1. ΑΝΑΛΥΣΗ ΠΡΟΒΛΗΜΑΤΟΣ

1.1 Τι είναι πρόβλημα

Με τον όρο **Πρόβλημα** εννοείται μια κατάσταση η οποία χρήζει αντιμετώπισης, απαιτεί λύση, η δε λύση της δεν είναι γνωστή, ούτε προφανής.

1.2 Δώστε 3 παραδείγματα προβλημάτων

- Το πρόβλημα με το ψύχος που αντιμετώπισαν τα στρατεύματα του Ναπολέοντα στην εκστρατεία του στη Ρωσία, είχε σαν αποτέλεσμα την ανακοπή της προέλασης και την οπισθοχώρησή του.
- Σοβαρότατα προβλήματα επιδημιών, όπως η πανούκλα, η χολέρα και η λύσσα, αφάνιζαν καθημερινά χιλιάδες ανθρώπους τον περασμένο αιώνα μέχρις ότου επιστήμονες, όπως ο Pasteur και ο Fleming, να ανακαλύψουν τα κατάλληλα εμβόλια.
- Το πρόβλημα της μεταφοράς της ηλεκτρικής ενέργειας από τον τόπο παραγωγής στα σημεία κατανάλωσης πονοκεφάλιασε πολύ τους υπεύθυνους περασμένων εποχών μέχρι να εμφανιστούν οι μετασχηματιστές οι οποίοι έδωσαν λύση στο πρόβλημα.

1.3 Συνάρτηση ποιών παραγόντων είναι η κατανόηση ενός προβλήματος;

Η κατανόηση ενός προβλήματος αποτελεί συνάρτηση δύο παραγόντων,

- της σωστής διατύπωσης εκ μέρους του δημιουργού του και
- της αντίστοιχα σωστής ερμηνείας από τη μεριά εκείνου που καλείται να το αντιμετωπίσει.

1.4 Από τι εξαρτάται η κατανόηση ενός προβλήματος

Η κατανόηση ενός προβλήματος εξαρτάται σε μεγάλο βαθμό από την διατύπωσή του.

1.5 Ποια μέσα χρησιμοποιούμε για την διατύπωση 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**.

1.6 Τι ονομάζεται δεδομένο

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

2.1 Τι είναι πληροφορία

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

1.7 Τι δηλώνει ο όρος 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.

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.

1.9 Με ποιο τρόπο αντιμετωπίζουμε δύσκολα προβλήματα

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.

1.10 Ποιες οι προϋποθέσεις σωστής επίλυσης ενός προβλήματος

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.
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 αποριών σχετικά με τα ζητούμενα, τον τρόπο παρουσίασής 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.

1.11 Ποια τα στάδια αντιμετώπισης ενός προβλήματος

- **κατανόηση**, όπου απαιτείται η σωστή και πλήρης αποσαφήνιση των δεδομένων και των ζητούμενων του προβλήματος
- **ανάλυση**, όπου το αρχικό πρόβλημα διασπάται σε άλλα επί μέρους απλούστερα προβλήματα
- **επίλυση**, όπου υλοποιείται η λύση του προβλήματος, μέσω της λύσης των επιμέρους προβλημάτων.

1.12 Με κριτήριο τη δυνατότητα επίλυσης ενός προβλήματος, ποιες κατηγορίες προβλημάτων διακρίνουμε.

- **Επιλύσιμα**, είναι εκείνα τα προβλήματα για τα οποία η λύση τους είναι ήδη γνωστή και έχει διατυπωθεί. Επιλύσιμα μπορεί επίσης να χαρακτηριστούν και προβλήματα, των οποίων η λύση δεν έχει ακόμα διατυπωθεί, αλλά ή συνάφειά τους με άλλα ήδη επιλυμένα μας επιτρέπει να θεωρούμε σαν βέβαιη τη δυνατότητα επίλυσής τους.
- **Ανοικτά**, ονομάζονται εκείνα τα προβλήματα για τα οποία η λύση τους δεν έχει μεν ακόμα βρεθεί, αλλά παράλληλα δεν έχει αποδειχθεί, ότι δεν επιδέχονται λύση. Σαν παράδειγμα ανοικτού προβλήματος μπορούμε να αναφέρουμε το πρόβλημα της ενοποίησης των τεσσάρων πεδίων δυνάμεων, που αναφέρουμε σε προηγούμενη παράγραφο.
- **Άλυτα**, χαρακτηρίζονται εκείνα τα προβλήματα για τα οποία έχουμε φτάσει στην παραδοχή, ότι δεν επιδέχονται λύση. Τέτοιου είδους πρόβλημα είναι το γνωστό από τους αρχαίους ελληνικούς χρόνους πρόβλημα του τετραγωνισμού του κύκλου. Το πρόβλημα αυτό θεωρείται άλυτο, στην πραγματικότητα η λύση που επιδέχεται είναι προσεγγιστική.

1.13 Με κριτήριο το βαθμό δόμησης των λύσεών τους, τα επιλύσιμα προβλήματα σε ποιες κατηγορίες μπορούν να διακριθούν.

- **Δομημένα**, χαρακτηρίζονται εκείνα τα προβλήματα των οποίων η επίλυση προέρχεται από μια αυτοματοποιημένη διαδικασία. Για παράδειγμα, η επίλυση της δευτεροβάθμιας εξίσωσης αποτελεί ένα δομημένο πρόβλημα, αφού ο τρόπος επίλυσης της εξίσωσης είναι γνωστός και αυτοματοποιημένος.

- **Ημιδομημένα**, ονομάζονται τα προβλήματα εκείνα των οποίων η λύση επιδιώκεται στα πλαίσια ενός εύρους πιθανών λύσεων, αφήνοντας στον ανθρώπινο παράγοντα περιθώρια επιλογής της.
- **Αδόμητα**, χαρακτηρίζονται τα προβλήματα εκείνα των οποίων οι λύσεις δεν μπορούν να δομηθούν ή δεν έχει διερευνηθεί σε βάθος η δυνατότητα δόμησής τους. Πρωτεύοντα ρόλο στην επίλυση αυτού του τύπου προβλημάτων κατέχει η ανθρώπινη διαίσθηση

1.14 Με κριτήριο το είδος της επίλυσης που επιζητούν, τα προβλήματα σε ποιες κατηγορίες διακρίνονται.

- **Απόφασης**, όπου η απόφαση που πρόκειται να ληφθεί σαν λύση του προβλήματος που τίθεται, απαντά σε ένα ερώτημα και πιθανόν αυτή η απάντηση να είναι ένα "Ναι" ή ένα "Όχι". Αυτό που θέλουμε να διαπιστώσουμε σε ένα πρόβλημα απόφασης είναι αν υπάρχει απάντηση που ικανοποιεί τα δεδομένα που θέτονται από το πρόβλημα.
- **Υπολογιστικά**, όπου το πρόβλημα που τίθεται απαιτεί τη διενέργεια υπολογισμών, για να μπορεί να δοθεί μία απάντηση στο πρόβλημα. Σε ένα υπολογιστικό πρόβλημα ζητάμε να βρούμε τη τιμή της απάντησης που ικανοποιεί τα δεδομένα που παρέχει το πρόβλημα.
- **Βελτιστοποίησης**, όπου το πρόβλημα που τίθεται επιζητά το βέλτιστο αποτέλεσμα για τα συγκεκριμένα δεδομένα που διαθέτει. Σε ένα πρόβλημα βελτιστοποίησης αναζητούμε την απάντηση που ικανοποιεί κατά τον καλύτερο τρόπο τα δεδομένα που παρέχει το πρόβλημα.

1.15 Ποιοι οι λόγοι που αναθέτουμε την επίλυση ενός προβλήματος σε υπολογιστή.

- η πολυπλοκότητα των υπολογισμών,
- η επαναληπτικότητα των διαδικασιών,
- η ταχύτητα εκτέλεσης των πράξεων,
- το μεγάλο πλήθος των δεδομένων.

1.16 Ποιες οι λειτουργίες που μπορεί να εκτελεί ο υπολογιστής

- **πρόσθεση**, η οποία αποτελεί τη βασική αριθμητική πράξη, δεδομένου ότι και οι άλλες αριθμητικές πράξεις μπορούν να αντιμετωπιστούν, σαν διαδικασίες πρόσθεσης
- **σύγκριση**, η οποία συνιστά τη βασική λειτουργία για την επιτέλεση όλων των λογικών πράξεων,
- **μεταφορά δεδομένων**, λειτουργία που προηγείται και έπεται της επεξεργασίας δεδομένων.

1.17 Τι είναι διαγραμματική αναπαράσταση

Για τη γραφική απεικόνιση της δομής ενός προβλήματος χρησιμοποιείται συχνότατα η διαγραμματική αναπαράσταση.
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. Έτσι σε κάθε κατώτερο επίπεδο, δημιουργείται η γραφική αναπαράσταση των προβλημάτων στα οποία αναλύονται τα προβλήματα του αμέσως ψηλότερου επιπέδου.

Ερωτήσεις - Θέματα για συζήτηση

1) ***Να γίνει συζήτηση σχετικά με την αντιμετώπιση προβλημάτων με τη χρήση υπολογιστών.***

2) ***Να γίνει συζήτηση σχετικά με τη δημιουργία προβλημάτων εξ αιτίας της χρήσης υπολογιστών.***

3) ***Να δοθεί ο ορισμός των όρων δεδομένο, επεξεργασία δεδομένων, πληροφορία.***

4) ***Να αναφερθούν οι κατηγορίες των προβλημάτων.***

5) ***Για ποιους λόγους αναθέτεται η επίλυση ενός προβλήματος σε υπολογιστή;***

6) ***Περιγράψτε τους τρόπους περιγραφής και αναπαράστασης των προβλημάτων.***

2. ΒΑΣΙΚΕΣ ΕΝΝΟΙΕΣ ΑΛΓΟΡΙΘΜΩΝ

2.1 Τι είναι αλγόριθμος

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

2.2 Ποια τα απαραίτητα κριτήρια που ικανοποιεί ο κάθε αλγόριθμος

- **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 την βοήθεια συναρτήσεων παραγωγής τυχαίων αριθμών ή με τη βοήθεια άλλων απλών εντολών.
- **Έξοδος** (output).
  Ο αλγόριθμος πρέπει να δημιουργεί τουλάχιστον μία τιμή δεδομένων ως αποτέλεσμα προς το χρήστη ή προς έναν άλλο αλγόριθμο.
- **Καθοριστικότητα** (definiteness).
  Κάθε εντολή πρέπει να καθορίζεται χωρίς καμία αμφιβολία για τον τρόπο εκτέλεσής της. Λόγου χάριν, μία εντολή διαίρεσης πρέπει να θεωρεί και την περίπτωση, όπου ο διαιρέτης λαμβάνει μηδενική τιμή.
- **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).
- **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.

2.3 Η έννοια του αλγόριθμου συνδέεται αποκλειστικά και μόνο με προβλήματα της Πληροφορικής;

Η έννοια του αλγόριθμου **δεν** συνδέεται αποκλειστικά και μόνο με προβλήματα της Πληροφορικής.

2.4 Να δοθεί ο ορισμός της επιστήμης της πληροφορικής

Η Πληροφορική, μπορεί να ορισθεί ως η επιστήμη που μελετά τους αλγορίθμους από τις ακόλουθες σκοπιές:

- **Υλικού** (hardware). ,
  Η ταχύτητα εκτέλεσης ενός αλγορίθμου επηρεάζεται από τις διάφορες τεχνολογίες υλικού.
- **Γλωσσών Προγραμματισμού** (programming languages).
  Το είδος της γλώσσας προγραμματισμού που χρησιμοποιείται (δηλαδή, χαμηλότερου ή υψηλότερου επιπέδου) αλλάζει τη δομή και τον αριθμό των εντολών ενός αλγορίθμου.
- **Θεωρητική** (theoretical).
  Το ερώτημα που συχνά τίθεται είναι, αν πράγματι υπάρχει ή όχι κάποιος αποδοτικός αλγόριθμος για την επίλυση ενός προβλήματος.
- **Αναλυτική** (analytical).
  Μελετώνται οι υπολογιστικοί πόροι (computer resources) που απαιτούνται από έναν αλγόριθμο, όπως για παράδειγμα το μέγεθος της κύριας και της δευτερεύουσας μνήμης, ο χρόνος για λειτουργίες CPU και για λειτουργίες εισόδου/εξόδου κ.λπ.

2.5 *Ποια είναι η διαφορά της θεωρητικής από την αναλυτική προσέγγιση σρην επίλυση ενός προβλήματος με χρήση αλγορίθμου;*

Η θεωρητική προσέγγιση προσδιορίζει τα όρια της λύσης ενός αλγορίθμου σε σχέση με το πρόβλημα που καλείται να επιλυθεί, ενώ η αναλυτική προσέγγιση μελετά τους υπολογιστικούς πόρους που απαιτούνται από τον αλγόριθμο.

2.6 Ποιοι οι τρόποι αναπαράστασης ενός αλγορίθμου:

- **με ελεύθερο κείμενο** (free text),
  που αποτελεί τον πιο ανεπεξέργαστο και αδόμητο τρόπο παρουσίασης αλγορίθμου. Έτσι εγκυμονεί τον κίνδυνο ότι μπορεί εύκολα να οδηγήσει σε μη εκτελέσιμη παρουσίαση παραβιάζοντας το τελευταίο χαρακτηριστικό των αλγορίθμων, δηλαδή την αποτελεσματικότητα.
- **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.
- **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.
- **με κωδικοποίηση** (coding),
  δηλαδή με ένα πρόγραμμα που όταν εκτελεσθεί θα δώσει τα ίδια αποτελέσματα με τον αλγόριθμο.

2.7 Ποια τα Σύμβολα διαγράμματος ροής

- **έλλειψη**, που δηλώνει την αρχή και το τέλος του κάθε αλγορίθμου,
- **ρόμβος**, που δηλώνει μία ερώτηση με δύο ή περισσότερες εξόδους για απάντηση,
- **ορθογώνιο**, που δηλώνει την εκτέλεση μίας ή περισσότερων πράξεων, και
- **πλάγιο παραλληλόγραμμο**, που δηλώνει είσοδο ή έξοδο στοιχείων.

2.8 Πότε χρησιμοποιείται η ακολουθιακή (σειριακή) δομή εντολών

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

2.9 Τι ονομάζεται εντολή

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

2.10 Είδη εντολών

- Εκτελεστέα εντολή, πχ. Διάβασε
- Δηλωτική εντολή, πχ Αλγόριθμος

2.11 Τι ονομάζουμε σταθερές

Σταθερές (constands). Με τον όρο αυτό αναφερόμαστε σε προκαθορισμένες τιμές που παραμένουν αμετάβλητες σε όλη τη διάρκεια της εκτέλεσης ενός αλγορίθμου.

2.12 Πως διακρίνουμε τις σταθερές

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

2.13 Τι είναι μεταβλητές

Μεταβλητές (variables). 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.

2.14 Πως διακρίνουμε τις μεταβλητές

Ανάλογα με το είδος της τιμής που μπορούν να λάβουν, οι μεταβλητές διακρίνονται σε:

- Αριθμητικές
- Αλφαριθμητικές
- Λογικές.

2.15 Τι είναι τελεστές

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

2.16 Τι είναι εκφράσεις

Εκφράσεις (expressions). Οι εκφράσεις διαμορφώνονται από τους **τελεστέους** (operands), που είναι σταθερές και μεταβλητές 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 των παρενθέσεων. Μια έκφραση μπορεί να αποτελείται από μια μόνο μεταβλητή ή σταθερά μέχρι μια πολύπλοκη μαθηματική παράσταση.

2.17 Να περιγράψετε τη γενική μορφή τίτλου ενός αλγορίθμου.

- **Αλγόριθμος** όνομα_αλγορίθμου
- **Δεδομένα** //ονόματα δεδομένων//
- **Αποτελέσματα** //ονόματα αποτελεσμάτων//

2.18 Να περιγράψετε τη γενική μορφή ενός αλγορίθμου με ψευδοκώδικα.

ΑΛΓΟΡΙΘΜΟΣ όνομα_αλγορίθμου
Δεδομένα //ονόματα δεδομένων//
… … …
εντολές
Αποτελέσματα //ονόματα αποτελεσμάτων//
… … …
ΤΕΛΟΣ όνομα_αλγορίθμου

2.19 Με ποιου τρόπο μπορούμε να σημειώσουμε σχόλια σ' έναν αλγόριθμο;.

Οτιδήποτε ακολουθεί το θαυμαστικό (!) θεωρείται ότι είναι σχόλιο.

2.20 Να σημειώσετε σ' έναν πίνακα το συμβολισμό των αριθμητικών πράξεων στους αλγορίθμους.

Σύμβολο	*Πράξη*
+	Πρόσθεση
-	Αφαίρεση
*	Πολλαπλασιασμός
/	Διαίρεση
mod	Υπολογισμός ακεραίου υπολοίπου
div	Υπολογισμός ακεραίου πηλίκου
^	Υπολογισμός δύναμης

2.21 Τι γνωρίζεται για την εντολή εκχώρισης;

Το σύμβολο ← χρησιμοποιείται προκειμένου να συμβολίζουμε την εντολή εκχώρισης τιμής. Στα αριστερά του συμβόλου βρίσκεται η μεταβλητή της οποίας η τιμή είναι άγνωστη,

ενώ στα δεξιά βρίσκεται μια παράσταση που χρησιμοποιεί σταθερές και μεταβλητές των οποίων η τιμή είναι γνωστή. Για τον υπολογισμό της τιμής της παράστασης, ισχύουν οι κανόνες υπολογισμού από τα μαθηματικά.

2.22 Πότε χρησιμοποιείται η δομή επιλογής;

Η δομή επιλογής χρησιμοποιείται όταν πρέπει να ληφθεί μία απόφαση με έλεγχο συνθήκης και ανάλογα να εκτελεστούν διαφορετικές ενέργειες.

2.23 Πότε χρησιμοποιείται η δομή πολλαπλής επιλογής;

Οι διαδικασία των πολλαπλής επιλογής εφαρμόζεται στα προβλήματα όπου μπορεί να ληφθούν διαφορετικές αποφάσεις ανάλογα με την τιμή που παίρνει μία έκφραση.

2.24 Τι είναι εμφωλευμένη διαδικασία

Είναι μια διαδικασία κατά την οποία μια εντολή **Αν...τότε** εκτελείται, όταν ισχύει (ή δεν) ισχύει η συνθήκη μίας άλλης εντολής **Αν...τότε**.
Η λογική αυτή επεκτείνεται, δηλαδή να έχουμε νέα εμφωλευμένη δομή μέσα σε μία εμφωλευμένη δομή κοκ.

2.25 Ποιες οι τιμές των τριών λογικών πράξεων;

Πρόταση Α	Πρόταση Β	Α ή Β	Α και Β	όχι Α
Αληθής	Αληθής	Αληθής	Αληθής	Ψευδής
Αληθής	Ψευδής	Αληθής	Ψευδής	Ψευδής
Ψευδής	Αληθής	Αληθής	Ψευδής	Αληθής
Ψευδής	Ψευδής	Ψευδής	Ψευδής	Αληθής

2.26 Πότε χρησιμοποιείται η δομή επανάληψης;

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

2.27 Τι ονομάζεται βρόχος;

Το τμήμα του αλγορίθμου που επαναλαμβάνεται δηλαδή κάθε δομή επανάληψης ονομάζεται βρόχος.

2.28 Μπορεί ένας βρόχος επανάληψης να μην εκτελεστεί ποτέ;

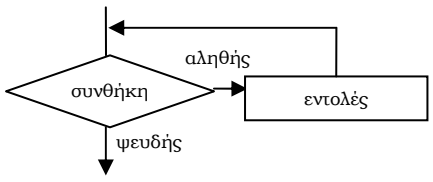
Ο βρόχος επανάληψης μπορεί να μην εκτελεσθεί καμία φορά.

2.29 Ποιες είναι οι δομές επανάληψης:

- Η **ΟΣΟ ... ΕΠΑΝΑΛΑΒΕ ... ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ**
- Η **ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ ... ΜΕΧΡΙΣ_ΟΤΟΥ**
- Η **ΓΙΑ ...ΑΠΟ ... ΜΕΧΡΙ ... ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ**

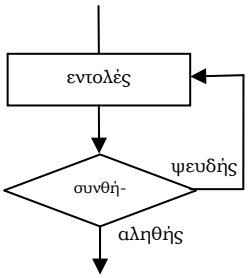
2.30 Αναφέρεται την δομή επανάληψης ΟΣΟ ... ΕΠΑΝΑΛΑΒΕ ... ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ και κάνετε το διάγραμμα ροής της.

ΌΣΟ συνθήκη **ΕΠΑΝΑΛΑΒΕ**
εντολές
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ



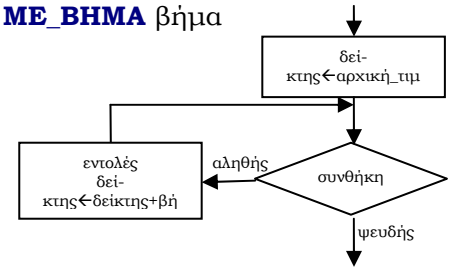
2.31 Αναφέρεται την δομή επανάληψης ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ ... ΜΕΧΡΙΣ_ΟΤΟΥ και κάνετε το διάγραμμα ροής της.

ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
εντολές
ΜΕΧΡΙΣ_ΟΤΟΥ συνθήκη



2.32 Αναφέρεται την δομή επανάληψης ΓΙΑ ...ΑΠΟ ... ΜΕΧΡΙ ... ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ και κάνετε το διάγραμμα ροής της.

ΓΙΑ i **ΑΠΟ** αρχή **ΜΕΧΡΙ** τέλος **ΜΕ_ΒΗΜΑ** βήμα
εντολές
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ



2.33 Πότε συνήθως χρησιμοποιούμε την εκάστοτε δομή επανάληψης;

- Την **ΟΣΟ ... ΕΠΑΝΑΛΑΒΕ ... ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ** την χρησιμοποιούμε συνήθως όταν δεν γνωρίζουμε εκ των προτέρων το πλήθος των επαναλήψεων και δεν μας ενδιαφέρει αν η δομή επανάληψης εκτελεστεί τουλάχιστον μια φορά.
- Την **ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ ... ΜΕΧΡΙΣ_ΟΤΟΥ** την χρησιμοποιούμε συνήθως όταν δεν γνωρίζουμε εκ των προτέρων το πλήθος των επαναλήψεων και μας ενδιαφέρει η δομή επανάληψης να εκτελεστεί τουλάχιστον μια φορά.
- Την **ΓΙΑ ...ΑΠΟ ... ΜΕΧΡΙ ... ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ** την χρησιμοποιούμε συνήθως όταν γνωρίζουμε εκ των προτέρων το πλήθος των επαναλήψεων.

2.34 Ο βρόχος ΓΙΑ k ΑΠΟ n ΜΕΧΡΙ n πόσες φορές εκτελείται;

Ακριβώς μια φορά

2.35 Ο βρόχος ΓΙΑ k ΑΠΟ n ΜΕΧΡΙ 1 πόσες φορές εκτελείται;

Δεν εκτελείται καμία φορά

2.36 Ποια η πρακτική σημασία του αλγορίθμου του 'πολλαπλασιασμού αλά ρωσικά; Πότε και πως γίνεται χρήση αυτού του τρόπου πολλαπλασιασμού δύο ακεραίων ;

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.
Έστω, 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 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 παράδειγμά μας, τα στοιχεία αυτά παρουσιάζονται στην τρίτη στήλη.

Αριθμός	Πολλαπλασιαστής	
45	19	45
90	9	90
180	4	
360	2	
720	1	720
	Γινόμενο=	855

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 διά 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 διά 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.
Στη συνέχεια παρουσιάζεται ο αλγόριθμος πολλαπλασιασμού ακεραίων αλά ρωσικά με φυσική γλώσσα κατά βήματα.

Αλγόριθμος:	Πολλαπλασιασμός δύο θετικών ακεραίων (αλά ρωσικά)
Είσοδος:	Δύο ακέραιοι M1 και M2, όπου M1, M2 ≥ 1
Έξοδος:	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.

Ακολουθεί ο αλγόριθμος σε ψευδοκώδικα για το ίδιο πρόβλημα του πολλαπλασιασμού αλά ρωσικά.

```
ΑΛΓΟΡΙΘΜΟΣ Πολλαπλασιασμός_αλά_ρωσικά
        ΔΕΔΟΜΕΝΑ // M1,M2 ακέραιοι //
        P ←0
        ΌΣΟ M2 > 0 ΕΠΑΝΑΛΑΒΕ
                ΑΝ M2 MOD 2 = 1 ΤΟΤΕ
                        P ←P+M1
                ΤΕΛΟΣ_ΑΝ
                M1 ←M1*2
                M2 ←Α_Μ(M2/2)
        ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
        ΑΠΟΤΕΛΕΣΜΑΤΑ // P, το γινόμενο των ακεραίων M1,M2 //
ΤΕΛΟΣ Πολλαπλασιασμός_αλά_ρωσικά
```

2.37 Τι ονομάζουμε Ολίσθηση (shift)

Η ολίσθηση προς τα αριστερά (προσθήκη ενός 0 στο τέλος ενός αριθμού σε δυαδική μορφή και μετατόπιση των υπολοίπων προς τα αριστερά) ισοδυναμεί με πολλαπλασιασμό επί δύο, ενώ η ολίσθηση προς τα δεξιά (αποκοπή του τελευταίου ψηφίου από το τέλος ενός αριθμού

σε δυαδική μορφή και μετατόπιση των υπολοίπων προς τα δεξιά) ισοδυναμεί με την ακέραια διαίρεση δια δύο.

2.38 Ποια είναι τα στοιχεία της ψευδογλώσσας

1) *Σταθερές*

Αριθμητικές: χρησιμοποιούνται οι αριθμητικοί χαρακτήρες, το +, το – και η τελεία ως δεκαδικό σημείο,

Αλφαριθμητικές: σχηματίζονται από οποιουσδήποτε χαρακτήρες εντός διπλών εισαγωγικών,

Λογικές: υπάρχουν δύο, οι Αληθής και Ψευδής.

2) *Μεταβλητές*

Για τη σύνθεση του ονόματος μιας μεταβλητής χρησιμοποιούνται οι αριθμητικοί χαρακτήρες, οι αλφαβητικοί χαρακτήρες πεζοί και κεφαλαίοι, καθώς και ο χαρακτήρας _ (underscore). Οι μεταβλητές μπορούν επίσης να είναι αριθμητικές, αλφαριθμητικές και λογικές.

3) *Τελεστές*

Αριθμητικοί +, -, *, /, ^

Συγκριτικοί: <=, <, =, <>, >, <=

Λογικοί: και (σύζευξη), ή (διάζευξη), όχι (άρνηση).

4) *Εκφράσεις*

Σχηματίζονται από σταθερές, μεταβλητές, συναρτήσεις, τελεστές και παρενθέσεις.

5) *Εντολή εκχώρησης*

Μεταβλητή ←έκφραση

6) *Σχήματα λογικών υποθέσεων*

- **Απλή επιλογή**
  ΑΝ <συνθήκη> **ΤΟΤΕ** < διαδικασία >
  ή
  ΑΝ <συνθήκη> **ΤΟΤΕ**
  <διαδικασία>
  ΤΕΛΟΣ_ΑΝ

- **Σύνθετη επιλογή**
  ΑΝ <συνθήκη> **ΤΟΤΕ**
  <διαδικασία_1>
  ΑΛΛΙΩΣ
  <διαδικασία_2>
  ΤΕΛΟΣ_ΑΝ

- **Πολλαπλή επιλογή (α μορφή)**
  ΕΠΙΛΕΞΕ έκφραση
  ΠΕΡΙΠΤΩΣΗ 1
  Διαδικασία_1
  .....
  ΠΕΡΙΠΤΩΣΗ ν
  Διαδικασία _ν
  ΠΕΡΙΠΤΩΣΗ ΑΛΛΙΩΣ
  Διαδικασία _αλλιώς
  ΤΕΛΟΣ_ΕΠΙΛΟΓΩΝ

- **Πολλαπλή επιλογή (β μορφή)**
  ΑΝ <συνθήκη_1> **ΤΟΤΕ**
  <διαδικασία_1>
  ΑΛΛΙΩΣ_ΑΝ <συνθήκη_2> **ΤΟΤΕ**
  <διαδικασία_2>
  .....
  ΑΛΛΙΩΣ_ΑΝ <συνθήκη_ν> **ΤΟΤΕ**
  <διαδικασία_ν>
  ΑΛΛΙΩΣ
  <διαδικασία_αλλιώς>
  ΤΕΛΟΣ_ΑΝ
  όπου ως διαδικασία λαμβάνεται ένα σύνολο εντολών

7) *Επαναληπτικές διαδικασίες*

- **Επαναληπτικό σχήμα με έλεγχο επανάληψης στην αρχή**
  ΌΣΟ <συνθήκη> **ΕΠΑΝΑΛΑΒΕ**
  διαδικασία
  ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

- **Επαναληπτικό σχήμα με έλεγχο επανάληψης στο τέλος**
  ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
  Διαδικασία
  ΜΕΧΡΙΣ_ΟΤΟΥ <συνθήκη>
- **Επαναληπτικό σχήμα ορισμένων φορών επανάληψης**
  ΓΙΑ μεταβλητή **ΑΠΟ** τ1 **ΜΕΧΡΙ** τ2 **ΜΕ_ΒΗΜΑ** β
  Διαδικασία
  ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

8) *Ρήματα σε προστακτική*
Για παράδειγμα, “Διάβασε”, “Γράψε”, “Εκτέλεσε” κ.λπ.

9) *Ουσιαστικά*
Σε ορισμένες περιπτώσεις όταν οι ζητούμενες ενέργειες είναι πολλές ή προφανείς, καθορίζονται με τη χρήση ουσιαστικών αντί ρημάτων, όπως “εισαγωγή δεδομένων”, “εμφάνιση πεδίων στην οθόνη” κ.λπ.

10) *Σχόλια*
Προκειμένου να διαχωρίζονται οι επεξηγηματικές φράσεις από τις λέξεις-κλειδιά του αλγορίθμου, στις πρώτες προτάσσεται το σύμβολο !, για παράδειγμα !Σχόλια.

11) *Πρώτη και τελευταία γραμμή* ενός αλγορίθμου είναι αντίστοιχα
Αλγόριθμος <όνομα_αλγορίθμου> και **Τέλος** <όνομα_αλγορίθμου>

12) *Δεδομένα* και αποτελέσματα
Τα δεδομένα εισόδου (αν υπάρχουν) περιγράφονται στη δεύτερη γραμμή του αλγορίθμου εντός των συμβόλων // ... //. Αντίστοιχα τα αποτελέσματα εξόδου δίνονται στην προτελευταία γραμμή του αλγορίθμου εντός των συμβόλων // ... //.

Τεστ αξιολόγησης επίδοσης

Συμπληρώστε με σωστό ή λάθος

1) Ο αλγόριθμος είναι απαραίτητος μόνο για την επίλυση προβλημάτων Πληροφορικής.

2) Ο αλγόριθμος αποτελείται από ένα πεπερασμένο σύνολο εντολών.

3) Ο αλγόριθμος μπορεί να περιλαμβάνει και εντολές που δεν είναι σαφείς.

4) Η Πληροφορική μελετά τους αλγορίθμους μόνο από το πρίσμα των γλωσσών προγραμματισμού.

5) Η αναπαράσταση των αλγορίθμων μπορεί να γίνει μόνο με χρήση ελεύθερου κειμένου και φυσικής γλώσσας.

6) Τα κυριότερα σύμβολα των διαγραμμάτων ροής είναι η έλλειψη, ο ρόμβος, το ορθογώνιο και το πλάγιο παραλληλόγραμμο.

7) Οι κυριότερες εντολές ψευδογλώσσας των αλγορίθμων είναι οι αριθμητικές και αλφαριθμητικές αναθέσεις τιμών σε μεταβλητές.

8) Η ακολουθιακή δομή εντολών χρησιμοποιείται για την επίλυση απλών προβλημάτων με δεδομένη τη σειρά εκτέλεσης ενός συνόλου ενεργειών.

9) Η δομή της ακολουθίας είναι ιδιαίτερα χρήσιμη για την αντιμετώπιση πολύπλοκων προβλημάτων.

10) Η δομή της επιλογής χρησιμοποιείται στις περιπτώσεις όπου υπάρχει μία συγκεκριμένη σειρά βημάτων για την επίλυση ενός προβλήματος.

11) Όταν χρειάζεται να υπάρξει απόφαση με βάση κάποιο κριτήριο, τότε χρησιμοποιείται η δομή της επιλογής.

12) Η δομή της επιλογής περιλαμβάνει τον έλεγχο κάποιας συνθήκης που μπορεί να έχει δύο τιμές (Αληθής ή Ψευδής).

13) Οι διαδικασίες των πολλαπλών επιλογών εφαρμόζονται στα προβλήματα όπου πάντοτε λαμβάνεται η ίδια απόφαση ανάλογα με την τιμή που παίρνει μία μεταβλητή.

14) Μία εμφωλευμένη δομή μπορεί να συμπεριλαμβάνει μόνο την πράξη της ανάθεσης τιμών.

15) Μία εντολή «Αν…τότε» δεν μπορεί να περιληφθεί στα όρια κάποιας άλλης εντολής «Αν…τότε».

Επιλέξτε μεταξύ των προτεινόμενων μία σωστή απάντηση

1) Κάθε αλγόριθμος πρέπει να ικανοποιεί το κριτήριο της:

α) επιλογής
β) ακολουθίας
γ) ανάθεσης
δ) περατότητας

2) Η επιστήμη της Πληροφορικής περιλαμβάνει τη μελέτη των αλγορίθμων μεταξύ άλλων και από τη σκοπιά:

α) υλικού και λογισμικού

β) ελεύθερου κειμένου
γ) αποτελεσματικότητας
δ) ανάγνωσης /εκτύπωσης

3) Ένας από τους τρόπους αναπαράστασης των αλγορίθμων είναι:
α) γλώσσα προγραμματισμού
β) θεωρητική τυποποίηση
γ) διαγραμματικές τεχνικές
δ) αριθμητικές πράξεις

4) Ποια από τις παρακάτω αναπαραστάσεις εκχωρεί στη μεταβλητή Α την τιμή 138
α) Α=138
β) Α=:138
γ) Α:=138
δ) Α ←138

5) Ποια από τα παρακάτω αποτελεί σύμβολο για τα διαγράμματα ροής:
α) έλλειψη
β) τραπέζιο
γ) κύκλος
δ) τετράγωνο

6) Ποια από τα παρακάτω αποτελούν εντολές της ψευδογλώσσας των αλγορίθμων:
α) Α+Β = 10
β) Α ←Β*3
γ) Α+Β ←12
δ) Α ←2*Β ←22

7) Μία εντολή «Αν…τότε» περιλαμβάνει κάποια:
α) συνθήκη
γ) ανάθεση
β) ακολουθία
δ) επανάληψη

8) Οι εμφωλευμένες δομές περιλαμβάνουν συνδυασμό:
α) συνθήκης και εκτύπωσης
β) διαφόρων αλγοριθμικών δομών
γ) συνθήκης και ανάγνωσης
δ) ανάγνωσης και εκτύπωσης

9) Μία εμφωλευμένη δομή χρησιμοποιείται όταν χρειάζεται:
α) μία ενέργεια να περιληφθεί μέσα σε άλλη ενέργεια
β) να υπάρχει επανάληψη τυποποιημένων ενεργειών
γ) να υπάρχει εκτύπωση και ανάγνωση τιμών
δ) να επαναληφθεί μία ενέργεια πολλές φορές

10) Η λογική πράξη ***ή*** μεταξύ 2 προτάσεων είναι αληθής όταν:
α) οποιαδήποτε από τις δύο προτάσεις είναι αληθής.
β) η πρώτη πρόταση είναι ψευδής.
γ) η δεύτερη πρόταση είναι ψευδής.
δ) και οι δύο προτάσεις είναι αληθής.

11) Η λογική πράξη ***και*** μεταξύ 2 προτάσεων είναι αληθής όταν:
α) οποιαδήποτε από τις δύο προτάσεις είναι αληθής.
β) η πρώτη πρόταση είναι αληθής.

γ) η δεύτερη πρόταση είναι αληθής.
δ) και οι δύο προτάσεις είναι αληθείς.

12) Η λογική των επαναληπτικών διαδικασιών εφαρμόζεται στις περιπτώσεις όπου:
α) μία ακολουθία εντολών πρέπει να εφαρμοσθεί σε δύο περιπτώσεις
β) μία ακολουθία εντολών πρέπει να εφαρμοσθεί σε ένα σύνολο περιπτώσεων
γ) υπάρχει απαίτηση να ληφθεί μία απόφαση με βάση κάποια συνθήκη
δ) υπάρχουν δύο συνθήκες που πρέπει να ισχύουν η μία μετά την άλλη.

Συμπληρώστε με σωστό ή λάθος

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

2) Οι επαναληπτικές διαδικασίες εφαρμόζονται όταν μία ακολουθία εντολών πρέπει να εφαρμοσθεί σε δύο περιπτώσεις με βάση κάποια συνθήκη.

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

4) Με χρήση της εντολής "Όσο...επανάλαβε" επιτυγχάνεται η επανάληψη μίας διαδικασίας με βάση κάποια συνθήκη.

5) Με την εντολή «Αρχή_επανάληψης...Μέχρις_ότου...» υπάρχει ένας βρόχος που θα εκτελεσθεί οπωσδήποτε τουλάχιστον μία φορά.

6) Η εντολή "Για i από .. μέχρι .. βήμα .." πρέπει να περιλαμβάνει για βήμα πάντοτε ένα θετικό αριθμό.

Συμπλήρωσε τα κενά με το σωστή λέξη που λείπει

7) Η διαδικασία της _____ είναι ιδιαίτερα συχνή, για πλήθος προβλημάτων μπορεί να επιλυθεί με κατάλληλες επαναληπτικές διαδικασίες.

8) Η επαναληπτική δομή «Επανάλαβε όσο» περιλαμβάνει κάποια(ες) διαδικασίες και λήγει με τη φράση _____

9) Η επαναληπτική δομή που περιλαμβάνει έλεγχο επανάληψης στο τέλος της διαδικασίας ξεκινά με τη φράση «Αρχή_επανάληψης » και λήγει με τη φράση_____

10) Η δομή «_____ από τ1 μέχρι τ2 με_βήμα β » αποτελεί ένα επαναληπτικό σχήμα ορισμένων φορών επανάληψης.

11) Ο πολλαπλασιασμός _____ απαιτεί πολλαπλασιασμό επί δύο, διαίρεση διά δύο και πρόσθεση.

12) Ο αλγόριθμος που δεν διαθέτει τρόπο τερματισμού χαρακτηρίζεται ως _____ βρόχος..

Ερωτήσεις - Θέματα για συζήτηση

1. Να δοθεί ο ορισμός του όρου αλγόριθμος.

2. Ποια είναι τα κριτήρια που πρέπει να ικανοποιεί κάθε αλγόριθμος;

3. Υπό ποια πρίσματα η Πληροφορική επιστήμη μελετά τους αλγορίθμους;

4. Ποια η διαφορά της θεωρητικής από την αναλυτική προσέγγιση στην επίλυση ενός προβλήματος με χρήση αλγορίθμου;

5. Περιγράψτε τους τρόπους περιγραφής και αναπαράστασης των αλγορίθμων.

6. Ποιές είναι οι βασικοί τύποι συνιστωσών/εντολών ενός αλγορίθμου ;

7. Να περιγραφεί η δομή της ακολουθίας και να δοθεί σε διάγραμμα ροής ένα παράδειγμα αυτής της αλγοριθμικής προσέγγισης.

8. Να περιγραφεί η δομή της επιλογής και να δοθεί με ακολουθία βημάτων ένα παράδειγμα αυτής της αλγοριθμικής προσέγγισης.

9. Να περιγραφεί η δομή των επαναληπτικών διαδικασιών και να δοθεί με ακολουθία βημάτων και με διάγραμμα ροής ένα παράδειγμα αυτής της αλγοριθμικής προσέγγισης.

10. Να περιγραφεί η δομή των διαδικασιών πολλαπλών επιλογών και να δοθεί με ακολουθία βημάτων και με διάγραμμα ροής ένα παράδειγμα αυτής της αλγοριθμικής προσέγγισης.

11. Να περιγραφεί η δομή των εμφωλευμένων διαδικασιών και να δοθεί με ακολουθία βημάτων και με διάγραμμα ροής ένα παράδειγμα αυτής της αλγοριθμικής προσέγγισης.

12. Να περιγραφεί με ακολουθία βημάτων το πρόβλημα του ‘πολλαπλασιασμού αλά ρωσικά’.

3. ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΑΛΓΟΡΙΘΜΟΙ

3.1 Τι είναι δεδομένα (data) δώστε ένα παράδειγμα

Τα δεδομένα (data), είναι ακατέργαστα γεγονότα, και κάθε φορά η επιλογή τους εξαρτάται από τον τύπο του προβλήματος.
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.

3.2 Τι είναι πληροφορία (information).

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.

3.3 Ποιο το αντικείμενο της Θεωρίας Πληροφοριών (Information Theory)

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.

3.4 Τι ονομάζουμε Πληροφορική

Πληροφορική θεωρείται η επιστήμη που μελετά τα δεδομένα από τις ακόλουθες σκοπιές:

1) 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.
2) 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.
3) 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. Άλλη 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.
4) 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 Modelling) και της

Αναπαράστασης Γνώσης (Knowledge Representation) ανήκουν σε αυτή τη σκοπιά μελέτης των δεδομένων.

3.5 Τι ονομάζουμε Δομή Δεδομένων

Δομή Δεδομένων είναι ένα σύνολο αποθηκευμένων δεδομένων που υφίστανται επεξεργασία από ένα σύνολο λειτουργιών.
Κάθε μορφή δομής δεδομένων αποτελείται από ένα σύνολο κόμβων (nodes).

3.6 Ποιες οι βασικές λειτουργίες (πράξεις) στις δομές δεδομένων

Οι βασικές λειτουργίες (ή αλλιώς πράξεις) επί των δομών δεδομένων είναι οι ακόλουθες:

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

3.7 Γιατί έχουμε διαφορετικές δομές και ποια η σπουδαιότητα επιλογής της κατάλληλης κάθε φορά

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.

3.8 Ποια η σχέση προγράμματος, δομής δεδομένων και αλγορίθμου

Το πρόγραμμα πρέπει να θεωρεί τη δομή δεδομένων και τον αλγόριθμο ως μία αδιάσπαστη ενότητα.

3.9 Τι αναφέρει η εξίσωση που διατυπώθηκε το 1976 από τον Wirth (που σχεδίασε και υλοποίησε τη γλώσσα Pascal)

Αλγόριθμοι + Δομές Δεδομένων = Προγράμματα

3.10 Ποιες οι κατηγορίες των δομών δεδομένων

Οι δομές δεδομένων διακρίνονται σε δύο μεγάλες κατηγορίες: τις στατικές (static) και τις δυναμικές (dynamic).

3.11 Ποιες οι διαφορές μεταξύ στατικών και δυναμικών δομών

Οι δυναμικές δομές

5) δεν αποθηκεύονται σε συνεχόμενες θέσεις μνήμης αλλά στηρίζονται στην τεχνική της λεγόμενης δυναμικής παραχώρησης μνήμης (dynamic memory allocation).
6) δεν έχουν σταθερό μέγεθος, αλλά ο αριθμός των κόμβων τους μεγαλώνει και μικραίνει καθώς στη δομή εισάγονται νέα δεδομένα ή διαγράφονται κάποια δεδομένα αντίστοιχα.

Οι στατικές δομές

7) αποθηκεύονται σε συνεχόμενες θέσεις μνήμης
8) ότι το ακριβές μέγεθος της απαιτούμενης κύριας μνήμης καθορίζεται κατά τη στιγμή του προγραμματισμού τους, και κατά συνέπεια κατά τη στιγμή της μετάφρασής τους και όχι κατά τη στιγμή της εκτέλεσης τους προγράμματος.

3.12 Τι είναι πίνακας

Πίνακα είναι μια δομή που περιέχει στοιχεία του ίδιου τύπου (δηλαδή ακέραιους, πραγματικούς κ.λπ).

3.13 Πώς γίνεται η αναφορά στα στοιχεία ενός πίνακα

Η αναφορά στα στοιχεία ενός πίνακα γίνεται με τη χρήση του συμβολικού ονόματος του πίνακα ακολουθούμενου από την τιμή ενός ή περισσότερων δεικτών (indexes) σε παρένθεση ή αγκύλη.

3.14 Τι διαστάσεις μπορεί να έχει ένας πίνακας

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

3.15 Τι είναι τετραγωνικός πίνακας

Ένας δισδιάστατος πίνακας που έχει ίδιο αριθμό γραμμών και στηλών λέγεται **τετραγωνικός** (square) και συμβολίζεται n x n. Μάλιστα μπορούμε να θεωρήσουμε το δισδιάστατο πίνακα ότι είναι ένας μονοδιάστατος πίνακας, όπου κάθε θέση του περιέχει ένα νέο μονοδιάστατο πίνακα.

3.16 Τι είναι στοίβα;

Στοίβα είναι μια δομή δεδομένων όπου η εισαγωγή και η εξαγωγή των στοιχείων γίνονται πάντα από την ίδια πλευρά (από την κορυφή της), ενώ η προσπέλαση συγκεκριμένου κόμβου απαιτεί εξαγωγή όλων των κόμβων που έχουν εισαχθεί μετά από αυτό. Η στοίβα είναι μια δομή όπου το στοιχείο που έχει εισαχθεί τελευταίο εξάγεται πρώτο (LIFO – Last In First Out).

3.17 Ποιες είναι οι βασικές λειτουργίες (πράξεις) στη στοίβα.

Δύο είναι οι κύριες λειτουργίες (πράξεις) σε μία στοίβα:

- η **ώθηση** (push) στοιχείου στην κορυφή της στοίβας, και
- η **απώθηση** (pop) στοιχείου από τη στοίβα.

3.18 Τι είναι ώθηση σε μια στοίβα;

Είναι η εισαγωγή νέου κόμβου στην κορυφή της στοίβας

3.19 Τι είναι απώθηση σε μια στοίβα;

Είναι η εξαγωγή ενός κόμβου από την κορυφή της στοίβας.

3.20 Τι είναι υπερχείλιση στοίβας;

Είναι η εξάντληση της διαθέσιμης μνήμης για την στοίβα

3.21 Τι είναι υποχείλιση;

Είναι το άδειασμα της στοίβας

3.22 Τι πρέπει να ελέγχει ο αλγόριθμος πριν πραγματοποιήσει ώθηση;
Πρέπει να κάνει έλεγχο για υπερχείλιση.

3.23 Τι πρέπει να ελέγχει ο αλγόριθμος πριν πραγματοποιήσει απώθηση;
Πρέπει να κάνει έλεγχο για υποχείλιση.

3.24 Πως μπορεί να γίνει η υλοποίηση της στοίβας με χρήση μονοδιάστατου πίνακα;
Χρησιμοποιούμε έναν μονοδιάστατο πίνακα Ν θέσεων και μία μεταβλητή top η οποία δείχνει το στοιχείο που τοποθετήθηκε τελευταίο στην κορυφή της στοίβας. Στην διαδικασία της ώθησης η μεταβλητή top παίρνει την τιμή top+1 (αφού γίνει έλεγχος υπερχείλισης)και στην διαδικασία της απώθησης η μεταβλητή top παίρνει την τιμή top-1 (αφού γίνει έλεγχος υποχείλισης).

3.25 Τι είναι ουρά;
Ουρά είναι μια δομή δεδομένων όπου η εισαγωγή των στοιχείων γίνεται πάντα από τη μια πλευρά (από πίσω), ενώ η εξαγωγή γίνεται πάντα από την άλλη (από μπροστά). Η ουρά είναι μια δομή όπου το στοιχείο που έχει εισαχθεί πρώτο εξάγεται πρώτο (FIFO – First In First Out).

3.26 Ποιες είναι οι βασικές λειτουργίες (πράξεις) στη ουρά;
Δύο είναι οι κύριες λειτουργίες (πράξεις) που εκτελούνται σε μία ουρά:
- η εισαγωγή (enqueue) στοιχείου στο πίσω άκρο της ουράς, και
- η εξαγωγή (dequeue) στοιχείου από το εμπρός άκρο της ουράς.

3.27 Τι είναι εισαγωγή στην ουρά;
Είναι η εισαγωγή νέου κόμβου στην πίσω πλευρά της ουράς.

3.28 Τι είναι εξαγωγή στην ουρά;
Είναι η εξαγωγή ενός κόμβου από την μπροστινή πλευρά της ουράς.

3.29 Τι πρέπει να κάνει ο αλγόριθμος πριν πραγματοποιήσει εισαγωγή στην ουρά;
Πρέπει να κάνει χρήση ενός δείκτη, ο οποίος δείχνει πάντα το τελευταίο στοιχείο που έχει εισαχθεί.

3.30 Τι πρέπει να κάνει ο αλγόριθμος πριν πραγματοποιήσει εξαγωγή από την ουρά;
Πρέπει να κάνει χρήση ενός δείκτη, ο οποίος δείχνει πάντα το τελευταίο στοιχείο που έχει εξαχθεί.

3.31 Τι πρέπει να ελέγχει σε κάθε περίπτωση ο αλγόριθμος σε μια ουρά.
Σε κάθε περίπτωση, πρέπει να ελέγχεται πριν από οποιαδήποτε ενέργεια, αν υπάρχει ελεύθερος χώρος στον πίνακα για την εισαγωγή και αν υπάρχει ένα τουλάχιστον στοιχείο για την εξαγωγή.

3.32 Πως μπορεί να γίνει η υλοποίηση της ουράς με χρήση μονοδιάστατου πίνακα;
Χρησιμοποιούμε έναν μονοδιάστατο πίνακα Ν θέσεων και δύο μεταβλητές front (δείχνει το στοιχείο στην αρχή της ουράς) και rear (δείχνει το στοιχείο στο τέλος της ουράς). Στην εισαγωγή στοιχείο η μεταβλητή rear παίρνει την τιμή rear+1 ενώ στην εξαγωγή στοιχείου , η μεταβλητή front παίρνει την τιμή front+1.

3.33 Δώστε ένα παράδειγμα πως λειτουργεί η στοίβα και πως η ουρά.

Ας θεωρήσουμε για παράδειγμα την περίπτωση ενός αποθηκευτικού χώρου μιας επιχείρησης. Σε κάθε αποθήκη γίνονται εισαγωγές ειδών που προέρχονται από αγορές από προμηθευτές, αν η επιχείρηση είναι εμπορική ή από την παραγωγή, αν πρόκειται για βιομηχανική επιχείρηση. Τα εμπορεύματα ή προϊόντα τοποθετούνται σε κάποιους χώρους, αποθήκες, ράφια κ.λπ. Όταν γίνονται πωλήσεις κάποιων ειδών, τα είδη αυτά βγαίνουν από την αποθήκη και αποστέλλονται στους πελάτες. Έτσι εισαγωγές και εξαγωγές ειδών γίνονται συνεχώς στην αποθήκη ανάλογα με τη διαδικασία προμηθειών και τη ροή των πωλήσεων.
Σε μια δεδομένη στιγμή για κάποιο είδος μπορεί να υπάρχουν αποθηκευμένα κάποια τεμάχια που προέρχονται από μια παραλαβή και κάποια άλλα που υπήρχαν πιο πριν. Όταν πρέπει να εξαχθεί λοιπόν ένα τεμάχιο από αυτό το είδος, προκύπτει το πρόβλημα, από ποια παρτίδα πρέπει να είναι;
Η απάντηση στο ερώτημα αυτό έχει φυσική και λογιστική αξία. Αν το είδος αυτό δεν επηρεάζεται από το χρόνο, τότε ίσως δεν έχει μεγάλη σημασία η επιλογή. Αν όμως πρόκειται για είδος που μπορεί να αλλοιωθεί ή έχει ημερομηνία λήξης (π.χ. φάρμακα), τότε είναι φανερό ότι πρέπει να επιλεγεί το παλαιότερο. Στην περίπτωση αυτή λοιπόν πρέπει η εξαγωγή των ειδών να γίνεται με τη μέθοδο FIFO και συνήθως επαφίεται στον αποθηκάριο να κάνει τη σωστή επιλογή.
Εξ ίσου δύσκολο είναι το πρόβλημα αυτό από την οικονομική και λογιστική σκοπιά, που μάλιστα αφορά όλα τα είδη με ή χωρίς ημερομηνία λήξης. Ας υποθέσουμε ότι μια επιχείρηση έχει πραγματοποιήσει τις επόμενες αγορές και πωλήσεις για ένα είδος.

Αγορές

Ημ/νία	Ποσότητα	Τιμή μονάδας	Αξία
1/1/02	4	10	40
15/1/02	6	12	72
ΣΥΝΟΛΟ	10		112

Πωλήσεις

Ημ/νία	Ποσότητα	Τιμή μονάδας	Αξία
30/1/99	5	20	100

Από τα παραπάνω στοιχεία αγορών και πωλήσεων δημιουργείται η επόμενη καρτέλα είδους.

Καρτέλα είδους

Ημ/νία	Αιτιολογία	Ποσότητα			Αξία κόστους		
		Εισαγωγή	Εξαγωγή	Υπόλοιπο	Εισαγωγή	Εξαγωγή	Υπόλοιπο
1/1/02	Αγορά	4		4	40		40
15/1/02	Αγορά	6		10	72		112
30/1/02	Πώληση		5	5		X	Y

Το πρόβλημα που ανακύπτει στις εφαρμογές αυτές είναι ο καθορισμός των τιμών X και Y. Από τις τιμές αυτές εξάγεται στη συνέχεια το καθαρό κέρδος, με το οποίο η επιχείρηση θα φορολογηθεί.
α) *Λειτουργία LIFO*
Στις 30/1/02 τα 5 τεμάχια που πουλήθηκαν θεωρούνται ότι ανήκουν στα 6 τεμάχια της τελευταίας αγοράς, δηλαδή με τιμή μονάδας 12 €.
Άρα X=5x12 = 60 €. και Y=112-52=60. Κατ' επέκταση το καθαρό κέρδος από την πώληση είναι 100-60=40.
β) *Λειτουργία FIFO*
Στις 30/1/02 από τα 5 τεμάχια που πουλήθηκαν, τα 4 είναι από την αγορά της 1/1/02 και το 1 από την αγορά της 15/1/02. Άρα το κόστος τους είναι X=4x10+1x12=52 και Y=112-52=60. Τώρα, το καθαρό κέρδος της πώλησης γίνεται 100-52=48.

γ) *Λειτουργία με τη σταθμική μέση τιμή*
Λόγω της αυξημένης πολυπλοκότητας των αντίστοιχων προγραμμάτων, αλλά και των απαιτούμενων διαδικασιών οι περισσότερες επιχειρήσεις εφαρμόζουν τη μέθοδο της σταθμικής μέσης τιμής. Η τελευταία για το προηγούμενο παράδειγμα είναι 112/10=11,2. Άρα Χ=5x11,2=56 και Υ=112-56=56. Στην περίπτωση αυτή το καθαρό κέρδος γίνεται 100-56=44 €.

3.34 Από ποιους παράγοντες εξαρτάται η μέθοδος που χρησιμοποιούμε για την αναζήτηση ενός στοιχείου σ' έναν πίνακα;

- Ο πίνακας είναι ταξινομημένος ή όχι.
- Ο πίνακας περιέχει στοιχεία που είναι όλα διάφορα μεταξύ τους ή όχι.
- Τα στοιχεία του πίνακα μπορεί να είναι αριθμητικά ή αλφαριθμητικά.

3.35 Ποια είναι η ποιο απλή μέθοδος αναζήτησης;

Η πιο απλή μορφή αναζήτησης στοιχείου σε πίνακα είναι η **σειριακή** (sequential) ή **γραμμική** (linear) μέθοδος.

3.36 Να εξηγηθεί η σειριακή αναζήτηση.

Η λειτουργία της αναζήτησης σε πίνακα είναι η εύρεση της θέσης στην οποία υπάρχει μια συγκεκριμένη τιμή που ενδιαφέρει τον χρήστη. Οι πιο γνωστές μέθοδοι αναζήτησης είναι η σειριακή και η δυαδική. Η σειριακή μέθοδος αναζήτησης είναι αρκετά απλή στην υλοποίηση, αλλά όχι τόσο αποδοτική, ενώ η δυαδική είναι πιο αποδοτική, αλλά απαιτεί ο πίνακας να είναι ταξινομημένος.
Για να εξηγήσουμε τη σειριακή αναζήτηση σε μονοδιάστατο πίνακα, ας υποθέσουμε ότι έχουμε τον ακόλουθο πίνακα 10 στοιχείων και έστω ότι θέλουμε να βρούμε τη θέση του πίνακα που υπάρχει η τιμή 100.

2	3	5	-1	100	2	2	30

Τα βήματα του αλγόριθμου της σειριακής αναζήτησης είναι:
1° Βήμα: Επισκεπτόμαστε το πρώτο στοιχείο του πίνακα.
2° Βήμα: Ελέγχουμε αν το στοιχείο του πίνακα ισούται με 100. Αν ισούται πάμε στο 4° βήμα, αλλιώς στο 3° βήμα.
3° Βήμα: Υπάρχουν ακόμη στοιχεία στον πίνακα; Αν ναι τότε επισκεπτόμαστε το επόμενο στοιχείο του πίνακα και πάμε στο 2° βήμα. Αν δεν υπάρχουν στοιχεία για έλεγχο η εκτέλεση του αλγορίθμου πηγαίνει στο 5° βήμα.
4° Βήμα: Η τιμή που αναζητούσαμε βρέθηκε και ο αλγόριθμος τερματίζει εμφανίζοντας τη θέση που βρέθηκε το στοιχείο.
5° Βήμα: Ο αλγόριθμος τερματίζει, γιατί εξετάσαμε όλα τα στοιχεία του πίνακα και η τιμή που αναζητούσαμε δεν βρέθηκε.

3.37 Η μέθοδος της σειριακής αναζήτησης σε ποιες περιπτώσεις μπορεί να χωριστεί;

Η μέθοδος λοιπόν της σειριακής αναζήτησης μπορεί να χωριστεί σε 3 περιπτώσεις:

- Εμφάνιση **όλων των θέσεων** που βρίσκεται το στοιχείο που αναζητούμε
- Εμφάνιση της **πρώτης θέσης** που βρίσκεται το στοιχείο που αναζητούμε
- Εμφάνιση της **τελευταίας θέσης** που βρίσκεται το στοιχείο που αναζητούμε

3.38 Ποια η γενική μορφή σειριακής αναζήτησης για την εμφάνιση όλων των θέσεων του στοιχείου που αναζητάμε

```
ΑΛΓΟΡΙΘΜΟΣ Σειριακή_Αναζήτηση_Εμφάνιση_Όλων
        ΔΕΔΟΜΕΝΑ // Ν, Π, x //
! Ν: διάσταση πίνακα, Π: πίνακας, x: τιμή αναζήτησης
        βρέθηκε ← Ψευδής
        ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ Ν
                ΑΝ Π[i] = x ΤΟΤΕ
```

```
                ΕΜΦΑΝΙΣΕ "Η τιμή βρέθηκε στη θέση ", i
                βρέθηκε ← Αληθής
            ΤΕΛΟΣ_ΑΝ
        ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
        ΑΝ βρέθηκε = Ψευδής ΤΟΤΕ
            ΕΜΦΑΝΙΣΕ "Η τιμή δεν βρέθηκε στον πίνακα"
        ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ Σειριακή_Αναζήτηση_Εμφάνιση_Όλων
```

3.39 Ποια η γενική μορφή σειριακής αναζήτησης για την εμφάνιση της πρώτης θέσης του στοιχείου που αναζητάμε

```
ΑΛΓΟΡΙΘΜΟΣ Σειριακή_Αναζήτηση_Πρώτη_Εμφάνιση
        ΔΕΔΟΜΕΝΑ // Ν, Π, x //
! Ν: διάσταση πίνακα, Π: πίνακας, x: τιμή αναζήτησης
        βρέθηκε ← Ψευδής
        i←1
        ΌΣΟ i <= Ν ΚΑΙ βρέθηκε = Ψευδής ΕΠΑΝΑΛΑΒΕ
            ΑΝ Π[i] = x ΤΟΤΕ
                ΕΜΦΑΝΙΣΕ "Η τιμή βρέθηκε στη θέση ", i
                βρέθηκε ← Αληθής
            ΤΕΛΟΣ_ΑΝ
            i ← i+1
        ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
        ΑΝ βρέθηκε = Ψευδής ΤΟΤΕ
            ΕΜΦΑΝΙΣΕ "Η τιμή δεν βρέθηκε στον πίνακα"
        ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ Σειριακή_Αναζήτηση_Πρώτη_Εμφάνιση
```

3.40 Ποια η γενική μορφή σειριακής αναζήτησης για την εμφάνιση της πρώτης θέσης του στοιχείου που αναζητάμε

```
ΑΛΓΟΡΙΘΜΟΣ Σειριακή_Αναζήτηση_Τελευταία_Εμφάνιση
        ΔΕΔΟΜΕΝΑ // Ν, Π, x //
        ! Ν.· διάσταση πίνακα, Π: πίνακας, x: τιμή αναζήτησης
        βρέθηκε ← Ψευδής
        ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ Ν
            ΑΝ Π[i] = x ΤΟΤΕ
                θέση ← i
                βρέθηκε ← Αληθής
            ΤΕΛΟΣ_ΑΝ
        ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
        ΑΝ βρέθηκε = Ψευδής ΤΟΤΕ
            ΕΜΦΑΝΙΣΕ "Η τιμή δεν βρέθηκε στον πίνακα"
        ΑΛΛΙΩΣ
            ΕΜΦΑΝΙΣΕ "Η τιμή βρέθηκε στον πίνακα ", θέση
        ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ Σειριακή_Αναζήτηση_Τελευταία_Εμφάνιση
```

3.41 Σε ποιες μόνο περιπτώσεις χρησιμοποιείται η μέθοδος της σειριακής αναζήτησης;

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

- ο πίνακας είναι μη ταξινομημένος,
- ο πίνακας είναι μικρού μεγέθους (για παράδειγμα, n ≤20),

- η αναζήτηση σε ένα συγκεκριμένο πίνακα γίνεται σπάνια,

3.42 Τι ονομάζουμε ταξινόμηση;

Δοθέντων των στοιχείων a1,a2,...,an η ταξινόμηση συνίσταται στη μετάθεση (permutation) της θέσης των στοιχείων, ώστε να τοποθετηθούν σε μία σειρά $a_{k1},a_{k2},...,a_{kn}$ έτσι ώστε, δοθείσης μίας συνάρτησης διάταξης (ordering function), f, να ισχύει:

$$f(a_{k1}) \leq f(a_{k2}) \leq ... \leq f(a_{kn})$$

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.

3.43 Ποια η μέθοδος της ευθείας ανταλλαγής ή φυσαλίδας;

Η μέθοδος της ταξινόμησης ευθείας ανταλλαγής (straight exchange sort) 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 (bubbles) 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.

3.44 Ποια η γενική μορφή αλγορίθμου για την ταξινόμηση ενός πίνακα με την μέθοδο της φυσαλίδας;

```
ΑΛΓΟΡΙΘΜΟΣ Φυσαλίδα
ΔΕΔΟΜΕΝΑ // table, n //
ΓΙΑ i ΑΠΟ 2 ΜΕΧΡΙ n
        ΓΙΑ j ΑΠΟ n ΜΕΧΡΙ i ΜΕ_ΒΗΜΑ –1
                ΑΝ table[j-1] > table[j] ΤΟΤΕ
                        ΑΝΤΙΜΕΤΑΘΕΣΕ table[j-1], table[j]
                ΤΕΛΟΣ_ΑΝ
        ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΑΠΟΤΕΛΕΣΜΑΤΑ // table //
ΤΕΛΟΣ 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 εξής τρεις εντολές.

```
temp ←table[j-1]
table[j-1] ←table[j]
table[j] ←temp
```

3.45 Τι ονομάζουμε αρχεία (files) και γιατί προτιμούνται;

Είναι ειδικές δομές για την αποθήκευση δεδομένων στην δευτερεύουσα μνήμη (π.χ. σκληρό δίσκο). Αυτό γίνεται γιατί το μέγεθος της κύριας μνήμης δεν επαρκεί για την αποθήκευση των δεδομένων, αλλά και γιατί στην περίπτωση του δίσκου, τα δεδομένα δεν χάνονται, αν διακοπεί η ηλεκτρική παροχή κάτι που δεν συμβαίνει στην περίπτωση των δομών της κύριας μνήμης, όπως είναι οι πίνακες, όπου τα δεδομένα χάνονται όταν τελειώσει το πρόγραμμα.

3.46 Πώς ονομάζονται τα στοιχεία ενός αρχείου;

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.

3.47 Ποιους αλγόριθμους γνωρίζετε για την ταξινόμηση πινάκων;

- Αλγόριθμος με την μέθοδο της φυσαλίδας (ο ποιο απλός αλλά και ο ποιο αργός)
- Αλγόριθμος με την μέθοδο της επιλογής
- Αλγόριθμος με την μέθοδο της παρεμβολής
- Αλγόριθμός με την μέθοδο της γρήγορης ταξινόμησης (quicksort) (ο ποιο γρήγορος)

Τεστ αξιολόγησης επίδοσης

Συμπληρώστε με σωστό ή λάθος

1) Αποτελεί δεδομένο ότι το ύψος ενός ατόμου είναι 1,90. Πληροφορία είναι ότι το άτομο αυτό είναι ψηλό.

2) Κάθε δομή μπορεί να χρησιμοποιηθεί σε οποιοδήποτε πρόβλημα ή εφαρμογή

3) Δυναμικές είναι οι δομές που αποθηκεύονται σε συνεχόμενες θέσεις μνήμης

4) Ένας πίνακας έχει σταθερό μέγεθος αλλά μεταβαλλόμενο περιεχόμενο

5) Ένας πίνακας μπορεί να αποθηκεύσει και ακεραίους αλλά και πραγματικούς αριθμούς

Τεστ αξιολόγησης επίδοσης
Συμπληρώστε με σωστό ή λάθος

6) Μία ουρά διατηρεί τα δεδομένα ταξινομημένα ως προς τη σειρά άφιξής τους.

7) Η υλοποίηση της ουράς χρησιμοποιεί μία μόνο μεταβλητή-δείκτη για τη διαχείριση των εισαγωγών/διαγραφών, όπως και η περίπτωση της στοίβας.

8) Όταν ψάχνουμε σε ένα τηλεφωνικό κατάλογο χρησιμοποιούμε τη σειριακή μέθοδο αναζήτησης.

Συμπληρώστε με σωστό ή λάθος

1) Η ταξινόμηση είναι χρήσιμη διαδικασία γιατί έτσι εκτελείται γρηγορότερα η αναζήτηση.

2) Η ταξινόμηση ευθείας ανταλλαγής είναι πολύ αποτελεσματική σε πίνακες που είναι ταξινομημένοι κατά την αντίστροφη φορά σε σχέση με την επιθυμητή.

3) Η ταξινόμηση ευθείας ανταλλαγής είναι πολύ αποτελεσματική αν ο πίνακας περιέχει ίσα κλειδιά.
4) Όταν δεν είναι ιδιαίτερα κρίσιμος ο χρόνος απόκρισης μίας εφαρμογής, τότε μπορεί ο αντίστοιχος αλγόριθμος να είναι αναδρομικός.

5) Η αναδρομή στηρίζεται στη δομή της ουράς.

6) Σε ένα πίνακα δεν υπάρχει τρόπος διάκρισης της φυσικής σειράς αποθήκευσης από τη λογική αλληλουχία των στοιχείων.

7) Βασική λειτουργία σε μία δομή λίστας είναι η προσπέλαση σε τυχαίο κόμβο της δομής.

8) Η ρίζα ενός δένδρου δεν είναι παιδί κάποιου άλλου κόμβου.

Ερωτήσεις - Θέματα για συζήτηση

1) Τι είναι δεδομένα και τι είναι πληροφορία ; Να δοθεί σύντομος ορισμός των όρων αυτών.

2) Ποιες είναι οι απόψεις από τις οποίες η επιστήμη της Πληροφορικής μελετά τα δεδομένα;

3) Να δοθεί ο ορισμός της δομής δεδομένων.

4) Ποιες είναι οι βασικές πράξεις επί των δομών δεδομένων;

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

6) Να περιγραφούν οι δύο κυριότερες κατηγορίες των δομών δεδομένων.

7) Να περιγραφεί η δομή του πίνακα και να δοθεί παράδειγμα χρήσης του.

8) Να δοθεί ο ορισμός της στοίβας.

9) Ποιες είναι οι βασικές λειτουργίες που γίνονται σε μία στοίβα;

10. Να δοθεί ο ορισμός της ουράς.

10) Ποιες είναι οι βασικές λειτουργίες που γίνονται σε μία ουρά;

11) Να περιγραφεί η λειτουργία της αναζήτησης.

12) Να δοθεί ο ορισμός της έννοιας της ταξινόμησης.

13) Να περιγραφεί η ταξινόμηση ευθείας ανταλλαγής και να δοθεί ένα παράδειγμα.

4. ΤΕΧΝΙΚΕΣ ΣΧΕΔΙΑΣΗΣ ΑΛΓΟΡΙΘΜΩΝ

4.1 Τι περιλαμβάνει η ανάλυση ενός προβλήματος σε ένα σύγχρουο υπολογιστικό περιβάλλου;

- την καταγραφή της υπάρχουσας πληροφορίας για το πρόβλημα,
- την αναγνώριση των ιδιαιτεροτήτων του προβλήματος,
- την αποτύπωση των συνθηκών και προϋποθέσεων υλοποίησής του

και στη συνέχεια:

- την πρόταση επίλυσης με χρήση κάποιας μεθόδου, και
- την τελική επίλυση με χρήση υπολογιστικών συστημάτων.

4.2 Κατά την ανάλυση ενός προβλήματος σε ποιες ερωτήσεις πρέπει να δοθεί απάντηση;

- 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);
- Ποιος είναι ο τρόπος υλοποίησης στο συγκεκριμένο υπολογιστικό σύστημα (π.χ. επιλογή γλώσσας προγραμματισμού)

4.3 Για ποιο λόγο οι μέθοδοι ανάλυσης και επίλυσης των προβλημάτων παρουσιάζουν ιδιαίτερο ενδιαφέρου;

- 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.

4.4 Τι χρειάζεται να εκπληρώνει κάθε τυποποιημένη τεχνική που χρησιμοποιείται για την επίλυση ενός προβλήματος

- να αντιμετωπίζει με τα δικό της τρόπο τα δεδομένα,
- να έχει τη δική της ακολουθία εντολών και
- να διαθέτει τη δική της αποδοτικότητα.

4.5 Αναφέρετε μερικές τυποποιημένες κατηγορίες τεχνικών για την επίλυση προβλημάτων

- Μέθοδος διαίρει και βασίλευε
- Μέθοδος δυναμικού προγραμματισμού
- Άπληστη μέθοδος

4.6 Υπάρχει αλγόριθμος που να σχεδιάζει αλγορίθμους;

Δεν υπάρχει αλγόριθμος για τη σχεδίαση αλγορίθμων.

4.7 Τι ονομάζονται ευριστικές τεχνικές;

Υπάρχουν πολλά προβλήματα που για την επίλυση τους απαιτούν την εφαρμογή μιας άλλης αντίληψης έξω από τις τεχνικές σχεδίασης αλγορίθμων που αναφέρονται στην βιβλιογραφία. Οι μέθοδοι αυτοί ονομάζονται **ευριστικές τεχνικές.**

Τεστ αξιολόγησης επίδοσης

Συμπληρώστε με σωστό ή λάθος

1) Η λύση σε ένα πρόβλημα μπορεί να προέλθει από ποικίλες και διαφορετικές προσεγγίσεις, τεχνικές και μεθόδους. Η ανάλυση ενός προβλήματος σε ένα σύγχρονο υπολογιστικό περιβάλλον περιλαμβάνει την εισαγωγή και εξαγωγή δεδομένων. Δεν υπάρχουν πολλά πρακτικά προβλήματα που να μπορούν να περιγραφούν με τη βοήθεια ενός διαγράμματος ή γράφου.
4) Ο γράφος αποθηκεύεται (συνήθως) στη μνήμη του υπολογιστή με τη βοήθεια ενός δισδιάστατου πίνακα.
5) Υπάρχει ένας ενιαίος κανόνας που να αναφέρεται στην επίλυση του συνόλου των προβλημάτων.
6) Στην κατηγορία “Διαίρει και Βασίλευε” εντάσσονται οι τεχνικές που αναλύουν ένα πρόβλημα σε μεγαλύτερα προβλήματα.
7) Ο αλγόριθμος της Δυαδικής αναζήτησης εφαρμόζεται σε έναν μη ταξινομημένο πίνακα.
8) Με την τεχνική “Διαίρει και Βασίλευε” υποδιαιρείται το στιγμιότυπο του προβλήματος σε υποστιγμιότυπα του ίδιου προβλήματος. Η φιλοσοφία της τεχνικής του Δυναμικού προγραμματισμού από την αρχή να επιλύονται τα μεγαλύτερα προβλήματα και σταδιακά να επιλύονται τα μικρότερα.
10) Η μέθοδος του Δυναμικού προγραμματισμού χρησιμοποιείται κυρίως για την επίλυση προβλημάτων βελτιστοποίησης, δηλαδή χρησιμοποιείται κυρίως όταν χρειάζεται να βρεθεί το ελάχιστο ή το μέγιστο κάποιου μεγέθους.
11) Η μέθοδος της Άπληστης προσέγγισης προχωρά με βάση σταδιακές επιλογές που αφορούν το βέλτιστο κάθε βήματος, χωρίς μέριμνα για το τελικό βέλτιστο.

Επιλέξτε μεταξύ των προτεινόμενων μία σωστή απάντηση

1) Η ανάλυση προβλημάτων περιλαμβάνει:
   α) καταγραφή υπάρχουσας πληροφορίας
   β) καταγραφή αποτελεσμάτων
   γ) αναγνώριση στοιχείων εισόδου του προβλήματος
   δ) πρόταση για την είσοδο και έξοδο των δεδομένων

2) Ένας από τους αλγορίθμους που εντάσσονται στην κατηγορία «Διαίρει και Βασίλευε» είναι:
   α) ταξινόμηση με ευθεία ανταλλαγή
   β) ταξινόμηση με ευθεία επιλογή
   γ) δυαδική αναζήτηση
   δ) υπολογισμός δύναμης

3) Κάθε τεχνική αλγορίθμων πρέπει να έχει:
   α) τη δική της υπολογιστική μηχανή
   β) τη δική της ακολουθία εντολών
   γ) τη δική της είσοδο και έξοδο
   δ) τη δική της γλώσσα προγραμματισμού

Συμπλήρωσε τα κενά με το σωστή λέξη που λείπει

1) Η φιλοσοφία της τεχνικής _____ είναι από την αρχή να επιλύονται τα μικρότερα προβλήματα και σταδιακά να επιλύονται τα μεγαλύτερα ως σύνθεση των απλούστερων.
2) Η μέθοδος της Άπληστης προσέγγισης προχωρά με βάση σταδιακές επιλογές που αφορούν το _____ κάθε βήματος, χωρίς μέριμνα για το τελικό _____.

3) Η _____ μέθοδος, μπορεί να τυποποιηθεί σύμφωνα με την παραδοχή ότι σε κάθε βήμα γίνεται επιλογή της τρέχουσας βέλτιστης επιλογής και υπάρχει επιβεβαίωση ότι αυτή η προσέγγιση εγγυάται τη συνολική βέλτιστη λύση.

Ερωτήσεις - Θέματα για συζήτηση

1) ***Τι περιλαμβάνει η ανάλυση ενός προβλήματος σε ένα σύγχρονο υπολογιστικό περιβάλλον;***

2) ***Ποιοι είναι οι λόγοι για τους οποίους οι μέθοδοι ανάλυσης και επίλυσης των προβλημάτων παρουσιάζουν ιδιαίτερο ενδιαφέρον;***

3) ***Ποια είναι τα δύο βασικά είδη προσέγγισης της επίλυσης ενός προβλήματος;***

5. Εισαγωγή στον προγραμματισμό

6.1 ***Ποια στάδια περιλαμβάνει η επίλυση ενός προβλήματος;***

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

6.2 ***Με τι ασχολείται ο προγραμματισμός;***

Ο προγραμματισμός ασχολείται με τη διατύπωση του αλγορίθμου σε κατανοητή μορφή από τον υπολογιστή δηλαδή του συνόλου των εντολών που πρέπει να δοθούν στον υπολογιστή, ώστε να υλοποιηθεί ο αλγόριθμος για την επίλυση του προβλήματος, αλλά και με τα δεδομένα, όπως και με τις δομές δεδομένων επί των οποίων ενεργεί.

6.3 ***Από ποιου και πότε διατυπώθηκαν οι βασικές αρχές λειτουργίας των υπολογιστών;***

Οι βασικές αρχές λειτουργίας των υπολογιστών διατυπώθηκαν το 1945 από τον Φον Νόυμαν, και δεν άλλαξαν πρακτικά καθόλου. Όμως οι πρώτοι υπολογιστές, τεράστιοι σε μέγεθος αλλά με πάρα πολύ περιορισμένες δυνατότητες και μικρές ταχύτητας επεξεργασίας εξελίχθηκαν σε πολύ μικρούς σε μέγεθος υπολογιστές με τεράστιες δυνατότητες και ταχύτητες επεξεργασίας.

6.4 ***Ποιος ο σκοπός ανάπτυξης των γλωσσών προγραμματισμού και ποια η εξέλιξη τους στο πέρασμα του χρόνου;***

Οι γλώσσες προγραμματισμού αναπτύχθηκαν με σκοπό την επικοινωνία του ανθρώπου (προγραμματιστή) με τη μηχανή (υπολογιστή). Αν και εξελίσσονται και συνεχώς εμπλουτίζονται με νέες δυνατότητες, τα χαρακτηριστικά τους και οι βασικές τους ιδιότητες ουσιαστικά παραμένουν τα ίδια.

6.5 ***Τι είναι γλώσσα μηχανής;***

Οι εντολές ενός προγράμματος μετατρέπονται σε ακολουθίες που αποτελούνται από 0 και 1 (δηλαδή εντολές σε μορφή κατανοητή από τον υπολογιστή αλλά ακατανόητες από τον άνθρωπο), οι οποίες εκτελούνται από τον υπολογιστή. Αυτή η ακολουθία δυαδικών ψηφίων ονομάζεται γλώσσα μηχανής.

Αλλιώς, ένα πρόγραμμα σε γλώσσα μηχανής είναι μια ακολουθία δυαδικών ψηφίων, που αποτελούν εντολές προς τον επεξεργαστή για στοιχειώδεις λειτουργίες.

6.6 ***Τι είναι συμβολικές γλώσσες ή γλώσσες χαμηλού επιπέδου***

Είναι γλώσσα η οποία ενώ έχει έννοια για τον άνθρωπο, μετατρέπεται εσωτερικά από τους υπολογιστές στις αντίστοιχες ακολουθίες από 0 και 1.

6.7 ***Ποια τα βασικά μειονεκτήματα των συμβολικών γλωσσών;***

- Παραμένουν στενά συνδεδεμένες με την αρχιτεκτονική του κάθε υπολογιστή.
- Δεν διαθέτουν εντολές πιο σύνθετων λειτουργιών οδηγώντας έτσι σε μακροσκελή προγράμματα, που ήταν δύσκολο να γραφούν και κύρια να συντηρηθούν.
- Δεν μπορούν να μεταφερθούν σε άλλον διαφορετικό υπολογιστή, ακόμη και του ίδιου κατασκευαστή, δηλαδή εξαρτώνται από την αρχιτεκτονική του υπολογιστή.

6.8 ***Από τι αποτελούνται οι εντολές σε μια συμβολική γλώσσα;***

Οι εντολές σε συμβολική γλώσσα αποτελούνται από συμβολικά ονόματα που αντιστοιχούν σε εντολές της γλώσσας μηχανής.

6.9 ***Τι είναι γλώσσες υψηλού επιπέδου;***

Οι γλώσσες υψηλού επιπέδου χρησιμοποιούν ως εντολές απλές λέξεις της αγγλικής γλώσσας ακολουθώντας αυστηρούς κανόνες σύνταξης, οι οποίες μεταφράζονται από τον ίδιο τον υπολογιστή σε εντολές σε γλώσσα μηχανής

6.10 ***Ποια είναι η πρώτη γλώσσα υψηλού επιπέδου και από ποιον δημιουργήθηκε;***

Το 1957 η IBM ανάπτυξε την πρώτη γλώσσα υψηλού επιπέδου τη FORTRAN. Το όνομα FORTRAN προέρχεται από τις λέξεις FORmula TRANslation, που σημαίνουν μετάφραση τύπων. Η FORTRAN αναπτύχθηκε ως γλώσσα κατάλληλη για την επίλυση μαθηματικών και επιστημονικών προβλημάτων. Υστερεί στη διαχείριση αρχείων δεδομένων και γενικότερα αλφαριθμητικών πληροφοριών. Γνώρισε πολλές βελτιώσεις με κυριότερους σταθμούς τις εκδόσεις 4, 77, 90/95 και Visual FORTRAN.

6.11 ***Τι είναι μεταφραστής ή συμβολομεταφραστής (assembler).;***

Το έργο της μετάφρασης το αναλαμβάνει ένα ειδικό πρόγραμμα, ο συμβολομεταφραστής (assembler).Μεταφράζει την γλώσσα υψηλού επιπέδου σε γλώσσα μηχανής.

6.12 ***Πότε αναπτύχθηκε η COBOL και ποια τα χαρακτηριστικά της;***

Το 1960 αναπτύχθηκε μία άλλη γλώσσα, σταθμός στον προγραμματισμό η γλώσσα COBOL. Η COBOL όπως δηλώνει και το όνομα της (COmmon Business Oriented Language -Κοινή γλώσσα προσανατολισμένη στις επιχειρήσεις) είναι κατάλληλη για ανάπτυξη εμπορικών εφαρμογών, και γενικότερα διαχειριστικών εφαρμογών, τομέας όπου η FORTRAN υστερούσε. Η γλώσσα COBOL δημιουργήθηκε από την Grace Marray Hopper αξιωματικό του πολεμικού ναυτικού των ΗΠΑ το 1960.

6.13 ***Πότε αναπτύχθηκε η ALGOL και ποια τα χαρακτηριστικά της;***

Μια από τις σημαντικότερες γλώσσα προγραμματισμού με ελάχιστη πρακτική εφαρμογή αλλά που επηρέασε ιδιαίτερα τον προγραμματισμό και τις επόμενες γλώσσες, είναι η ALGOL (ALGOrithmic Language – Αλγοριθμική γλώσσα). Αναπτύχθηκε από Ευρωπαίους επιστήμονες, αρχικά το1960, με σκοπό τη δημιουργία γενικής φύσης προγραμμάτων που να μη συνδέονται με συγκεκριμένες εφαρμογές.

6.14 ***Πότε και γιατί αναπτύχθηκε η PL/1;***

Στα μέσα της δεκαετίας του 60 αναπτύχθηκε η γλώσσα PL/1 (Programming Language/1 – Γλώσσα Προγραμματισμού υπ΄ αριθμόν 1) που προσπάθησε, χωρίς επιτυχία να καλύψει όλους τους τομείς του προγραμματισμού, επιστημονικούς και εμπορικούς, αντικαθιστώντας τόσο τη FORTRAN όσο και την COBOL .

6.15 ***Ποιες γλώσσες αναπτύχθηκαν στο χώρο της Τεχνητής Νοημοσύνης***

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

6.16 ***Πότε από ποιους και για ποιους σκοπούς δημιουργήθηκε η γλώσσα BASIC;***

Η γλώσσα BASIC (Beginner's All Purpose Symbolic Instruction Code – Συμβολικός Κώδικας Εντολών Γενικής Χρήσης για Αρχάριους) δημιουργήθηκε το 1964 στο Dartmouth College από τους καθηγητές Kemeny και Kurtz. Στόχος των δημιουργών της ήταν η υλοποίηση μιας απλής γλώσσας προγραμματισμού για εκπαιδευτικούς σκοπούς.

6.17 ***Πότε από ποιους και για ποιους σκοπούς δημιουργήθηκε η γλώσσα PASCAL;***

Η γλώσσα PASCAL (δημιούργημα του καθηγητή Niklaus Wirth) έφερε μεγάλες αλλαγές στον προγραμματισμό. Παρουσιάστηκε το 1970 και στηρίχτηκε πάνω στην ALGOL. Είναι μία γλώσσα γενικής χρήσης, η οποία είναι κατάλληλη τόσο για την εκπαίδευση όσο και τη δημιουργία ισχυρών προγραμμάτων κάθε τύπου. Χαρακτηριστικό της γλώσσας είναι η η καταλληλότητα για τη δημιουργία δομημένων προγραμμάτων.

6.18 ***Πότε από ποιους και για ποιους σκοπούς δημιουργήθηκε η γλώσσα LOGO;***

Η γλώσσα προγραμματισμού LOGO ολοκληρώθηκε το 1967 στη Βοστώνη από τον Seymoyr Papert. Το όνομά της προέρχεται από την ελληνική λέξη "λόγος". Πρόκειται για γλώσσα κατάλληλη για την εισαγωγή στον προγραμματισμό-μαθητών μικρής ηλικίας.

6.19 ***Για ποιους σκοπούς δημιουργήθηκε η γλώσσα C;***

Η C αναπτύχθηκε στα εργαστήρια της εταιρείας BELL και χρησιμοποιήθηκε για την ανάπτυξη του λειτουργικού συστήματος Unix, γλώσσα με ισχυρά χαρακτηριστικά, μερικά από αυτά κοινά με την Pascal κατάλληλη για ανάπτυξη δομημένων εφαρμογών αλλά και με πολλές δυνατότητες γλώσσας χαμηλού επιπέδου.

6.20 ***Τι γνωρίζεται για την C++***

Η C εξελίχτηκε στη γλώσσα C++, που είναι αντικειμενοστραφής.
Η ιδέα του αντικειμενοστραφούς προγραμματισμού παρουσιάστηκε για πρώτη φορά στη δεκαετία του 70 και συνεχίζει ακόμη να απλώνεται αλλάζοντας τον παραδοσιακό προγραμματισμό.

6.21 ***Τι γνωρίζεται για την JAVA***

Τα τελευταία χρόνια χρησιμοποιείται ιδιαίτερα, ειδικά για προγραμματισμό στο Διαδίκτυο (Internet), η JAVA. Η JAVA είναι μία αντικειμενοστραφής γλώσσα που αναπτύχθηκε από την εταιρεία SUN με σκοπό την ανάπτυξη εφαρμογών, που θα εκτελούνται σε κατανεμημένα περιβάλλοντα, δηλαδή σε διαφορετικούς υπολογιστές οι οποίοι είναι συνδεδεμένοι στο Διαδίκτυο.
Τα προγράμματα αυτά μπορούν να εκτελούνται από διαφορετικούς υπολογιστές, προσωπικούς ή μεγάλα συστήματα με διαφορετικά λειτουργικά συστήματα χωρίς αλλαγές.
Η εμφάνιση των γραφικών περιβαλλόντων εργασίας δημιούργησε την ανάγκη για ανάπτυξη προγραμμάτων που να εκμεταλλεύονται τον γραφικό αυτό τρόπο επικοινωνίας χρήστη-υπολογιστή. Στα περισσότερα προγραμματιστικά περιβάλλοντα που υπήρχαν, ήταν πολύ δύσκολη έως αδύνατη η ανάπτυξη εφαρμογών, ικανών να εκμεταλλεύονται τα γραφικά αυτά χαρακτηριστικά.

6.22 ***Τι γνωρίζεται για την γλώσσα προγραμματισμού LISP***

Η γλώσσα LISP δημιουργήθηκε το 1959 στο MIT. Πρόκειται για μη-διαδικασιακή γλώσσα που προορίζεται για την επεξεργασία συμβολικών δεδομένων. Βασικός τύπος δεδομένων, από τον οποίο εξ άλλου πήρε και το όνομά της, είναι η συνδεδεμένη λίστα.

6.23 ***Τι εννοούμε με τον όρο οπτικός προγραμματισμός***

Με τον όρο οπτικό εννοούμε τη δυνατότητα να δημιουργούμε γραφικά ολόκληρο το περιβάλλον της εφαρμογής για παράδειγμα τα πλαίσια διαλόγου ή τα μενού.

6.24 ***Τι εννοούμε με τον όρο οδηγούμενος προγραμματισμός***

Με τον όρο οδηγούμενο από το γεγονός εννοούμε τη δυνατότητα να ενεργοποιούνται λειτουργίες του προγράμματος με την εκτέλεση ενός γεγονότος, για παράδειγμα την επιλογή μίας εντολής από ένα μενού ή το κλικ του ποντικιού.

6.25 ***Τι γνωρίζετε για την γλώσσα προγραμματισμού dBASE***

Η dBASE παρουσιάστηκε στα τέλη της δεκαετίας του 70 από την εταιρία Ashton-Tate αρχικά για μικρουπολογιστές 8-bit και αργότερα για προσωπικούς υπολογιστές.
Η dBASE υπήρξε ο σπουδαιότερος εκπρόσωπος εξελιγμένων γλωσσών και εργαλείων προγραμματισμού της εποχής με κύριο χαρακτηριστικό τις εξαιρετικές δυνατότητες διαχείρισης αρχείων (βάσεων) δεδομένων.

6.26 *Τι γνωρίζετε για τις γλώσσες 4ης γενιάς;*

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

6.27 *Ποια τα πλεονεκτήματα των γλωσσών υψηλού επιπέδου*

Στα πλεονεκτήματα των γλωσσών προγραμματισμού υψηλού επιπέδου σε σχέση με τις συμβολικές είναι:

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

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

6.28 *Ποια η ταξινόμηση των γλωσσών προγραμματισμού*

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

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

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

- Γλώσσες γενικής χρήσης. Θεωρητικά κάθε γλώσσα γενικής χρήσης μπορεί να χρησιμοποιηθεί για την επίλυση οποιουδήποτε προβλήματος. Στην πράξη ωστόσο κάθε γλώσσα έχει σχεδιαστεί για να ανταποκρίνεται καλύτερα σε ορισμένη κατηγορία προβλημάτων. Διακρίνονται σε:
- Γλώσσες επιστημονικής κατεύθυνσης (science-oriented languages) π.χ. FORTRAN
- Γλώσσες εμπορικής κατεύθυνσης (business-oriented languages) π.χ. COBOL.

Ας σημειωθεί ότι ορισμένες γλώσσες τα καταφέρνουν εξίσου καλά και στους δύο πρηγούμενους τομείς π.χ. BASIC, Pascal.

- Γλώσσες προγραμματισμού συστημάτων (system programming languages) π.χ. C.
- Γλώσσες τεχνητής νοημοσύνης (artificial intelligence languages) π.χ. LISP, PROLOG.

- Γλώσσες ειδικής χρήσης. Πρόκειται για γλώσσες που χρησιμοποιούνται σε ειδικές περιοχές εφαρμογών όπως π.χ. στα γραφικά με υπολογιστή, στη ρομποτική, στη σχεδίαση ολοκληρωμένων κυκλωμάτων, στα Συστήματα Διοίκησης Βάσεων Δεδομένων, στην εκπαίδευση μέσω υπολογιστή κ.α.

6.29 ***Ποια είναι η καλύτερη γλώσσα προγραμματισμού***
Μία γλώσσα προγραμματισμού που να είναι αντικειμενικά καλύτερη από τις άλλες δεν υπάρχει, ούτε πρόκειται να υπάρξει.

6.30 ***Ποιες οι βασικές έννοιες και αρχές μιας γλώσσας προγραμματισμού;***
Οι γλώσσες προγραμματισμού, πού είναι τεχνητές γλώσσες, ακολουθούν τις βασικές έννοιες και αρχές της γλωσσολογίας, επιστήμη που μελετά τις φυσικές γλώσσες.

6.31 ***Από τι προσδιορίζεται μια γλώσσα;***
Μία γλώσσα προσδιορίζεται από το **αλφάβητό** της, το **λεξιλόγιό** της, τη **γραμματική** της και τέλος τη **σημασιολογία** της.

6.32 ***Τι ονομάζεται αλφάβητο μιας γλώσσας;***
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 a-z) καθώς και τα ψηφία και όλα τα σημεία στίξης που χρησιμοποιούνται.

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

6.34 ***Από τι αποτελείται η γραμματική;***
Η Γραμματική αποτελείται από το **τυπικό** ή **τυπολογικό** (accidence) και το **συντακτικό** (syntax).

6.35 ***Τι είναι τυπικό μιας γλώσσας;***
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.

6.36 ***Τι είναι συντακτικό μιας γλώσσας;***
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.

6.37 ***Τι είναι σημασιολογία***
Η σημασιολογία (Semantics) 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.

6.38 ***Ποιες οι διαφορές φυσικών και τεχνητών γλωσσών;***

Μία βασική διαφορά μεταξύ φυσικών και τεχνητών γλωσσών είναι η δυνατότητα εξέλιξής τους. Οι φυσικές γλώσσες εξελίσσονται συνεχώς, νέες λέξεις 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 QuickBasic) ή σε επίπεδο επέκτασης (για παράδειγμα Basic και Visual Basic).

6.39 ***Ποιες τεχνικές σχεδίασης προγραμμάτων υπάρχουν;***

- Ιεραρχική σχεδίαση προγράμματος
- Τμηματικός προγραμματισμός
- Δομημένος προγραμματισμός

6.40 ***Περιγράψτε την τεχνική της 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.

6.41 ***Περιγράψτε την τεχνική του τμηματικού προγραμματισμού***

Ο τμηματικός προγραμματισμός επιτυγχάνεται με την ανάλυση του προβλήματος σε αντίστοιχα υποπροβλήματα, κάθε υποπρόβλημα αποτελεί ανεξάρτητη ενότητα (module), που γράφεται ξεχωριστά από τα υπόλοιπα τμήματα προγράμματος.

Ο τμηματικός προγραμματισμός διευκολύνει τη δημιουργία του προγράμματος, μειώνει τα λάθη και επιτρέπει την ευκολότερη παρακολούθηση, κατανόηση και συντήρηση του προγράμματος από τρίτους.

6.42 ***Τι κάνει η εντολή GOTO;***

Η εντολή GOTO έχει ως αποτέλεσμα την αλλαγή της ροής του προγράμματος, της διακλάδωσης σε μία άλλη εντολή του προγράμματος εκτός από την επόμενη.

6.43 ***Πότε από ποιου και για ποιο σκοπό δημιουργήθηκε η τεχνική του δομημένου προγραμματισμού.***

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

Συγκεκριμένα το 1964 σε ένα συνέδριο στο Ισραήλ παρουσιάστηκε ένα κείμενο των Bohm και Jacopini με τις θεωρητικές αρχές του δομημένου προγραμματισμού.

Οι απόψεις τους δεν έγιναν αρχικά ευρύτερα γνωστές και αποδεκτές, αλλά το 1968 ο καθηγητής Edsger Dijkstra δημοσίευσε ένα κείμενο που έκανε ιδιαίτερη αίσθηση και έμελλε να αλλάξει σταδιακά τον τρόπο προγραμματισμού καθώς και τις ίδιες τις γλώσσες προ-

γραμματισμού. Ο τίτλος της μελέτης αυτής ήταν “GO TO Statement Considered Harmful - η εντολή GOTO θεωρείται επιβλαβής” και θεμελίωνε το δομημένο προγραμματισμό.
Χρειάστηκε όμως να περάσουν αρκετά χρόνια, ώστε να αρχίσει να διαδίδεται η χρήση του δομημένου προγραμματισμού.
Την εποχή εκείνη δεν υπήρχε μία μεθοδολογία για την ανάπτυξη των προγραμμάτων, τα προγράμματα ήταν μεγάλα και ιδιαίτερα μπερδεμένα με αποτέλεσμα να ξοδεύεται πάρα πολύς χρόνος τόσο στην συγγραφή όσο κύρια στη διόρθωση και τη μετέπειτα συντήρηση τους. Βασικός λόγος για τα προβλήματα αυτά ήταν η αλόγιστη χρήση μίας εντολής, της εντολής GOTO πού χρησιμοποιούμενη άλλαζε διαρκώς τη ροή του προγράμματος.
Ο δομημένος προγραμματισμός αναπτύχθηκε από την ανάγκη να υπάρχει μία κοινή μεθοδολογία στην ανάπτυξη των προγραμμάτων και τη μείωση των εντολών GOTO που χρησιμοποιούνται στο πρόγραμμα.

6.44 *Τι είναι 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.

6.45 *Που στηρίζεται ο δομημένος προγραμματισμός*

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

6.46 *Γιατί σήμερα η τεχνική του δομημένου προγραμματισμού είναι η κυρίαρχη τεχνική;*

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.

6.47 *Σήμερα οι σύγχρονες γλώσσες προγραμματισμού περιέχουν την εντολή GOTO;*

Όλες οι σύγχρονες γλώσσες προγραμματισμού, υποστηρίζουν το δομημένο προγραμματισμό και διαθέτουν εντολές που καθιστούν τη χρήση του GOTO περιττή. Για λόγους όμως συμβατότητας με τις παλιότερες εκδόσεις τους καθώς και για λόγους συντήρησης παλιών προγραμμάτων, μερικές τη διατηρούν στο ρεπερτόριο των εντολών τους.
Στη συνέχεια αυτού του βιβλίου η εντολή GOTO δεν θα μας απασχολήσει
και καλό είναι να μη χρησιμοποιείται στην ανάπτυξη προγραμμάτων.

6.48 *Ποια τα πλεονεκτήματα του δομημένου προγραμματισμού;*

Επιγραμματικά μπορούμε να αναφέρουμε τα εξής πλεονεκτήματα του δομημένου προγραμματισμού.

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

6.49 *Πότε και που γεννήθηκε ο αντικειμενοστραφής προγραμματισμός;*

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

6.50 ***Τι είναι αντικειμενοστραφής προγραμματισμός;***
Η αντικειμενοστραφής σχεδίαση εκλαμβάνει ως πρωτεύοντα δομικά στοιχεία ενός προγράμματος τα δεδομένα, από τα οποία δημιουργούνται με κατάλληλη μορφοποίηση τα αντικείμενα (objects). Αυτή η σχεδίαση αποδείχθηκε ότι επιφέρει καλύτερα αποτελέσματα, αφού τα προγράμματα που δημιουργούνται είναι περισσότερο ευέλικτα και επαναχρησιμοποιήσιμα.

6.51 ***Ο αντικειμενοστραφής προγραμματισμός χρησιμοποιεί τις άλλες τεχνικές προγραμματισμού;***
Φυσικά ο αντικειμενοστραφής προγραμματισμός χρησιμοποιεί την ιεραρχική σχεδίαση, τον τμηματικό προγραμματισμό και ακολουθεί τις αρχές του δομημένου προγραμματισμού.

6.52 ***Τι γνωρίζετε για τον παράλληλο προγραμματισμό***
Υπάρχουν υπολογιστές που διαθέτουν περισσότερους από έναν επεξεργαστές. Οι επεξεργαστές αυτοί μοιράζονται την ίδια μνήμη και λειτουργούν παράλληλα εκτελώντας διαφορετικές εντολές του ιδίου προγράμματος. Οι υπολογιστές αυτοί εμφανίζονται θεωρητικά να πετυχαίνουν ταχύτητες, που είναι ασύλληπτες για τους τυπικούς υπολογιστές με έναν επεξεργαστή. Για να εκμεταλλευτούμε όμως την ταχύτητα που προσφέρει η αρχιτεκτονική τους, πρέπει το πρόβλημα να διαιρεθεί σε τμήματα που εκτελούνται παράλληλα και στη συνέχεια να προγραμματιστεί σε ένα προγραμματιστικό περιβάλλον που να επιτρέπει τον παράλληλο προγραμματισμό.
Μια γλώσσα προγραμματισμού που υποστηρίζει παράλληλο προγραμματισμό είναι η OCCAM.

6.53 ***Πως μετατρέπεται ένα πρόγραμμα σε γλώσσα μηχανής;***
Η μετατροπή επιτυγχάνεται με τη χρήση ειδικών μεταφραστικών προγραμμάτων. Υπάρχουν δύο μεγάλες κατηγορίες τέτοιων προγραμμάτων, οι **μεταγλωττιστές (compilers**) και οι **διερμηνευτές (interpreters).**

6.54 ***Τι κάνει ο μεταγλωττιστής***
Ο μεταγλωττιστής δέχεται στην είσοδο ένα πρόγραμμα γραμμένο σε μια γλώσσα υψηλού επιπέδου και παράγει ένα ισοδύναμο πρόγραμμα σε γλώσσα μηχανής. Μπορεί να εκτελείται οποτεδήποτε από τον υπολογιστή και είναι τελείως ανεξάρτητο από το αρχικό πρόγραμμα.

6.55 ***Τι κάνει ο διερμηνευτής***
Ο διερμηνευτής διαβάζει μία προς μία τις εντολές του αρχικού προγράμματος και για κάθε μια εκτελεί αμέσως μια ισοδύναμη ακολουθία εντολών μηχανής.

6.56 ***Τι είναι πηγαίο και τι αντικείμενο πρόγραμμα;***
Το αρχικό πρόγραμμα λέγεται πηγαίο πρόγραμμα (source), ενώ το πρόγραμμα που παράγεται από το μεταγλωττιστή λέγεται αντικείμενο πρόγραμμα (object).

6.57 ***Τι είναι συνδέτης – φορτωτής***
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 εκτέλεσή του, τμήματα που είτε τα γράφει ο

προγραμματιστής είτε βρίσκονται στις βιβλιοθήκες (libraries) της γλώσσας. Το πρόγραμμα που επιτρέπει αυτή τη σύνδεση ονομάζεται συνδέτης – φορτωτής (linkerloader).

6.58 ***Τι ονομάζεται μεταγλώττιση και σύνδεση;***
Το αποτέλεσμα του συνδέτη είναι η παραγωγή του εκτελέσιμου προγράμματος (executable), το οποίο είναι το τελικό πρόγραμμα που εκτελείται από τον υπολογιστή. Για το λόγο αυτό η συνολική διαδικασία αποκαλείται μεταγλώττιση και σύνδεση.

6.59 ***Ποια τα είδη λαθών ενός προγράμματος;***
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.

6.60 ***Πότε εμφανίζονται τα λογικά και πότε τα συντακτικά λάθη;***
Τα λογικά λάθη εμφανίζονται μόνο στην εκτέλεση, ενώ τα συντακτικά λάθη στο στάδιο της μεταγλώττισης.

6.61 ***Που οφείλονται τα λογικά λάθη;***
τα λογικά λάθη που είναι τα πλέον σοβαρά και δύσκολα στη διόρθωση τους, οφείλονται σε σφάλματα κατά την υλοποίηση του αλγορίθμου,

6.62 ***Που οφείλονται τα συντακτικά λάθη;***
Τα συντακτικά λάθη οφείλονται σε αναγραμματισμούς ονομάτων εντολών, παράληψη δήλωσης δεδομένων και πρέπει πάντα να διορθωθούν, ώστε να παραχθεί το τελικό εκτελέσιμο πρόγραμμα.
Ο μεταγλωττιστής ή ο διερμηνευτής ανιχνεύει λοιπόν τα λάθη και εμφανίζει κατάλληλα διαγνωστικά μηνύματα. Το στάδιο που ακολουθεί είναι η διόρθωση των λαθών. Το διορθωμένο πρόγραμμα επαναϋποβάλεται για μεταγλώττιση και η διαδικασία αυτή επαναλαμβάνεται, μέχρις ότου εξαληφθούν πλήρως όλα τα λάθη.

6.63 ***Ποια τα πλεονεκτήματα και ποια τα μειονεκτήματα του μεταγλωττιστή – διερμηνευτή;***
Η χρήση μεταγλωττιστή έχει το μειονέκτημα, ότι προτού χρησιμοποιηθεί ένα πρόγραμμα, πρέπει να περάσει από τη διαδικασία της μεταγλώττισης και σύνδεσης. Από την άλλη μεριά η χρήση διερμηνευτή έχει το πλεονέκτημα της άμεσης εκτέλεσης και συνεπώς και της άμεσης διόρθωσης. Όμως η εκτέλεση του προγράμματος καθίσταται πιο αργή, σημαντικά μερικές φορές, από εκείνη του ισοδύναμου εκτελέσιμου προγράμματος που παράγει ο μεταγλωττιστής.

6.64 ***Τι χρησιμοποιούν, μεταγλωττιστή ή διερμηνευτή, τα σύγχρονα προγραμματιστικά περιβάλλοντα;***
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.

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

6.66 ***Ποια είναι τα προγράμματα για δημιουργία, τη μετάφραση και την εκτέλεση ενός προγράμματος;***
- ο συντάκτης,
- ο μεταγλωττιστής και

- ο συνδέτης.

Ερωτήσεις - Θέματα για συζήτηση

1. ***Τι ονομάζεται πρόγραμμα;***

2. ***Τι είναι οι γλώσσες μηχανής;***

3. ***Ποιες οι διαφορές των γλωσσών υψηλού επιπέδου από αυτές χαμηλού επιπέδου;***

4. ***Ποιες γλώσσες υψηλού επιπέδου γνωρίζεις;***

5. ***Τι ονομάζουμε οπτικό προγραμματισμό και τι οδηγούμενο από τα γεγονότα;***

6. ***Πώς προσδιορίζεται μία φυσική γλώσσα;***

7. ***Ποιες οι κυριότερες διαφορές των φυσικών και των τεχνητών γλωσσών;***

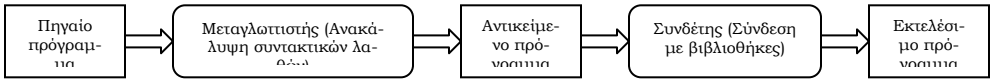
8. ***Πώς γίνεται η παράσταση της ιεραρχικής σχεδίασης προγράμματος;***

9. ***Ποιες οι αρχές του δομημένου προγραμματισμού;***

10. ***Ποια τα πλεονεκτήματα του δομημένου προγραμματισμού;***

11. ***Τι ονομάζεται αντικειμενοστραφής προγραμματισμός;***

12. ***Ποια η διαδικασία για την μετάφραση και εκτέλεση ενός προγράμματος;***



13. ***Ποιες οι διαφορές μεταγλωττιστή και διερμηνευτή;***

14. ***Ποια προγράμματα και εργαλεία περιέχει ένα προγραμματιστικό περιβάλλον;***

Τεστ αξιολόγησης επίδοσης

Συμπληρώστε με σωστό ή λάθος.

1) Τα προγράμματα σε γλώσσες υψηλού επιπέδου είναι ανεξάρτητα του υπολογιστή που αναπτύχθηκαν.
2) Ο μεταγλωττιστής μας επιτρέπει να συντάσσουμε ένα πρόγραμμα.
3) Τα λογικά λάθη ενός προγράμματος εμφανίζονται κατά τη μεταγλώττιση.
4) Ο δομημένος προγραμματισμός επιτρέπει την άμεση μεταφορά των αλγορίθμων σε πρόγραμμα.

Επιλέξτε μεταξύ των προτεινόμενων μία σωστή απάντηση.

5) Χαρακτηριστικό του οπτικού προγραμματισμού είναι
   - Α Επιτρέπει τη γραφική δημιουργία του περιβάλλοντος
   - Β. Επιτρέπει την ανάπτυξη του προγράμματος σε τμήματα
   - Γ. Είναι ταχύτερος στην εκτέλεση των προγραμμάτων
   - Δ. Επιτρέπει την διαγραμματική παράσταση της σχεδίασης του προγράμματος
6) Η Basic είναι
   - Α. Κατάλληλη για εφαρμογές τεχνητής νοημοσύνης
   - Β. Υποστηρίζει την ανάπτυξη παράλληλου προγραμματισμού

Γ. Μία γλώσσα γενικής χρήσης
Δ. Κατάλληλη μόνο για εκπαίδευση.

Να συμπληρωθούν τα κενά

7) Ο δομημένος προγραμματισμός στηρίζεται στη χρήση τριών στοιχειωδών δομών: της ……., της …… και της ……
8) Η μεταγλώττιση ενός προγράμματος γίνεται από τους …… ή τους ……

6. Βασικά στοιχεία προγραμματισμού

7.1 *Από τι εξαρτάται η επιλογή της γλώσσας προγραμματισμού που θα χρησιμοποιήσουμε.*

Η επιλογή της κατάλληλης γλώσσας δεν είναι εύκολη και εξαρτάται από

- το είδος του προγράμματος,
- το διαθέσιμο εξοπλισμό και σαφώς
- τις γνώσεις και τις ιδιαίτερες προτιμήσεις του προγραμματιστή.

Συχνά το ίδιο πρόβλημα μπορεί να λυθεί εξίσου ικανοποιητικά με πολλές διαφορετικές γλώσσες προγραμματισμού.

7.2 *Ποιοι οι λόγοι δημιουργίας τόσων διαφορετικών γλωσσών προγραμματισμού;*

Πρέπει να έχουμε πάντα υπόψη μας ότι:

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

7.3 *Ποια τα χαρακτηριστικά της γλώσσας προγραμματισμού ΓΛΩΣΣΑ;*

Η ΓΛΩΣΣΑ, είναι σχεδιασμένη έτσι ώστε να αποτελεί ένα εργαλείο προγραμματισμού κατάλληλο για εκπαιδευτικούς σκοπούς.

Περιέχει τα χαρακτηριστικά, τις δομές και τις εντολές που περιέχονται σε διάφορες σύγχρονες γλώσσες προγραμματισμού όπως η Pascal, Visual Basic, C, C++, Java και άλλες, χωρίς όμως να ασχολείται με τις τεχνικές λεπτομέρειες αυτών.

Ο προγραμματισμός με τη ΓΛΩΣΣΑ εστιάζεται στην ανάπτυξη του αλγορίθμου και τη μετατροπή του σε σωστό πρόγραμμα.

7.4 *Ποιο το αλφάβητο της ΓΛΩΣΣΑΣ;*

Το αλφάβητο της ΓΛΩΣΣΑΣ αποτελείται από

- τα γράμματα του ελληνικού
  - Κεφαλαία ελληνικού αλφαβήτου (Α-Ω)
  - Πεζά ελληνικού αλφαβήτου (α-ω)
- τα γράμματα του λατινικού αλφαβήτου,
  - Κεφαλαία λατινικού αλφαβήτου (A-Z)
  - Πεζά λατινικού αλφαβήτου (a-z)
- τα ψηφία,
  - 0-9
- τα ειδικά σύμβολα, που χρησιμοποιούνται για προκαθορισμένες ενέργειες,
  - + - * / = ^ ( ) . , ‘ ! & κενός χαρακτήρας

7.5 *Ποιους τύπους δεδομένων υποστηρίζει 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.

7.6 *Τι είναι ακέραιος τύπος;*

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.

7.7 *Τι είναι πραγματικός τύπος;*

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.

7.8 *Τι είναι τύπος χαρακτήρας ;*

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.

7.9 *Τι είναι λογικός χαρακτήρας ;*

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

7.10 *Τι είναι σταθερές*

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

7.11 *Τι είναι συμβολικές σταθερές*

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

7.12 *Πως γίνεται η σύνταξη των σταθερών*

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

7.13 *Που καταχωρούνται τα δεδομένα;*

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. Έτσι 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) σε 4 ή 8 bytes.

7.14 *Το είναι τα ονόματα*

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

7.15 *Ποιες ονομάζονται δεσμευμένες λέξεις;*

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

7.16 *Δώστε παραδείγματα ονομάτων αποδεκτά και μη αποδεκτά από την ΓΛΩΣΣΑ*

- Παραδείγματα ονομάτων που είναι αποδεκτά από τη ΓΛΩΣΣΑ είναι:
  - Α, Όνομα, Τιμή, Τυπική_Απόκλιση, Α100, ΦΠΑ, μέγιστο, Υπολογισμός_Ταχύτητας.
- Παραδείγματα ονομάτων που δεν είναι αποδεκτά είναι:
  - 100Α, Μέση Τιμή, Κόστος$.

7.17 *Τι είναι μεταβλητή;*

Μια μεταβλητή είναι 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 μπορεί να μεταβάλλεται κατά τη διάρκεια της εκτέλεσης του προγράμματος.
Ενώ μένει υποχρεωτικά αναλλοίωτος ο τύπος της μεταβλητής.

7.18 *Τι τύπους μεταβλητών έχουμε;*

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.

7.19 *Ποιοι είναι οι κανόνες ονοματολογίας των 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.

7.20 *Πως γίνεται η σύνταξη των μεταβλητών;*

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

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

7.21 *Ποιοι είναι οι αριθμητικοί τελεστές*

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

Αριθμητικός τελεστής	Πράξη
+	Πρόσθεση
-	Αφαίρεση
*	Πολλαπλασιασμός
/	Διαίρεση
^	Ύψωση σε δύναμη
DIV	Ακέραια διαίρεση
MOD	Υπόλοιπο ακέραιας διαίρεσης

7.22 *Ποιες συναρτήσεις χρησιμοποιούμε στην ΓΛΩΣΣΑ;*

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

ΗΜ(Χ)	Υπολογισμός ημιτόνου
ΣΥΝ(Χ)	Υπολογισμός συνημιτόνου
ΕΦ(Χ)	Υπολογισμός εφαπτομένης
Τ_Ρ(Χ)	Υπολογισμός τετραγωνικής ρίζας
ΛΟΓ(Χ)	Υπολογισμός φυσικού λογαρίθμου
Ε(Χ)	Υπολογισμός του ex
Α_Μ(Χ)	Ακέραιο μέρος του Χ
Α_Τ(Χ)	Απόλυτη τιμή του Χ

7.23 *Τι είναι αριθμητικές εκφράσεις;*

Όταν μια τιμή προκύπτει από υπολογισμό, τότε αναφερόμαστε σε εκφράσεις (expressions). Για τη σύνταξη μιας αριθμητικής έκφρασης χρησιμοποιούνται αριθμητικές σταθερές, μεταβλητές, συναρτήσεις, αριθμητικοί τελεστές και παρενθέσεις. Οι αριθμητικές εκφράσεις υλοποιούν απλές ή σύνθετες μαθηματικές πράξεις.
Κάθε έκφραση παριστάνει μια συγκεκριμένη αριθμητική τιμή, η οποία βρίσκεται μετά την εκτέλεση των πράξεων. Γι' αυτό είναι απαραίτητο όλες οι μεταβλητές, που εμφανίζονται σε μια έκφραση να έχουν οριστεί προηγούμενα, δηλαδή να έχουν κάποια τιμή.

7.24 *Ποια είναι η ιεραρχία των πράξεων*

Οι πράξεις που παρουσιάζονται σε μια έκφραση, εκτελούνται σύμφωνα με την επόμενη ιεραρχία
1. Ύψωση σε δύναμη
2. Πολλαπλασιασμός και διαίρεση
3. Πρόσθεση και αφαίρεση

Όταν η ιεραρχία είναι ίδια, τότε οι πράξεις εκτελούνται από τ' αριστερά προς τα δεξιά. Σε πολλές όμως περιπτώσεις είναι απαραίτητο να προηγηθεί μια πράξη χαμηλότερης ιεραρχίας. Αυτό επιτυγχάνεται με την εισαγωγή των παρενθέσεων.

7.25 *Τι είναι η εντολή εκχώρησης*

Η εντολή εκχώρησης χρησιμοποιείται για την απόδοση τιμών στις μεταβλητές κατά τη διάρκεια εκτέλεσης του προγράμματος.
Σε μια εντολή εκχώρησης η μεταβλητή και η έκφραση πρέπει να είναι του ιδίου τύπου.

7.26 *Πως συντάσσεται η εντολή εκχώρησης;*

Σύνταξη
Όνομα-Μεταβλητής ← έκφραση

Μια εντολή εκχώρησης σε καμιά 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.

7.27 *Ποιος ο σκοπός των εντολών εισόδου εξόδου;*

Σχεδόν όλα τα προγράμματα υπολογιστή δέχονται κάποια δεδομένα, τα επεξεργάζονται, υπολογίζουν τα αποτελέσματα και τέλος τα εμφανίζουν.
Τα δεδομένα εισάγονται κατά τη διάρκεια της εκτέλεσης του προγράμματος από μία μονάδα εισόδου, για παράδειγμα το πληκτρολόγιο και τα αποτελέσματα γράφονται σε μία μονάδα εξόδου, για παράδειγμα την οθόνη.

7.28 *Ποιες είναι οι εντολές εισόδου-εξόδου για την ΓΛΩΣΣΑ*

Η ΓΛΩΣΣΑ υποστηρίζει για την εισαγωγή δεδομένων από το πληκτρολόγιο την εντολή ΔΙΑΒΑΣΕ και για την εμφάνιση των αποτελεσμάτων την εντολή ΓΡΑΨΕ.
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.
Η εντολή ΓΡΑΨΕ έχει ως αποτέλεσμα την εμφάνιση τιμών στη μονάδα εξόδου. Συσκευή εξόδου μπορεί να είναι η οθόνη του υπολογιστή, ο εκτυπωτής, βοηθητική μνήμη ή γενικά οποιαδήποτε συσκευή εξόδου έχει οριστεί στο πρόγραμμα. Για τα παραδείγματα αυτού του κεφαλαίου θεωρούμε ότι η εμφάνιση γίνεται πάντοτε στην οθόνη. Η λίστα των στοιχείων μπορεί να περιέχει σταθερές τιμές και ονόματα μεταβλητών.
Κατά την εκτέλεση του προγράμματος η εντολή ΓΡΑΨΕ προκαλεί την εμφάνιση στην οθόνη των σταθερών τιμών. Όταν κάποιο όνομα μεταβλητής περιέχεται στη λίστα τότε αρχικά ανακτάται η τιμή της και στη συνέχεια η τιμή αυτή εμφανίζεται στην οθόνη.
Η χρήση της εντολής ΓΡΑΨΕ είναι κυρίως η εμφάνιση μηνυμάτων από τον υπολογιστή, καθώς και αποτελεσμάτων που περιέχονται στις μεταβλητές.

7.29 *Ποια η δομή προγράμματος της ΓΛΩΣΣΑΣ*

Όπως 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 ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ.

Τέλος αν το πρόγραμμα χρησιμοποιεί διαδικασίες, 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.

Τεστ αξιολόγησης επίδοσης

Οι ερωτήσεις του τεστ τόσο σε αυτό το κεφάλαιο όσο και στα επόμενα αναφέρονται στη ΓΛΩΣΣΑ, η οποία παρουσιάζεται στη θεωρία και περιλαμβάνεται στο βιβλίο. Οι ερωτήσεις όμως μπορούν να μετατραπούν εύκολα έτσι, ώστε, να αναφέρονται στην πραγματική γλώσσα προγραμματισμού η οποία χρησιμοποιείται στο εργαστήριο.

Συμπληρώστε με σωστό ή λάθος

1) Οι τύποι μεταβλητών που δέχεται η ΓΛΩΣΣΑ είναι μόνο ΠΡΑΓΜΑΤΙΚΕΣ και ΑΚΕΡΑΙΕΣ.
2) Οι δηλώσεις των σταθερών προηγούνται πάντοτε των δηλώσεων των μεταβλητών.
3) Τα σχόλια τοποθετούνται πάντα στην αρχή του προγράμματος.

Επιλέξτε μεταξύ των προτεινόμενων μία σωστή απάντηση.

4) Ποια από τις παρακάτω εντολές αυξάνει τη μεταβλητή **Πλήθος** κατά μία μονάδα
   Α) Πλήθος<-Πλήθος+1
   Β) Πλήθος<- +1
   Γ) Πλήθος<- 1
   Δ) Πλήθος+1<-Πλήθος

5) Ποια η τιμή της μεταβλητής Α μετά την εκτέλεση της παρακάτω εντολής:

Α ← (5+4/2*2)*2-(3*2+5-3)^2+9/3-2

   Α) -53
   Β) -37
   Γ) -125
   Δ) –45

Να συμπληρωθούν τα κενά

6)

```
ΠΡΟΓΡΑΜΜΑ Τεστ
…….
Π=3.14
ΜΕΤΑΒΛΗΤΕΣ
        ……:Ε, Ακτίνα
ΑΡΧΗ
        ΔΙΑΒΑΣΕ Ακτίνα
        Ε <-Π*Ακτίνα^2
        ΓΡΑΨΕ 'Εμβαδό :', ……..
……..
```

Ερωτήσεις - Θέματα για συζήτηση

1) Ποιους τύπους δεδομένων γνωρίζετε. Αναφέρατε δύο παραδείγματα για κάθε τύπο;

2) Σε ποια θέση του προγράμματος αναγράφονται οι δηλώσεις των σταθερών;

3) Ποια η διαφορά μεταβλητών και σταθερών;

4) Ποια η σειρά εκτέλεσης των πράξεων;

5) Ποιος ο σκοπός των εντολών εισόδου εξόδου;

6) Ποια η διαφορά των εντολών ΔΙΑΒΑΣΕ και ΓΡΑΨΕ;

7) Περιγράψτε τη δομή ενός προγράμματος;

7. Επιλογή και επανάληψη

8.1 Τι είναι εντολές επιλογής

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

8.2 Πως συντάσσεται 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.

8.3 *Ποιοι είναι οι συγκριτικοί τελεστές*

Οι χρησιμοποιούμενοι συγκριτικοί τελεστές παρουσιάζονται στον επόμενο πίνακα.

Συγκριτικοί τελεστές		
Τελεστής	Ελεγχόμενη σχέση	Παράδειγμα
=	Ισότητα	Αριθμός=0
<>	Ανισότητα	Ονομα1 <> 'Κώστας'
>	Μεγαλύτερο από	Τιμή>10000
>=	Μεγαλύτερο ή ίσο	Χ+Υ >= (Α+Β)/Γ
<	Μικρότερο από	Β^2-4*Α*Γ < 0
<=	Μικρότερο ή ίσο	Βάρος <= 500

8.4 Πώς γίνονται οι συγκρίσεις στα διάφορα είδη 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.

8.5 Ποια η προτεραιότητα ανάμεσα σε αριθμητικούς και συγκριτικούς τελεστές;

Όταν αριθμητικοί και συγκριτικοί τελεστές συνδυάζονται σε μια έκφραση, οι αριθμητικές πράξεις εκτελούνται πρώτες.

8.6 Πως επιτυγχάνεται η δημιουργία σύνθετων εκφράσεων; Να δοθούν δυο παραδείγματα.

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

0<Χ<5	Χ>0 ΚΑΙ Χ<5
Χ=1 ή 2 ή 3	Χ=1 Ή Χ=2 Ή Χ=3

Η ιεραρχία των λογικών τελεστών είναι μικρότερη των αριθμητικών.

8.7 Να αναλυθεί η εντολή ΑΝ

Η δομή επιλογής υλοποιείται στη ΓΛΩΣΣΑ με την εντολή ΑΝ. Η εντολή ΑΝ εμφανίζεται με τρεις διαφορετικές μορφές.

Την απλή εντολή **ΑΝ...ΤΟΤΕ**, την εντολή **ΑΝ...ΤΟΤΕ...ΑΛΛΙΩΣ** και τέλος την εντολή ΑΝ**...ΤΟΤΕ...ΑΛΛΙΩΣ ΑΝ**. Κάθε εντολή ΑΝ πρέπει να κλείνει με ΤΕΛΟΣ_ΑΝ.

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

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

```
ΔΙΑΒΑΣΕ α
ΑΝ α >=0 ΤΟΤΕ
        Ρίζα ← Τ_Ρ(α)
ΤΕΛΟΣ_ΑΝ
```

8.8 Πως συντάσσεται η ΑΝ ... ΤΟΤΕ

ΑΝ συνθήκη **ΤΟΤΕ**
εντολή-1
εντολή-2
…
εντολή-ν
ΤΕΛΟΣ_ΑΝ

8.9 Πως συντάσσεται η ΑΝ ... ΤΟΤΕ ... ΑΛΛΙΩΣ

ΑΝ συνθήκη **ΤΟΤΕ**
    εντολή-1
    εντολή-2
    …
    εντολή-ν
ΑΛΛΙΩΣ
    εντολή-1
    εντολή-2
    …
    εντολή-ν
ΤΕΛΟΣ_ΑΝ

8.10 Πως συντάσσεται η ΑΝ ... ΤΟΤΕ ... ΑΛΛΙΩΣ_ΑΝ

ΑΝ συνθήκη-1 **ΤΟΤΕ**
    εντολή-1
    εντολή-2
    …
    εντολή-ν
ΑΛΛΙΩΣ_ΑΝ συνθήκη-2 **ΤΟΤΕ**
    εντολή-1
    εντολή-2
    …
    εντολή-ν
ΑΛΛΙΩΣ
    εντολή-1
    εντολή-2
    …
    εντολή-ν
ΤΕΛΟΣ_ΑΝ

8.11 Τι ονομάζονται εμφωλευμένα ΑΝ

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

8.12 Πως συντάσσεται η εντολή ΕΠΙΛΕΞΕ και πότε χρησιμοποιείται;

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

ΕΠΙΛΕΞΕ έκφραση
    ΠΕΡΙΠΤΩΣΗ λίστα_τιμών_1
        εντολές_1
    ΠΕΡΙΠΤΩΣΗ λίστα_τιμών_2
        εντολές_2
    .....
    ΠΕΡΙΠΤΩΣΗ ΑΛΛΙΩΣ
        εντολές_αλλιώς
ΤΕΛΟΣ_ΕΠΙΛΟΓΩΝ

8.13 Ποιο το κύριο χαρακτηριστικό της εντολής ΕΠΙΛΕΞΕ

Το κύριο χαρακτηριστικό της εντολής **ΕΠΙΛΕΞΕ** είναι η συμπαγής δομή της και γι' αυτό προσφέρει σημαντικά πλεονεκτήματα στον προγραμματισμό.

8.14 Τι είναι δομή επανάληψης ή βρόχος;

Η δομή επανάληψης, ή βρόχος, είναι η δομή 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.

8.15 Ποιες δομές επανάληψης υποστηρίζει η ΓΛΩΣΣΑ;

Η ΓΛΩΣΣΑ υποστηρίζει τρεις εντολές επανάληψης,

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

8.16 Πως συντάσσεται η εντολή ΟΣΟ...ΕΠΑΝΑΛΑΒΕ;

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

Σύνταξη

ΟΣΟ συνθήκη **ΕΠΑΝΑΛΑΒΕ**
    εντολή-1
    εντολή-2
    ...
    εντολή-ν
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

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

8.17 Τι ονομάζουμε τιμή φρουρός;

Η χρήση τιμών για τον τερματισμό μίας επαναληπτικής διαδικασίας, 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. Η τιμή αυτή συχνά αποκαλείται "τιμή φρουρός".

8.18 Πως συντάσσεται η εντολή ΜΕΧΡΙΣ_ΟΤΟΥ;

Σε αυτή οι εντολές του βρόχου εκτελούνται μέχρις ότου ικανοποιηθεί κάποια συνθήκη η οποία ελέγχεται στο τέλος της επανάληψης.
Πολύ συχνά η ίδια επαναληπτική διαδικασία μπορεί να γραφεί εξίσου σωστά χρησιμοποιώντας είτε τη δομή ΟΣΟ...ΕΠΑΝΑΛΑΒΕ είτε τη δομή ΜΕΧΡΙΣ_ΟΤΟΥ και είναι προσωπική επιλογή του προγραμματιστή ποια από τις δυο θα χρησιμοποιήσει. Υπάρχουν όμως περιπτώσεις όπου η χρήση της εντολής ΜΕΧΡΙΣ_ΟΤΟΥ οδηγεί σε απλούστερα και πιο ευκολονόητα προγράμματα.
Γενικά σε περιπτώσεις όπου η επανάληψη θα συμβεί υποχρεωτικά μία φορά, είναι προτιμότερη η χρήση της ΜΕΧΡΙΣ_ΟΤΟΥ.
Χαρακτηριστική περίπτωση όπου προτιμάται η εντολή ΜΕΧΡΙΣ_ΟΤΟΥ είναι στον έλεγχο αποδεκτών τιμών καθώς και στην επιλογή από προκαθορισμένες απαντήσεις ή μενού.

Σύνταξη

ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
    εντολή-1
    εντολή-2
    …
    εντολή-ν
ΜΕΧΡΙΣ_ΟΤΟΥ λογική-έκφραση

Όταν η λογική έκφραση γίνει Αληθής τότε σταματάει η επανάληψη και εκτελείται η εντολή μετά από την ΜΕΧΡΙΣ_ΟΤΟΥ.
Η εντολή επανάληψης ΜΕΧΡΙΣ_ΟΤΟΥ εκτελείται υποχρεωτικά τουλάχιστον μία φορά

8.19 Πως συντάσσεται η εντολή ΓΙΑ...ΑΠΟ...ΜΕΧΡΙ;

Πολύ συχνά ο αριθμός των επαναλήψεων που πρέπει να εκτελεστούν, είναι γνωστός από την αρχή. Αν και αυτού του είδους οι επαναλήψεις μπορούν να αντιμετωπιστούν με τη χρήση των προηγούμενων εντολών επανάληψης, η ΓΛΩΣΣΑ διαθέτει και την εντολή ΓΙΑ. Η εντολή αυτή χειρίζεται μια μεταβλητή, στην οποία αρχικά εκχωρείται η αρχική τιμή. Η τιμή της μεταβλητής συγκρίνεται με την τελική τιμή και εφόσον είναι μικρότερη από αυτή, τότε εκτελούνται οι εντολές που βρίσκονται στο βρόχο (ανάμεσα στις εντολές ΓΙΑ και ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ). Στη συνέχεια η μεταβλητή ελέγχου αυξάνεται κατά την τιμή που ορίζει το ΒΗΜΑ. Αν η νέα τιμή είναι μικρότερη της τελικής, τότε ο βρόχος εκτελείται ξανά. Η διαδικασία αυτή επαναλαμβάνεται συνεχώς, έως ότου η τιμή ελέγχου γίνει μεγαλύτερη της τελικής τιμής, οπότε η τερματίζεται η επανάληψη και το πρόγραμμα συνεχίζει με την εντολή που ακολουθεί το ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ.
Ας σημειωθεί ότι, αν η τιμή του βήματος είναι 1, τότε μπορεί να παραληφθεί.
Η εντολή ΓΙΑ...ΑΠΟ...ΜΕΧΡΙ χρησιμοποιείται στην περίπτωση που πρέπει να επαναληφθεί η εκτέλεση κάποιων εντολών για προκαθορισμένο αριθμό επαναλήψεων.

Σύνταξη

ΓΙΑ μεταβλητή **ΑΠΟ** τιμή1 **ΜΕΧΡΙ** τιμή2 **ΜΕ_ΒΗΜΑ** τιμή3

εντολή-1
εντολή-2
…
εντολή-ν
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

8.20 Ποια δομή επανάληψης μπορεί να υλοποιηθεί πάντα με την χρήση των άλλων δομών επανάληψης;

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

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

8.21 Ποιοι οι κανόνες χρήσης των εμφωλευμένων βρόχων;

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

Τεστ αξιολόγησης επίδοσης

Συμπληρώστε με σωστό ή λάθος

1) Κάθε εντολή ΑΝ περιλαμβάνει υποχρεωτικά το τμήμα ΑΛΛΙΩΣ.
2) Κάθε τμήμα προγράμματος που χρησιμοποιεί την εντολή ΕΠΙΛΕΞΕ μπορεί να γραφεί και με εντολές ΑΝ.
3) Η χρήση εμφωλευμένων ΑΝ είναι καλή προγραμματιστική τακτική.
4) Αν το Α έχει την τιμή 10 και το Β την τιμή 20 τότε η έκφραση (Α >8 ΚΑΙ Β<20) Ή (Α>10 Ή Β=10) είναι αληθής.
5) Οι εντολές που βρίσκονται σε μία επανάληψη ΓΙΑ εκτελούνται τουλάχιστο μία φορά.
6) Κάθε επανάληψη μπορεί να γραφεί με την εντολή ΟΣΟ- ΕΠΑΝΑΛΑΒΕ.

7) Σε περίπτωση εμφωλευμένων βρόχων,ο εσωτερικός πρέπει να περικλείεται ολόκληρος στον εξωτερικό.

8) Η τιμή του βήματος αναφέρεται υποχρεωτικά σε κάθε εντολή ΓΙΑ.

Επιλέξτε μεταξύ των προτεινόμενων μία σωστή απάντηση.

9) Τι θα εκτυπώσουν οι παρακάτω εντολές:

```
Α←0
Β←5
Γ←10
ΑΝ Α>10 ΤΟΤΕ
    ΑΝ Β >20 ΤΟΤΕ
            ΑΝ Γ >10 ΤΟΤΕ
                    ΓΡΑΨΕ Γ
            ΑΛΛΙΩΣ
                    ΓΡΑΨΕ 2*Γ
            ΤΕΛΟΣ_ΑΝ
    ΑΛΛΙΩΣ
            ΓΡΑΨΕ Β
    ΤΕΛΟΣ_ΑΝ
```

```
    ΑΛΛΙΩΣ
        ΑΝ Β <10 ΤΟΤΕ
                ΓΡΑΨΕ Α
        ΤΕΛΟΣ_ΑΝ
    ΤΕΛΟΣ_ΑΝ
```

Α. 0
Β. 10
Γ. 5
Δ. 20

10) Πόσες φορές θα εκτελεστεί η εντολή ΓΡΑΨΕ Α

```
Α←10
ΟΣΟ Α<>0 ΕΠΑΝΑΛΑΒΕ
ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 5
Α←Α-1
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΓΡΑΨΕ Α
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

Α. 10
Β. 0
Γ. 2
Δ. Άπειρες

11) Να συμπληρωθούν **τα κενά** ώστε οι επόμενες εντολές να τυπώνουν πάντα τον μεγαλύτερο αριθμό από τους δύο που διαβάστηκαν.

```
    ΔΙΑΒΑΣΕ Α,Β
        ΑΝ Α<Β …
        ……
        ΤΕΛΟΣ_ΑΝ
    ΓΡΑΨΕ Α
```

12) Να συμπληρωθούν **τα κενά** ώστε οι επόμενες εντολές να τυπώνουν την τετραγωνική ρίζα.

```
        ΔΙΑΒΑΣΕ Α
                ΑΝ Α…0 ΤΟΤΕ
                        Ρίζα ← Τ_Ρ(Α)
                        ΓΡΑΨΕ Ρίζα
                ……
                        ΓΡΑΨΕ ‘Δεν υπάρχει ρίζα’
                ΤΕΛΟΣ_ΑΝ
```

13) Να **συμπληρωθούν τα κενά** ώστε οι επόμενες εντολές να τυπώνουν το άθροισμα των τετραγώνων των περιττών αριθμών που είναι μικρότεροι από 10.

```
Άθροισμα<-…
ΓΙΑ… ΑΠΟ 1 ΜΕΧΡΙ 10 ΜΕ ΒΗΜΑ …
Άθροισμα← … + Ι^2
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΓΡΑΨΕ Άθροισμα
```

14) Να συμπληρωθούν τα κενά ώστε οι επόμενες εντολές να τυπώνουν το άθροισμα των αριθμών από 100 έως 200

```
Κ←…
Σ←…
        ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
                Σ←Σ+Κ
```

```
            Κ←Κ+1
        ΜΕΧΡΙΣ_ΟΤΟΥ ……
ΓΡΑΨΕ Σ
```

Ερωτήσεις - Θέματα για συζήτηση

1. Ποιες είναι οι τιμές που μπορεί να πάρει μία λογική έκφραση;

2. Ποιοι είναι οι βασικοί λογικοί τελεστές; Αναφέρατε δύο παραδείγματα για τη χρήση του καθενός;

3. Ποια είναι η σύνταξη της εντολής ΑΝ;

4. Ποια είναι η διαφορά της εντολής ΑΝ- ΑΛΛΙΩΣ και της ΑΝ- ΑΛΛΙΩΣ_ΑΝ;

5. Τι είναι τα εμφωλευμένα ΑΝ;

6. Πότε χρησιμοποιείται η εντολή ΕΠΙΛΕΞΕ;

7. Ποιες οι εντολές επανάληψης;

8. Ποιες οι διαφορές της εντολής ΟΣΟ και της εντολής ΜΕΧΡΙΣ_ΟΤΟΥ;

9. Πώς συντάσσεται η εντολή ΓΙΑ;

10. Ποια η βασική διαφορά της εντολής ΓΙΑ από τις άλλες δύο εντολές επανάληψης;

8. Επιλογή και επανάληψη

9.1 Τι είναι 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.

9.2 Ποια τα χαρακτηριστικά του πίνακα;

Το όνομα του πίνακα μπορεί να είναι οποιοδήποτε δεκτό όνομα της ΓΛΩΣΣΑΣ και ο δείκτης είναι μία ακέραια έκφραση, σταθερή ή μεταβλητή που περικλείεται μέσα στα σύμβολα [και]. Το όνομα του πίνακα καθορίζει μία ομάδα διαδοχικών θέσεων στη μνήμη.
Κάθε πίνακας πρέπει υποχρεωτικά να περιέχει δεδομένα του ιδίου τύπου, δηλαδή ακέραια, πραγματικά, λογικά, ή αλφαριθμητικά. Ο τύπος του πίνακα δηλώνεται μαζί με τις άλλες μεταβλητές του προγράμματος στο τμήμα δήλωσης μεταβλητών. Εκτός από τον τύπο του πίνακα πρέπει να δηλώνεται και ο αριθμός των στοιχείων που περιέχει ή καλύτερα ο μεγαλύτερος αριθμός στοιχείων που μπορεί να έχει ο συγκεκριμένος πίνακας και αυτό για να δεσμευτούν οι αντίστοιχες συνεχόμενες θέσεις μνήμης.

9.3 Τι είναι μονοδιάστατος πίνακας;

Οι πίνακες που χρησιμοποιούν ένα μόνο δείκτη για την αναφορά των στοιχείων τους, ονομάζονται μονοδιάστατοι πίνακες.

Η καλύτερη λύση στο πρόβλημα αυτό είναι η χρήση μεταβλητής με δείκτες, έννοια που είναι γνωστή από τα μαθηματικά και υλοποιείται στον προγραμματισμό με τη δομή δεδομένων του πίνακα. Χρησιμοποιείται λοιπόν μόνο ένα όνομα Θερμοκρασία, που αναφέρεται και στις τριάντα διαφορετικές θερμοκρασίες.
Κάθε συγκεκριμένη θέση μνήμης καλείται στοιχείο του πίνακα και προσδιορίζεται από την τιμή ενός δείκτη.
Ο δείκτης είναι μία μεταβλητή που μπορεί να έχει οποιοδήποτε δεκτό όνομα. Είναι σύνηθες όμως στον προγραμματισμό ως δείκτες να χρησιμοποιούνται οι μεταβλητές i,j,k.
Η ανάγνωση, η επεξεργασία και η εκτύπωση 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

9.4 Πότε πρέπει να χρησιμοποιούνται πίνακες

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

9.5 Ποια είναι τα μειονεκτήματα από την χρήση πινάκων;

- Οι πίνακες απαιτούν μνήμη. Κάθε πίνακας δεσμεύει από την αρχή του προγράμματος πολλές θέσεις μνήμης. Σε ένα μεγάλο και σύνθετο πρόγραμμα η άσκοπη χρήση μεγάλων πινάκων μπορεί να οδηγήσει ακόμη και σε αδυναμία εκτέλεσης του προγράμματος.
- Οι πίνακες περιορίζουν τις δυνατότητες του προγράμματος. Γιατί οι πίνακες είναι στατικές δομές και το μέγεθος τους πρέπει να δηλώνεται στην αρχή του προγράμματος, ενώ παραμένει υποχρεωτικά σταθερό κατά την εκτέλεση του προγράμματος.

9.6 Ποιο το κριτήριο χρήσης των πινάκων;

Η απόφαση για την χρήση ή όχι πίνακα για την διαχείριση των δεδομένων είναι κυρίως θέμα εμπειρίας στον προγραμματισμό.
Γενικά, αν τα δεδομένα που εισάγονται σε ένα πρόγραμμα πρέπει να διατηρούνται στη μνήμη μέχρι το τέλος της εκτέλεσης, τότε η χρήση πινάκων βοηθάει ή συχνά είναι απαραίτητη για την επίλυση του προβλήματος.
Σε άλλη περίπτωση μπορεί να αποφεύγεται η χρήση τους.

9.7 Τι είναι πολυδιάστατοι πίνακες;

Εκτός από μονοδιάστατους και δισδιάστατους πίνακες υπάρχουν πίνακες με περισσότερες διαστάσεις τρισδιάστατοι, τετραδιάστατοι και γενικά πολυδιάστατοι, ανάλογα με τον αριθμό των δεικτών που χρησιμοποιούνται για τον καθορισμό των στοιχείων. Ωστόσο τα περισσότερα προβλήματα αντιμετωπίζονται με τη χρήση πινάκων μονοδιάστατων ή δισδιάστατων.

9.8 Πως γίνεται η ανάγνωση, η επεξεργασία καθώς και η εκτύπωση των στοιχείων πολυδιάστατων πινάκων;

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

9.9 Ποιες είναι οι τυπικές επεξεργασίες πινάκων

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

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

9.10 Τι εννοούμε 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.

9.11 Τι εννοούμε 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.

9.12 Τι γνωρίζεται για την ταξινόμηση των στοιχείων του πίνακα.

Η μέθοδος ταξινόμησης της ευθείας ανταλλαγής, 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.

9.13 Ποιοι είναι οι πιο διαδεδομένοι αλγόριθμοι αναζήτησης; Ποιες οι διαφορές τους;

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.

9.14 Τι γνωρίζεται για την 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.

Τεστ αξιολόγησης επίδοσης

Συμπληρώστε με σωστό ή λάθος

1) Κάθε εντολή ΑΝ περιλαμβάνει υποχρεωτικά το τμήμα ΑΛΛΙΩΣ.

2) Κάθε τμήμα προγράμματος που χρησιμοποιεί την εντολή ΕΠΙΛΕΞΕ μπορεί να γραφεί και με εντολές ΑΝ.

3) Η χρήση εμφωλευμένων ΑΝ είναι καλή προγραμματιστική τακτική.

4) Αν το Α έχει την τιμή 10 και το Β την τιμή 20 τότε η έκφραση
(Α >8 ΚΑΙ Β<20) Ή (Α>10 Ή Β=10) είναι αληθής.

Επιλέξτε μεταξύ των προτεινόμενων μία σωστή απάντηση.

5) Τι θα εκτυπώσουν οι παρακάτω εντολές:

```
Α<-0
Β<-5
Γ<-10
ΑΝ Α>10 ΤΟΤΕ
    ΑΝ Β >20 ΤΟΤΕ
            ΑΝ Γ >10 ΤΟΤΕ
                    ΓΡΑΨΕ Γ
            ΑΛΛΙΩΣ
                    ΓΡΑΨΕ 2*Γ
            ΤΕΛΟΣ_ΑΝ
    ΑΛΛΙΩΣ
            ΓΡΑΨΕ Β
    ΤΕΛΟΣ_ΑΝ
ΑΛΛΙΩΣ
    ΑΝ Β <10 ΤΟΤΕ
            ΓΡΑΨΕ Α
    ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΑΝ
```

Α. 0
Β. 10
Γ. 5
Δ. 20

6) Να συμπληρωθούν **τα κενά** ώστε οι επόμενες εντολές να τυπώνουν πάντα τον μεγαλύτερο αριθμό από τους δύο που διαβάστηκαν.

```
ΔΙΑΒΑΣΕ Α,Β
        ΑΝ Α<Β …
                ……
        ΤΕΛΟΣ_ΑΝ
```

```
        ΓΡΑΨΕ Α
```

Να **συμπληρωθούν τα κενά** ώστε οι επόμενες εντολές να τυπώνουν την τετραγωνική ρίζα.

```
        ΔΙΑΒΑΣΕ Α
                ΑΝ Α…0 ΤΟΤΕ
                        Ρίζα ← Τ_Ρ(Α)
                        ΓΡΑΨΕ Ρίζα
                        ……
                        ΓΡΑΨΕ 'Δεν υπάρχει ρίζα'
                ΤΕΛΟΣ_ΑΝ
```

Τεστ αξιολόγησης επίδοσης

Οι ερωτήσεις του τεστ αναφέρονται στη ΓΛΩΣΣΑ η οποία παρουσιάζεται στη θεωρία και περιλαμβάνεται στο βιβλίο.
Οι ερωτήσεις όμως μπορούν να μετατραπούν εύκολα έτσι ώστε να αναφέρονται στην πραγματική γλώσσα προγραμματισμού η οποία χρησιμοποιείται στο εργαστήριο.

Συμπληρώστε με σωστό ή λάθος

1) Οι εντολές που βρίσκονται σε μία επανάληψη ΓΙΑ εκτελούνται τουλάχιστο μία φορά.

2) Κάθε επανάληψη μπορεί να γραφεί με την εντολή ΟΣΟ- ΕΠΑΝΑΛΑΒΕ.

3) Σε περίπτωση εμφωλευμένων βρόχων,ο εσωτερικός πρέπει να περικλείεται ολόκληρος στον εξωτερικό.

4) Η τιμή του βήματος αναφέρεται υποχρεωτικά σε κάθε εντολή ΓΙΑ.

Επιλέξτε μεταξύ των προτεινόμενων μία σωστή απάντηση.

5) Πόσες φορές θα εκτελεστεί η εντολή ΓΡΑΨΕ Α

```
Α<-10
ΟΣΟ Α<>0 ΕΠΑΝΑΛΑΒΕ
        ΓΙΑ Ι ΑΠΟ 1 ΜΕΧΡΙ 5
                Α<-Α-1
        ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
        ΓΡΑΨΕ Α
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

Α. 10
Β. 0
Γ. 2
Δ. Άπειρες

6) Να **συμπληρωθούν τα κενά** ώστε οι επόμενες εντολές να τυπώνουν το άθροισμα των τετραγώνων των περιττών αριθμών που είναι μικρότεροι από

10.

```
Άθροισμα<-…
        ΓΙΑ… ΑΠΟ 1 ΜΕΧΡΙ 10 ΜΕ ΒΗΜΑ …
                Άθροισμα<- … + Ι^2
        ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΓΡΑΨΕ Άθροισμα
```

7. Να συμπληρωθούν τα κενά ώστε οι επόμενες εντολές να τυπώνουν το άθροισμα

των αριθμών από 100 έως 200
Κ<-…
Σ<-…
ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
Σ<-Σ+Κ
Κ<-Κ+1
ΜΕΧΡΙΣ_ΟΤΟΥ ……
ΓΡΑΨΕ Σ

Ερωτήσεις - Θέματα για συζήτηση

1. Τι ονομάζεται πίνακας;

2. Για ποιο λόγο χρησιμοποιούνται οι πίνακες;

3. Τι μπορεί να είναι οι δείκτες των πινάκων;

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

5. Πού ορίζεται η διάσταση του πίνακα; (να δοθεί ένα παράδειγμα)

6. Τι είδους δομή είναι ο πίνακας;

7. Ποιοι πίνακες ονομάζονται μονοδιάστατοι;

8. Δώσε ένα παράδειγμα ενός τρισδιάστατου πίνακα.

9. Πού αποθηκεύονται τα στοιχεία ενός πίνακα;

10. Ποια τα μειονεκτήματα της χρήσης πινάκων;

11. Ποιες οι τυπικές επεξεργασίες πίνακα;

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

9. Υποπρογράμματα

10.1 Τι είναι τμηματικός προγραμματισμός

Τμηματικός προγραμματισμός ονομάζεται η τεχνική σχεδίασης και ανάπτυξης των προγραμμάτων ως ένα σύνολο από απλούστερα τμήματα προγραμμάτων.
Δηλ. στον τμηματικό προγραμματισμό τα προγράμματα αποτελούνται από ένα σύνολο τμημάτων τα οποία επιλύουν τα επιμέρους υποπροβλήματα του αρχικού προβλήματος.

10.2 Τι είναι υποπρογράμματα;

Υποπρόγραμμα ονομάζεται κάθε τμήμα του προγράμματος που επιτελεί ένα αυτόνομο έργο και έχει γραφεί χωριστά από το υπόλοιπο πρόγραμμα.

10.3 Υπάρχει συγκεκριμένη μεθοδολογία για τον χωρισμό ενός προγράμματος σε υποπρογράμματα;

Ο χωρισμός ενός προγράμματος σε υποπρογράμματα προϋποθέτει την ανάλυση του αρχικού προβλήματος σε μικρότερα υποπροβλήματα, τα οποία να μπορούν να αντιμετωπισθούν ανεξάρτητα το ένα από το άλλο. Η ανάλυση όμως αυτή δεν είναι πάντα εύκολη και όπως ισχύει γενικά στον προγραμματισμό, δεν υπάρχουν συγκεκριμένοι κανόνες για την επιτυχή ανάλυση.
Η δυσκολία δε αυξάνεται όσο πιο μεγάλο και πιο σύνθετο είναι το πρόβλημα. Η σωστή εφαρμογή του τμηματικού προγραμματισμού απαιτεί μελέτη στην ανάλυση του προβλήματος, εμπειρία στον προγραμματισμό, ταλέντο και φυσικά γνώσεις.

10.4 Ποιες είναι οι βασικές ιδιότητες που διακρίνουν τα υποπρογράμματα;

Υπάρχουν τρεις ιδιότητες που πρέπει να διακρίνουν τα υποπρογράμματα:

- Κάθε υποπρόγραμμα έχει μόνο μία είσοδο και μία έξοδο.
  Στην πραγματικότητα κάθε υποπρόγραμμα ενεργοποιείται με την είσοδο σε αυτό που γίνεται πάντοτε από την αρχή του, εκτελεί ορισμένες ενέργειες, και απενεργοποιείται με την έξοδο από αυτό που γίνεται πάντοτε από το τέλος του.
  Όταν λέμε ότι τα υποπρογράμματα έχουν μια μόνο είσοδο και έξοδο εννοούμε ότι ένα υποπρόγραμμα κάθε φορά που χρησιμοποιείται, δέχεται τιμές από ένα μόνο πρόγραμμα και επιστρέφει τιμές σε ένα μόνο πρόγραμμα.
- Κάθε υποπρόγραμμα πρέπει να είναι ανεξάρτητο από τα άλλα.
  Αυτό σημαίνει ότι κάθε υποπρόγραμμα μπορεί να σχεδιαστεί, να αναπτυχθεί και να συντηρηθεί αυτόνομα χωρίς να επηρεαστούν άλλα υποπρογράμματα. Στην πράξη βέβαια η απόλυτη ανεξαρτησία είναι δύσκολο να επιτευχθεί.
- Κάθε υποπρόγραμμα πρέπει να μην είναι πολύ μεγάλο.
  Η έννοια του μεγάλου προγράμματος είναι υποκειμενική, αλλά πρέπει κάθε υποπρόγραμμα να είναι τόσο, ώστε να είναι εύκολα κατανοητό για να μπορεί να ελέγχεται. Γενικά κάθε υποπρόγραμμα πρέπει να εκτελεί μόνο μία λειτουργία. Αν εκτελεί περισσότερες λειτουργίες, τότε συνήθως μπορεί και πρέπει να διασπαστεί σε ακόμη μικρότερα υποπρογράμματα.

10.5 Ποια τα πλεονεκτήματα του τμηματικού προγραμματισμού;

1. 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.

2. 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. Όπως έχει πολλές φορές τονιστεί αυτό είναι πολύ σημαντικό χαρακτηριστικό του σωστού προγραμματισμού, αφού ένα μεγάλο πρόγραμμα στον κύκλο της ζωής του χρειάζεται να συντηρηθεί από διαφορετικούς προγραμματιστές.

3. 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. Έτσι 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.

4. Επεκτείνει τις δυνατότητες των γλωσσών προγραμματισμού.
   Ένα 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.

10.6 Τι είναι παράμετροι σ' ένα υποπρόγραμμα;

Οι παράμετροι είναι σαν τις κοινές μεταβλητές ενός προγράμματος με μία ουσιώδη διαφορά, χρησιμοποιούνται για να περνούν τιμές στα υποπρογράμματα.
Μία παράμετρος είναι μία μεταβλητή που επιτρέπει το πέρασμα της τιμής της από ένα τμήμα προγράμματος σε ένα άλλο.

10.7 Πως ενεργοποιούνται τα υποπρογράμματα από κάποιο άλλο πρόγραμμα;

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. 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 και να επιστρέφουν στο πρόγραμμα τα αποτελέσματα.

10.8 Ποιες κατηγορίες υποπρογραμμάτων υπάρχουν και ποιες οι διαφορές τους;

Υπάρχουν δύο ειδών υποπρογράμματα, οι διαδικασίες και οι συναρτήσεις.

10.9 Τι είναι διαδικασίες;

Οι διαδικασίες είναι ένας τύπος υποπρογράμματος που μπορούν να εκτελούν όλες τις λειτουργίες ενός προγράμματος και να επιστρέφουν πολλές τιμές και να παράγουν πολλά μηνύματα.

10.10 Ποια η δομή μιας διαδικασίας;

Κάθε διαδικασία έχει την ακόλουθη δομή.

ΔΙΑΔΙΚΑΣΙΑ Όνομα (λίστα παραμέτρων)
Τμήμα δηλώσεων
ΑΡΧΗ
εντολές
ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ

Το όνομα της διαδικασίας είναι οποιοδήποτε έγκυρο όνομα της ΓΛΩΣΣΑΣ.
Η λίστα παραμέτρων είναι μια λίστα μεταβλητών, των οποίων οι τιμές μεταβιβάζονται προς τη διαδικασία κατά την κλήση ή/και επιστρέφονται στο κύριο πρόγραμμα μετά το τέλος της διαδικασίας. Στο σώμα της διαδικασίας μπορούν να υπάρχουν οποιεσδήποτε εντολές της γλώσσας.

10.11 Πως γίνεται η κλήση (χρήση) μιας διαδικασίας από ένα πρόγραμμα;

Για να εκτελεστούν οι διαδικασίες χρησιμοποιείται η ειδική εντολή ΚΑΛΕΣΕ και το όνομα της διαδικασίας.
Κάθε διαδικασία εκτελείται όταν καλείται από το κύριο πρόγραμμα ή άλλη διαδικασία. Η κλήση σε διαδικασία πραγματοποιείται με την εντολή ΚΑΛΕΣΕ, που ακολουθείται από το όνομα της διαδικασίας συνοδευόμενο μέσα σε παρενθέσεις με τη λίστα παραμέτρων.
Η γενική μορφή της εντολής ΚΑΛΕΣΕ είναι

Σύνταξη

ΚΑΛΕΣΕ όνομα-διαδικασίας (λίστα-παραμέτρων)

Παράδειγμα

ΚΑΛΕΣΕ Πράξεις (Α, Β, Διαφορά)

Λειτουργία

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.

10.12 Να δοθεί παράδειγμα διαδικασίας

Εισαγωγή βαθμού κάποιου μαθητή και έλεγχος ορθότητας του στην εικοσαβάθμια κλίμακα.

```
ΔΙΑΔΙΚΑΣΙΑ Είσοδος(βαθμός)
ΜΕΤΑΒΛΗΤΕΣ
ΑΚΕΡΑΙΟΣ: βαθμός
ΑΡΧΗ
  ΕΜΦΑΝΙΣΕ 'Δώσε βαθμό'
  ΔΙΑΒΑΣΕ βαθμός
        ΟΣΟ βαθμός>20 Ή βαθμός <0 ΕΠΑΝΑΛΑΒΕ
              ΕΜΦΑΝΙΣΕ 'Λάθος πληκτρολόγηση – Ξαναδώσε βαθμό'
              ΔΙΑΒΑΣΕ βαθμός
        ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ
```

10.13 Δώστε ένα παράδειγμα πώς καλείται μια διαδικασία από ένα πρόγραμμα.

Πρόγραμμα που υπολογίζει και εμφανίζει το μέσο όρο 20 μαθητών χρησιμοποιώντας το παραπάνω υποπρόγραμμα.

```
ΠΡΟΓΡΑΜΜΑ Μέσος_Όρος
ΜΕΤΑΒΛΗΤΕΣ
ΑΚΕΡΑΙΟΣ: α, i, sum
ΠΡΑΓΜΑΤΙΚΟΣ: ΜΟ
```

```
ΑΡΧΗ
        sum←0
        ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 20
                ΚΑΛΕΣΕ Είσοδος(α)
                sum←sum+α
        ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
                ΜΟ←sum/20
        ΓΡΑΨΕ ΜΟ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
ΔΙΑΔΙΚΑΣΙΑ Είσοδος(βαθμός)
ΜΕΤΑΒΛΗΤΕΣ
ΑΚΕΡΑΙΟΣ: βαθμός
ΑΡΧΗ
        ΕΜΦΑΝΙΣΕ 'Δώσε βαθμό'
        ΔΙΑΒΑΣΕ βαθμός
                ΟΣΟ βαθμός>20 Ή βαθμός <0 ΕΠΑΝΑΛΑΒΕ
                        ΕΜΦΑΝΙΣΕ 'Λάθος πληκτρολόγηση – Ξαναδώσε βαθμό'
                        ΔΙΑΒΑΣΕ βαθμός
                ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ
```

10.14 Ποια η χρησιμότητα των διαδικασιών;

Σε ένα πρόγραμμα πολύ συχνά παρατηρείται η ίδια λειτουργία. Χρησιμοποιώντας λοιπόν ένα υποπρόγραμμα και συγκεκριμένα μια διαδικασία που έχει ήδη γραφεί, μπορεί αυτή να καλείται σε πολλά σημεία του προγράμματος. Συνέπειες αυτού είναι η μείωση του μεγέθους του, καθώς και του χρόνου που απαιτείται για την υλοποίηση του.

10.15 Δώστε ένα παράδειγμα χρησιμότητας των διαδικασιών.

Πρόγραμμα που υπολογίζει και εμφανίζει τους μέσους όρους σε 5 μαθήματα, 20 μαθητών ενός τμήματος.

```
ΠΡΟΓΡΑΜΜΑ Μέσος_Όρος
ΜΕΤΑΒΛΗΤΕΣ
ΑΚΕΡΑΙΟΣ: α, b, c, d, e, i, sum1, sum2, sum3, sum4, sum5
ΠΡΑΓΜΑΤΙΚΟΣ: ΜΟ1, ΜΟ2, ΜΟ3, ΜΟ4, ΜΟ5
ΑΡΧΗ
        sum1←0, sum1←0, sum1←0, sum1←0, sum1←0,
        ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 20
                ΚΑΛΕΣΕ Είσοδος(α)
                ΚΑΛΕΣΕ Είσοδος(b)
                ΚΑΛΕΣΕ Είσοδος(c)
                ΚΑΛΕΣΕ Είσοδος(d)
                ΚΑΛΕΣΕ Είσοδος(e)
                sum1←sum1+α
                sum2←sum2+b
                sum3←sum3+c
                sum4←sum4+d
                sum5←sum5+e
        ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
                ΜΟ1←sum1/20
                ΜΟ2←sum2/20
                ΜΟ3←sum3/20
                ΜΟ4←sum4/20
                ΜΟ5←sum5/20
        ΓΡΑΨΕ ΜΟ1, ΜΟ2, ΜΟ3, ΜΟ4, ΜΟ5
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

```
ΔΙΑΔΙΚΑΣΙΑ Είσοδος(βαθμός)
ΜΕΤΑΒΛΗΤΕΣ
ΑΚΕΡΑΙΟΣ: βαθμός
ΑΡΧΗ
        ΕΜΦΑΝΙΣΕ 'Δώσε βαθμό'
        ΔΙΑΒΑΣΕ βαθμός
                ΟΣΟ βαθμός>20 Ή βαθμός <0 ΕΠΑΝΑΛΑΒΕ
                        ΕΜΦΑΝΙΣΕ 'Λάθος πληκτρολόγηση – Ξαναδώσε βαθμό'
                        ΔΙΑΒΑΣΕ βαθμός
                ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ
```

10.16 Τι είναι συνάρτηση;

Η συνάρτηση είναι ένας τύπος υποπρογράμματος που υπολογίζει και επιστρέφει μόνο μια τιμή με το όνομα της όπως οι μαθηματικές συναρτήσεις.

10.17 Ποια η δομή μιας συνάρτησης;

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

Οι συναρτήσεις εκτελούνται απλά με την εμφάνιση του ονόματος τους σε οποιαδήποτε έκφραση.

Το όνομα της συνάρτησης είναι οποιοδήποτε έγκυρο όνομα της ΓΛΩΣΣΑΣ.

Η λίστα παραμέτρων είναι μια λίστα μεταβλητών, των οποίων οι τιμές μεταβιβάζονται στη συνάρτηση κατά την κλήση.

Οι συναρτήσεις μπορούν να επιστρέφουν τιμές όλων των τύπων δεδομένων που υποστηρίζει η γλώσσα. Μια συνάρτηση λοιπόν μπορεί να είναι ΠΡΑΓΜΑΤΙΚΗ, ΑΚΕΡΑΙΑ, ΧΑΡΑΚΤΗΡΑΣ, ΛΟΓΙΚΗ.

Στις εντολές του σώματος της συνάρτησης πρέπει υποχρεωτικά να υπάρχει μία εντολή εκχώρησης τιμής στο όνομα της συνάρτησης.

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

10.18 Πως γίνεται η κλήση (χρήση) μιας συνάρτησης από ένα πρόγραμμα;

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

10.19 Ποια είναι τα βήματα για να κατασκευάσουμε μια συνάρτηση;

1) Αναγνωρίζουμε τα δεδομένα εισόδου, τα οποία θα δέχεται η συνάρτηση ως παραμέτρους.
2) Αναγνωρίζουμε το ζητούμενο και τον τύπο του, ώστε να δηλώσουμε τον τύπο της συνάρτησης.
3) Γράφουμε τον κώδικα της συνάρτησης και δηλώνουμε, στο τμήμα δηλώσεων της συνάρτησης, τον τύπο των μεταβλητών εισόδου, καθώς και τις τυχόν βοηθητικές μεταβλητές ή σταθερές της συνάρτησης.

10.20 Να δοθεί ένα παράδειγμα συνάρτησης.

Εύρεση μέγιστου μεταξύ τριών ακεραίων

```
ΣΥΝΑΡΤΗΣΗ max(α, β, γ): ΑΚΕΡΑΙΟΣ
ΜΕΤΑΒΛΗΤΕΣ
ΑΚΕΡΑΙΟΣ: α, β, γ
ΑΡΧΗ
  max←α
  ΑΝ β> max ΤΟΤΕ
        max←β
  ΤΕΛΟΣ_ΑΝ
  ΑΝ γ> max ΤΟΤΕ
        max←γ
  ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ
```

10.21 Δώστε ένα παράδειγμα πώς καλείται μια συνάρτηση από ένα πρόγραμμα.

Πρόγραμμα το οποίο υπολογίζει και εμφανίζει τη μεγαλύτερη ηλικία τριών ατόμων. Τα δεδομένα μας είναι τα έτη γέννησης των τριών αυτών ατόμων.

```
ΠΡΟΓΡΑΜΜΑ ηλικίες
ΜΕΤΑΒΛΗΤΕΣ
ΑΚΕΡΑΙΟΣ: α, έτος1, έτος2, έτος3, ηλικία1, ηλικία2, ηλικία3, Μ
ΑΡΧΗ
        ΓΡΑΨΕ 'Σε ποιο έτος βρισκόμαστε;'
        ΔΙΑΒΑΣΕ α
        ΓΡΑΨΕ 'Δώσε έτη γέννησης'
        ΔΙΑΒΑΣΕ έτος1, έτος2, έτος3
        ηλικία1←α- έτος1
        ηλικία2←α- έτος2
        ηλικία3←α- έτος3
        Μ←max(ηλικία1, ηλικία2, ηλικία3)
        ΓΡΑΨΕ Μ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
ΣΥΝΑΡΤΗΣΗ max(α, β, γ): ΑΚΕΡΑΙΟΣ
ΜΕΤΑΒΛΗΤΕΣ
ΑΚΕΡΑΙΟΣ: α, β, γ
ΑΡΧΗ
        max←α
        ΑΝ β> max ΤΟΤΕ
              max←β
        ΤΕΛΟΣ_ΑΝ
        ΑΝ γ> max ΤΟΤΕ
              max←γ
        ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ
```

10.22 Σε ποιο σημείο του προγράμματος τοποθετούνται τα υποπρογράμματα;

Οι συναρτήσεις και οι διαδικασίες τοποθετούνται μετά το τέλος του κυρίου προγράμματος και καλούνται στα σημεία που χρειάζονται.

10.23 Ποια τα χαρακτηριστικά και διαφορές των διαδικασιών και των συναρτήσεων;

Χαρακτηριστικά Συναρτήσεων	Χαρακτηριστικά Διαδικασιών

Οι συναρτήσεις υπολογίζουν μόνο μία τιμή, αριθμητική, χαρακτήρα ή λογική και μόνο αυτήν επιστρέφουν στο υποπρόγραμμα που την κάλεσε.	Οι διαδικασίες μπορούν να εκτελέσουν οποιαδήποτε λειτουργία, π.χ. να εισάγουν δεδομένα, να εκτελέσουν υπολογισμούς, να μεταβάλλουν τις τιμές των μεταβλητών και να τυπώσουν αποτελέσματα.
Οι συναρτήσεις μοιάζουν με τις συναρτήσεις των μαθηματικών και η χρήση τους είναι όμοια με τη χρήση των ενσωματωμένων συναρτήσεων που υποστηρίζει η γλώσσα προγραμματισμού.	Με τη χρήση παραμέτρων οι διαδικασίες μεταφέρουν τα αποτελέσματα τους στα άλλα υποπρογράμματα.
Οι συναρτήσεις εκτελούνται απλά με την εμφάνιση του ονόματος τους σε οποιαδήποτε έκφραση.	Για να ενεργοποιηθούν οι διαδικασίες χρησιμοποιείται η ειδική εντολή ΚΑΛΕΣΕ και το όνομα της διαδικασίας.

Τεστ αξιολόγησης επίδοσης

Συμπληρώστε με σωστό ή λάθος

1) Η κλήση των διαδικασιών γίνεται με απλή αναφορά του ονόματος τους.

2) Κάθε υποπρόγραμμα πρέπει να έχει μόνο μία είσοδο και μία έξοδο.

3) Οι συναρτήσεις μπορούν να υπολογίζουν και να επιστρέφουν μόνο μία τιμή.

Επιλέξτε μεταξύ των προτεινόμενων μία σωστή απάντηση.

4) Ποια η επικεφαλίδα της συνάρτησης **Εμβαδόν** που υπολογίζει το εμβαδόν ενός τριγώνου (Ε=1/2*β*υ).

Α) ΣΥΝΑΡΤΗΣΗ Εμβαδό(β,υ)
Β) ΣΥΝΑΡΤΗΣΗ Εμβαδό
Γ) ΣΥΝΑΡΤΗΣΗ Εμβαδό(β,υ):ΠΡΑΓΜΑΤΙΚΗ
Δ) ΠΡΑΓΜΑΤΙΚΗ ΣΥΝΑΡΤΗΣΗ Εμβαδό

5) Τι θα τυπώσουν οι επόμενες εντολές

```
…
Α<-10
Β<-5
ΚΑΛΕΣΕ διαδ(Α,Β)
ΓΡΑΨΕ Α,Β
…
ΔΙΑΔΙΚΑΣΙΑ διαδ(Γ,Δ)
…..
Α<-0
Β<-0
ΓΡΑΨΕ Α,Β
…..
```

Α) 10 5
0 0
Β) 10 5
10 5
Γ) 0 0
0 0

Δ) 0 0
10 5

6) Οι μεταβλητές που ισχύουν μόνο στο υποπρόγραμμα που δηλώθηκαν λέγονται .....

7) Η λίστα των ..... παραμέτρων εμφανίζεται στη δήλωση των υποπρογραμμάτων ενώ η λίστα των ..... παραμέτρων στην κλήση τους.