πως παρουσιάζονται οι μέθοδοι έναντι των ιδιοτήτων στον αντικειμενοστραφή προγραμματισμό;

Οι μέθοδοι παρουσιάζονται με βάση την ονοματοδοσία των υποπρογραμμάτων, ώστε να διαφοροποιείται ο ρόλος τους ως λειτουργικών στοιχείων των αντικειμένων (ενέργειες ή υπηρεσίες που εφαρμόζονται πάνω στα δεδομένα) έναντι των ιδιοτήτων, οι οποίες αναπαριστούν τα δεδομένα που προσδιορίζουν την υπόστασή τους.

Ερώτηση 11.9

Τι περιλαμβάνει η διαγραμματική αναπαράσταση των αντικειμένων;

Αφού εντοπίσουμε τα συστατικά επίλυσης του προβλήματος, μπορούμε να τα οργανώσουμε σε μια απλή διαγραμματική αναπαράσταση χρησιμοποιώντας παραλληλόγραμμα για την αποτύπωση

- των αντικειμένων,
- των ιδιοτήτων και των μεθόδων τους
- και γραμμές σύνδεσης για την περιγραφή του είδους της μεταξύ τους συνεργασίας .

Εικόνα 4.1 Διαγραμματική αναπαράσταση συστατικών επίλυσης προβλημάτων

Ερώτηση 11.10

Ποια η δομή ενός αντικειμενοστραφούς προγράμματος;

Ενα αντικειμενοστραφές πρόγραμμα δομείται ως ένα **δίκτυο συνεργαζόμενων οντοτήτων** που είναι τα αντικείμενα. Κάθε αντικείμενο έχει ένα συγκεκριμένο ρόλο στην εφαρμογή και παρέχει μια υπηρεσία ή εκτελεί μια ενέργεια (μέθοδο) που χρησιμοποιείται από άλλα μέλη του δικτύου, δηλαδή από άλλα αντικείμενα, για την υλοποίηση της συνεργασίας που θα επιλύσει το πρόβλημα.

Ερώτηση 11.11

Τι ονομάζεται ενθυλάκωση;

Σε μια αντικειμενοστραφή εφαρμογή κάθε αντικείμενο αποτελεί ξεχωριστή οντότητα και περιέχει ενσωματωμένες τις ιδιότητες (δεδομένα) και τους κανόνες συμπεριφοράς του (μεθόδους). Η δυνατότητα ενός αντικειμένου να συνδυάζει εσωτερικά τα δεδομένα και τις μεθόδους χειρισμού του καλείται **ενθυλάκωση** (encapsulation).

Την ενθυλάκωση μπορούμε να την παρομοιάσουμε σαν ένα κέλυφος που υπάρχει γύρω από κάθε αντικείμενο και διαχωρίζει τον εσωτερικό από τον εξωτερικό του κόσμο.

Ερώτηση 11.12

Τι ονομάζεται κλάση;

Ο γενικός τύπος ενός αντικειμένου καλείται **κλάση** (class) και καθορίζει τις αρχικές ιδιότητες και τη συμπεριφορά κάθε αντικειμένου που προέρχεται από αυτή.

Μια κλάση αποτελεί ένα **αφαιρετικό** (abstract) στοιχείο (τύπο) και μπορεί να παράγει ένα απεριόριστο πλήθος δομικά ίδιων αντικειμένων.

Ερώτηση 11.13

Δώστε δύο παραδείγματα κλάσεων.

Παράδειγμα 1ο

Ενα παράδειγμα από τον πραγματικό κόσμο είναι τα αυτοκίνητα. Ολα τα αυτοκίνητα ενός συγκεκριμένου μοντέλου παράγονται με βάση το ίδιο σχέδιο που καθορίζει τις προδιαγραφές του οχήματος, π.χ. διαστάσεις αμαξώματος, διαστάσεις τροχών, κυβισμός, είδος κιβωτίου ταχυτήτων, είδος καυσίμου, χρώμα αμαξώματος, επένδυση καθισμάτων, κ.λπ.

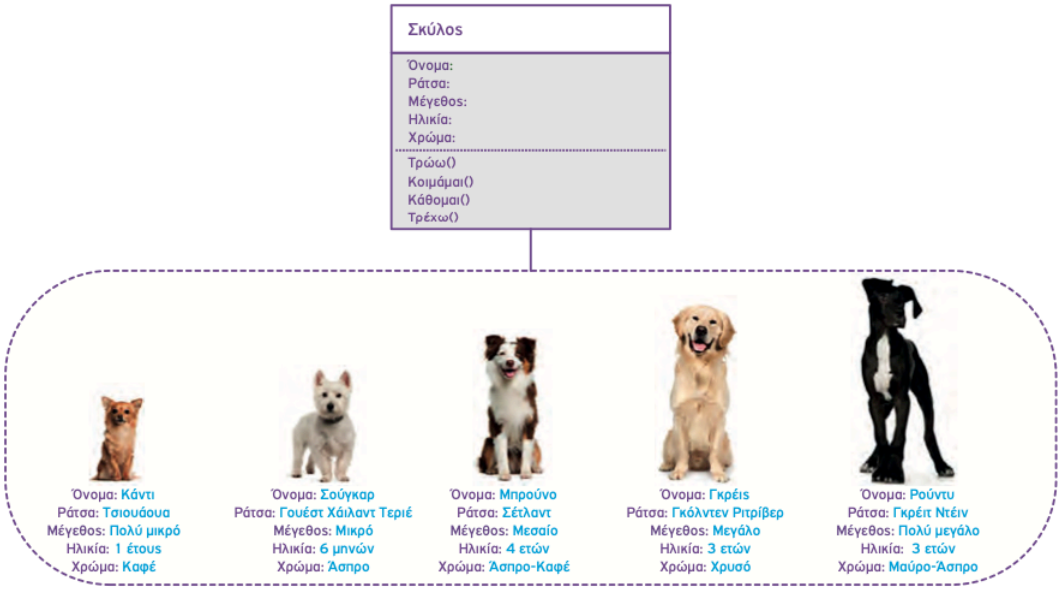
Με βάση αυτό το κοινό σχέδιο παράγονται από το εργοστάσιο πολλά διαφορετικά οχήματα του ίδιου μοντέλου. Κάθε όχημα διαφοροποιείται από τα υπόλοιπα στις τιμές κάποιων ιδιοτήτων.

Ακόμα όμως και αν παραχθούν δύο οχήματα με τις ίδιες ακριβώς τιμές για τις ιδιότητές τους (κάτι που είναι συνηθισμένο), τα οχήματα συνεχίζουν να αποτελούν διαφορετικές οντότητες.

Μπορούμε λοιπόν να θεωρήσουμε το σχέδιο του συγκεκριμένου μοντέλου αυτοκινήτου ως **κλάση** και τα οχήματα που κατασκευάζονται με βάση το σχέδιο ως **αντικείμενα της κλάσης**.

Παράδειγμα 2ο

Στο ηλεκτρονικό παιχνίδι «Η φάρμα των ζώων» ο σκύλος μπορεί να αποτελέσει μια **κλάση** με βάση την οποία έχουμε τη δυνατότητα να δημιουργούμε απεριόριστα διαφορετικά σκυλάκια (**αντικείμενα**) για καθένα από τα οποία θα ορίζουμε τη ράτσα, το μέγεθος, την ηλικία και το χρώμα (ιδιότητες), όπως επίσης και τις δραστηριότητες που θα μπορεί να έχει, π.χ. Τρώω(), Κοιμάμαι(), Κάθομαι(), Τρέχω() και ότι άλλο θέλουμε να μπορούν να κάνουν οι ψηφιακοί μας φίλοι μέσα στο παιχνίδι (**μέθοδοι**)!!!



Εικόνα 4.3. Η κλάση «Σκύλος» και μερικά αντικείμενά της

Ερώτηση 11.14

Τι περιλαμβάνει η διαγραμματική αναπαράσταση των κλάσεων της εφαρμογής και των μεταξύ τους συνεργασιών;

Για την υλοποίηση της διαγραμματικής αναπαράστασης απαιτείται η πλήρης προδιαγραφή καθεμιάς από τις συμμετέχουσες κλάσεις, δηλαδή ο εντοπισμός των ιδιοτήτων, των μεθόδων και του είδους της μεταξύ τους συνεργασίας.

Ερώτηση 11.15

Με ποια μεθοδολογία τοποθετούνται στις κλάσεις οι μέθοδοι που υλοποιούν τις σχέσεις μεταξύ τους;

Οταν μια λειτουργία απαιτεί ενέργειες και από τις δύο κλάσεις που συμμετέχουν σε μια σχέση, τότε πρέπει να υπάρχουν αντίστοιχες μέθοδοι και στις δύο κλάσεις. Οταν δεν απαιτείται ενέργεια και από τις δύο κλάσεις που συμμετέχουν σε μια σχέση, τότε πρέπει να αποφασίσουμε σε ποια κλάση θα οριστεί η μέθοδος.

Σε αυτές τις περιπτώσεις η μέθοδος ορίζεται στην κλάση που ολοκληρώνει την ενέργεια. Αυτό συμβαίνει επειδή η ολοκλήρωση της ενέργειας θα έχει ως αποτέλεσμα να μεταβληθούν οι τιμές κάποιων ιδιοτήτων, συνεπώς η μέθοδος και οι ιδιότητες πρέπει να βρίσκονται στην ίδια κλάση. Με αυτόν τον τρόπο, κάθε κλάση επιτρέπει την πρόσβαση στις ιδιότητές της μέσω των μεθόδων που διαθέτει.

Ερώτηση 11.16

Πως ορίζονται οι ιδιότητες σε κάθε κλάση;

Σε ένα αντικειμενοστραφές προγραμματιστικό περιβάλλον η υποστήριξη κλάσεων αποτελεί κυρίαρχο στοιχείο. Η κλάση είναι η στατική περιγραφή ενός συνόλου αντικειμένων. Ολα τα αντικείμενα δημιουργούνται ως ακριβή αντίγραφα της κλάσης τους. Για παράδειγμα, διαφορετικά αντικείμενα εικόνας, αποτελούν στιγμιότυπα της κλάσης εικόνα.

Κατά τη δημιουργία τους τα διαφορετικά αυτά αντικείμενα έχουν τις ίδιες ιδιότητες και αντιδρούν με τον ίδιο τρόπο σε όποιο μήνυμα ανιχνευτεί από το περιβάλλον τους. Καθένα από τα αντικείμενα εικόνας κληρονομεί τις αρχικές ιδιότητες και την συμπεριφορά του από την κλάση εικόνα.

Ωστόσο ο προγραμματιστής έχει τη δυνατότητα, μετατρέποντας τις τιμές των ιδιοτήτων τους (attributes ή properties), να διαμορφώσει διαφορετικά την εξωτερική εμφάνιση των αντικειμένων. Μπορεί επίσης με τη συγγραφή των κατάλληλων εντολών κώδικα να αντιστοιχήσει διαφορετικές εργασίες σε καθένα από αυτά τα αντικείμενα εικόνας.

Σε ένα γραφικό αντικειμενοστραφές περιβάλλον ανάπτυξης εφαρμογών, οι ιδιότητες των αντικειμένων είναι δυνατό να πάρουν αρχικές τιμές κατά το σχεδιασμό, αλλά τις περισσότερες φορές έχουμε τη δυνατότητα να τις μετατρέψουμε και κατά την εκτέλεση της εφαρμογής.

Ερώτηση 11.17

Τι ονομάζεται κληρονομικότητα και ποια τα χαρακτηριστικά από τις κλάσεις προγόνοι-απόγονοι;

Η δυνατότητα δημιουργίας ιεραρχιών αντικειμένων καλείται **κληρονομικότητα** (inheritance). Με βάση την κληρονομικότητα, μια κλάση μπορεί να περιγραφεί γενικά και στη συνέχεια μέσω αυτής της κλάσης να οριστούν υποκλάσεις αντικειμένων.

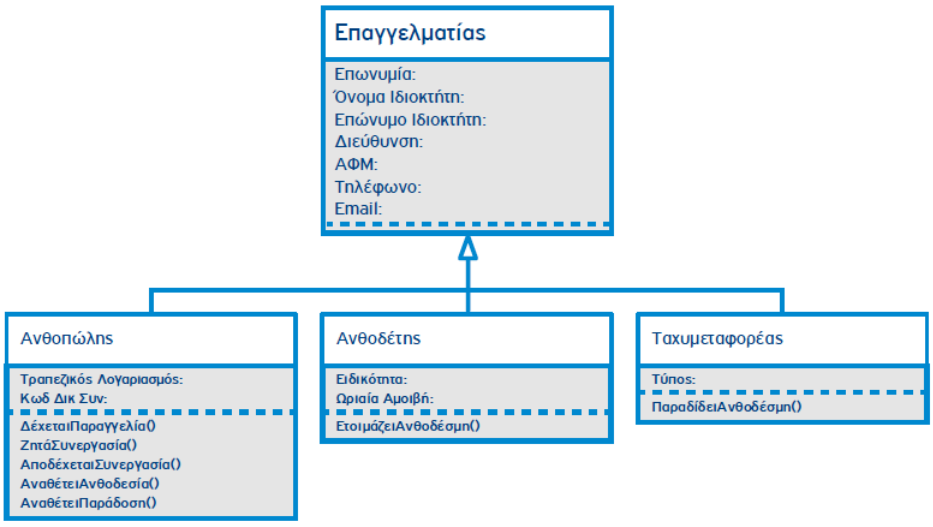
Η κλάση **απόγονος** (υποκλάση) κληρονομεί και μπορεί να χρησιμοποιήσει όλα τα δεδομένα (ιδιότητες) και τις μεθόδους που περιέχει η κλάση **πρόγονος** (υπερκλάση).

Σε μια σχέση κληρονομικότητας, η κλάση-πρόγονος περιλαμβάνει τις κοινές ιδιότητες και μεθόδους όλων των κλάσεων-απογόνων της, ενώ οι κλάσεις-απόγονοι εμφανίζουν μόνο τις διαφορετικές τους ιδιότητες και μεθόδους αφού τις κοινές τις κληρονομούν από τη «μητέρα» τους.

Ερώτηση 11.18

Πώς αποτυπώνεται η σχέση κληρονομικότητας στη διαγραμματική αναπαράσταση;

Η διαγραμματική αναπαράσταση της σχέσης κληρονομικότητας που μόλις περιγράψαμε γίνεται με τη βοήθεια του ειδικού συμβόλου γενίκευσης ↑ όπως παρουσιάζεται στην εικόνα.



Εικόνα 4.10. Ιεραρχία κλάσεων με σχέση κληρονομικότητας επαγγελματιών στην «Αποστολή λουλουδιών»

Ερώτηση 11.19

Πότε μπορεί να θεωρηθεί ως έγκυρη μια υποκλάση;

Η οργάνωση των εννοιών του κόσμου μας στο πλαίσιο ιεραρχιών κληρονομικότητας αποτελεί βασική νοητική λειτουργία κάθε ανθρώπου αλλά και απαραίτητη δεξιότητα στην αντικειμενοστραφή σχεδίαση και τον προγραμματισμό.

Μια κλάση Α μπορεί να είναι έγκυρη υποκλάση της Β αν έχει νόημα να πούμε

«ένα Α είναι ένα (is_a) Β»

Ερώτηση 11.20

Τι ονομάζεται πολυμορφισμός;

Πολυμορφισμός (polymorphism) είναι μια ιδιότητα του αντικειμενοστραφούς προγραμματισμού με την οποία μια λειτουργία μπορεί να υλοποιείται με πολλούς διαφορετικούς τρόπους.

Με άλλα λόγια πολυμορφισμός σημαίνει πολλές διαφορετικές μορφές ή πολλές διαφορετικές συνθήκες. Ο πολυμορφισμός μας επιτρέπει να επαναπροσδιορίσουμε τον τρόπο με τον οποίο λειτουργούν κάποια πράγματα, είτε αλλάζοντας τον τρόπο λειτουργίας τους είτε αλλάζοντας τα εργαλεία τα οποία χρησιμοποιούνται για την επίτευξη του στόχου.

Ερώτηση 11.21

Τι προσφέρει ο πολυμορφισμός στις μεθόδους των αντικείμενων στον αντικειμενοστραφή προγραμματισμό;

Τα αντικείμενα μπορούν δηλαδή να υποστηρίζουν συμπεριφορές (μεθόδους) με κοινό όνομα και τον ίδιο βασικό σκοπό αλλά με διαφορετική λειτουργική υλοποίηση. Κάθε φορά που καλείται μια πολυμορφική λειτουργία, το πρόγραμμα αποφασίζει ποια από τις διαφορετικές μεθόδους με την ίδια ονομασία θα ενεργοποιηθεί, με βάση την κλάση του αντικειμένου στην οποία απευθύνεται η εφαρμογή της λειτουργίας.

Κεφάλαιο 12

Σχεδίαση διεπαφής χρήστη

Είναι εκτός ύλης για το σχολικό έτος 2022-2023.

Κεφάλαιο 13

Εκσφαλμάτωση προγράμματος

Ερώτηση 13.1

Ποιες είναι οι βασικές κατηγορίες λαθών που είναι δυνατό να παρουσιαστούν στα προγράμματα;

Μπορούμε να διακρίνουμε τις εξής κατηγορίες λαθών:

- Συντακτικά λάθη (λάθη κατά την υλοποίηση)
- Λάθη που οδηγούν σε αντικανονικό τερματισμό του προγράμματος (λάθη κατά την εκτέλεση)
- Λογικά λάθη που παράγουν λανθασμένα αποτελέσματα

Ερώτηση 13.2

Που οφείλονται συνήθως συντακτικά λάθη (λάθη κατά την υλοποίηση) και πότε γίνονται αντιληπτά;

Τα λάθη κατά το χρόνο υλοποίησης, προκαλούνται κυρίως από λανθασμένη σύνταξη εντολών προγράμματος. Τέτοια λάθη μπορεί να είναι η λανθασμένη συγγραφή ή παράλειψη μιας δεσμευμένης λέξης της γλώσσας προγραμματισμού, η παράλειψη δήλωσης μιας μεταβλητής, ή η χρήση μιας δομής ελέγχου χωρίς την εντολή τερματισμού της.

Ενα λάθος που προκαλείται κατά τη συγγραφή του προγράμματος, ανιχνεύεται από το μεταγλωττιστή, ο οποίος εμφανίζει προς το προγραμματιστή κάποιο προειδοποιητικό μήνυμα. Αν το πρόγραμμα περιέχει ένα λάθος αυτής της μορφής, δεν επιτρέπεται η εκτέλεσή του, μέχρι να το διορθώσει ο προγραμματιστής.

Τα σύγχρονα προγραμματιστικά περιβάλλοντα μας προφυλάσσουν αυτόματα από τα λάθη κατά την υλοποίηση, αφού παρέχουν εργαλεία αυτόματου ελέγχου σύνταξης των εντολών και παρακολουθούν τον προγραμματιστή κατά τη συγγραφή του προγράμματος. Μόλις διαπιστώσουν κάποιο συντακτικό λάθος, σταματούν και απαιτούν τη διόρθωσή του. Συνήθως αντιλαμβάνονται ακριβώς το λάθος που δημιουργήθηκε και προτείνουν αναλυτικά τον τρόπο διόρθωσής του, εμφανίζοντας σε ενημερωτικό πλαίσιο την ορθή σύνταξη της εντολής που προκλήθηκε το λάθος.

Ερώτηση 13.3

Που οφείλονται συνήθως τα λάθη που οδηγούν σε αντικανονικό τερματισμό του προγράμματος και πότε γίνονται αντιληπτά;

Τα λάθη που προκαλούνται κατά το χρόνο εκτέλεσης του προγράμματος, είναι πιο επώδυνα γιατί συνήθως εμφανίζονται σε πραγματικό περιβάλλον εκτέλεσης και τις περισσότερες φορές προκαλούν τον αντικανονικό τερματισμό της εφαρμογής και το κρέμασμα (crash) του συστήματος.

Οταν ένα λάθος προκληθεί κατά την εκτέλεση της εφαρμογής, είναι δυνατό να αντιμετωπισθεί μόνο με τη χρήση εντολών προγράμματος που το παγιδεύουν και εκτελούν τις κατάλληλες διαδικασίες χειρισμού του.

Η πρόληψη τέτοιων λαθών είναι αρκετά δύσκολη, αφού συνήθως οφείλονται σε καταστάσεις που δεν είναι εύκολο να ελεγχθούν από τον προγραμματιστή, ενώ πολλές φορές εμφανίζονται μετά από μεγάλο χρονικό διάστημα.

Τέτοια λάθη είναι δυνατό να προκληθούν από την κλήση μιας διαδικασίας με δεδομένα που δεν μπορεί να χειριστεί, όπως η αναζήτηση διαγραμμένων αρχείων, η προσπάθεια διαίρεσης ενός αριθμού με το μηδέν, η αν κατά την ανάγνωση ενός αριθμού εισαχθεί ένα γράμμα, η υπερχείλιση μιας αριθμητικής μεταβλητής ή από δυσλειτουργία του υλικού μέρους του υπολογιστή, όπως η καταστροφή του σκληρού δίσκου του συστήματος, ο τερματισμός μιας σύνδεσης δικτύου και η αποσύνδεση του εκτυπωτή.

Ερώτηση 13.4

Που οφείλονται συνήθως τα λογικά λάθη και πότε γίνονται αντιληπτά;

Τα λογικά λάθη είναι συνήθως λάθη σχεδιασμού και δεν προκαλούν τη διακοπή της εκτέλεσης του προγράμματος. Ενώ ο μεταγλωττιστής της γλώσσας προγραμματισμού δεν ανιχνεύει κανένα συντακτικό λάθος και κατά την εκτέλεση του προγράμματος δεν παρουσιάζονται ανεπιθύμητες καταστάσεις σφαλμάτων, τελικά δεν παράγονται τα επιθυμητά αποτελέσματα.

Η ανίχνευση τέτοιων λαθών δεν είναι δυνατό να πραγματοποιηθεί από κάποιο εργαλείο του υπολογιστή και διαπιστώνονται μόνο με τη διαδικασία ελέγχου (testing) και την ανάλυση των αποτελεσμάτων των προγραμμάτων.

Ερώτηση 13.5

Τι ονομάζεται εκσφαλμάτωση και ποιος ο στόχος της;

Η διαδικασία ελέγχου, εντοπισμού και διόρθωσης των σφαλμάτων ενός προγράμματος καλείται εκσφαλμάτωση (debugging). Στόχος της διαδικασίας εκσφαλμάτωσης είναι ο εντοπισμός των σημείων του προγράμματος που προκαλούν προβλήματα στη λειτουργία του.

Η εργασία της εκσφαλμάτωσης δεν είναι εύκολη, απαιτεί βαθιά γνώση της γλώσσας προγραμματισμού και φυσικά αντίστοιχες ικανότητες από τον προγραμματιστή. Για τον εντοπισμό ενός λάθους δεν υπάρχουν ιδιαίτερα μυστικά και τρυκ. Η εκσφαλμάτωση είναι ένα πρόβλημα λογικής και όσο πιο καλά αντιλαμβάνεται ο προγραμματιστής τον τρόπο που εργάζεται το πρόγραμμα, τόσο πιο εύκολα και σύντομα θα εντοπίσει λάθη που προκαλούν δυσλειτουργίες.

Ερώτηση 13.6

Ποιος ο ρόλος των σχολίων στην εκσφαλμάτωση;

Η εισαγωγή γραμμών με σχόλια σε ένα πρόγραμμα υποβοηθά σημαντικά την εκσφαλμάτωση.

Ερώτηση 13.7

Εκσφαλμάτωση λογικών λαθών σε δομή επιλογής

Σε μια δομή επιλογής μπορεί να εμφανιστούν λογικά λάθη που σχετίζονται με:

- τη συνθήκη ή τις συνθήκες
- τις ομάδες εντολών που εκτελούνται όταν μια συνθήκη είναι αληθής ή ψευδής.

Στην ανίχνευση ενός λογικού λάθους στις δομές επιλογής δεν αρκεί η μεμονωμένη μελέτη των συνθηκών και των ομάδων εντολών που εκτελούνται όταν μια συνθήκη είναι αληθής ή ψευδής, αλλά χρειάζεται να μελετηθεί το αποτέλεσμα που παράγει ο συνδυασμός των συνθηκών και των ομάδων εντολών.

Ερώτηση 13.8

Εκσφαλμάτωση λογικών λαθών σε δομή επανάληψης

Σε μια δομή επανάληψης μπορεί να εμφανιστούν λογικά λάθη που σχετίζονται με:

- τη συνθήκη επανάληψης ή τερματισμού,
- την αρχικοποίηση της συνθήκης,
- την ενημέρωση της συνθήκης εντός του βρόχου επανάληψης,
- τις εντολές που περιλαμβάνονται εντός του βρόχου.

Στους υπολογισμούς που γίνονται εντός των δομών επανάληψης χρειάζεται να δοθεί ιδιαίτερη προσοχή στο αν θα συμπεριλάβουμε στον υπολογισμό την τιμή που λαμβάνει κάποια μεταβλητή στην τελευταία επανάληψη.

Συμβουλή: Κατά την εκσφαλμάτωση των δομών επανάληψης χρειάζεται να δίνετε προσοχή στα εξής:

- στους συγκριτικούς και τους λογικούς τελεστές των συνθηκών επανάληψης ή τερματισμού
- στην αρχικοποίηση της συνθήκης
- στην ενημέρωση της συνθήκης εντός του βρόχου
- στην αλληλουχία των εντολών του βρόχου και στη σειρά εκτέλεσής τους
- στο κριτήριο της περατότητας
- στην πρώτη επανάληψη και στην περίπτωση που ο βρόχος επανάληψης δεν πρέπει να εκτελεστεί ούτε μία φορά
- στην τελευταία επανάληψη

Ερώτηση 13.9

Εκσφαλμάτωση λογικών λαθών σε πίνακες

Κατά την εκσφαλμάτωση προγραμμάτων που χρησιμοποιούν πίνακες χρειάζεται να δίνετε ιδιαίτερη προσοχή:

- στο μέγεθος των πινάκων κατά τη δήλωσή τους,
- στους δείκτες των πινάκων κατά την προσπέλασή τους,
- στη μη υπέρβαση των ορίων του πίνακα.

Ερώτηση 13.10

Εκσφαλμάτωση λογικών λαθών στα υποπρογράμματα

Κατά την εκσφαλμάτωση προγραμμάτων που χρησιμοποιούν υποπρογράμματα χρειάζεται να δίνεται προσοχή στον εντοπισμό λογικών λαθών που σχετίζονται με:

- την κλήση του υποπρογράμματος και το πέρασμα των παραμέτρων
- τα λοιπά λογικά λάθη που εμφανίζονται και στα προγράμματα.

Ερώτηση 13.11

Τι προβλέπει ο έλεγχος «μαύρου κουτιού» (black-box testing).

Κατά τον έλεγχο «μαύρου κουτιού», εκτελούνται σενάρια ελέγχου με δεδομένα εισόδου που προκύπτουν από τις προδιαγραφές του προγράμματος, αγνοώντας εντελώς τον κώδικα.

Ερώτηση 13.12

Πώς γίνεται η επιλογή των δεδομένων στον έλεγχο μαύρου κουτιού;

Προσπαθούμε να βρούμε αντιπροσωπευτικές τιμές για τα δεδομένα εισόδου που θα παράγουν αντιπροσωπευτικά αποτελέσματα.

Το πρώτο βήμα είναι η δημιουργία ισοδύναμων διαστημάτων τιμών (equivalence partitioning) για τα δεδομένα εισόδου. Τα διαστήματα θεωρούνται ισοδύναμα, καθώς αν δεν υπάρχουν λάθη, τότε όλες οι τιμές ενός διαστήματος εισόδου θα παράγουν τιμές που θα ανήκουν στο ίδιο διάστημα αποτελεσμάτων. Είναι σημαντικό να δημιουργούνται διαστήματα και για τις μη έγκυρες τιμές εισόδου, καθώς δεν μπορούμε να είμαστε σίγουροι ότι ένα πρόγραμμα θα τροφοδοτείται μόνο με έγκυρες τιμές.

Μετά τον καθορισμό των διαστημάτων πρέπει να επιλεγούν τιμές για τα σενάρια ελέγχου που να καλύπτουν όλα τα διαστήματα. Αφού τα διαστήματα είναι ισοδύναμα, μπορεί να επιλεγεί οποιαδήποτε τιμή από κάθε διάστημα. Μια καλύτερη στρατηγική είναι να γίνει έλεγχος των ακραίων τιμών κάθε διαστήματος (boundary value analysis), καθώς η εμπειρία έχει δείξει ότι τα περισσότερα λάθη γίνονται σε αυτά τα σημεία.

Συμβουλή: Είναι σημαντικό να δημιουργούνται διαστήματα και για τις μη έγκυρες τιμές εισόδου, καθώς δεν μπορούμε να είμαστε σίγουροι ότι ένα πρόγραμμα θα τροφοδοτείται μόνο με έγκυρες τιμές.