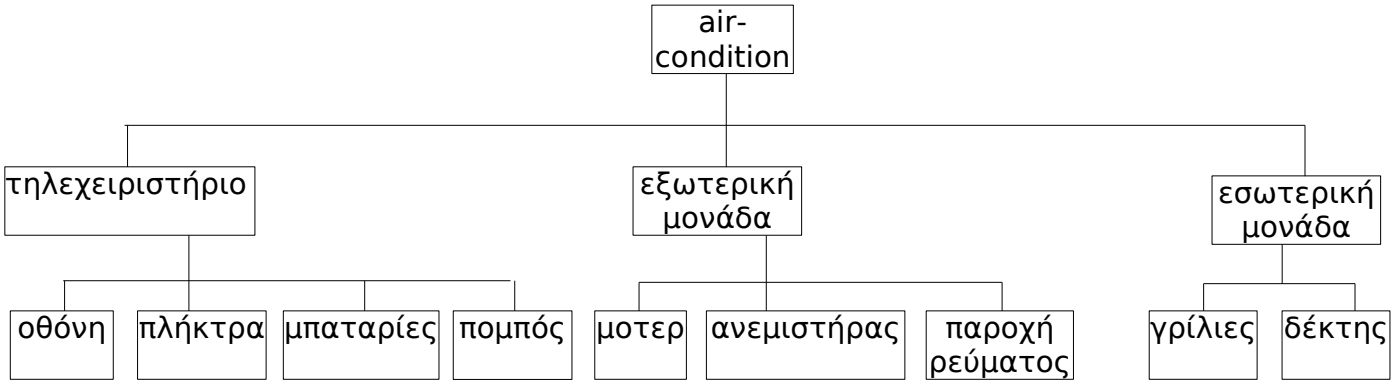
Ερωτήσεις Απαντήσεις Θεωρίας

Ανάπτυξη Εφαρμογών σε Προγραμματιστικό Περιβάλλον

Ευτύχης Χαιρετάκης
6976356215
eytyxis@gmail.com

Προβλήματα:

1. **Ποιός είναι ο ορισμός του προβλήματος**: Πρόβλημα είναι μία κατάσταση που πρέπει να αντιμετωπίσουμε, χρειάζεται λύση, που μπορεί να μην είναι ούτε γνωστή, ούτε προφανής.
2. **Ποιά είναι τα 3 στάδια αντιμετώπισης κάθε προβλήματος:** κατανόηση, ανάλυση, επίλυση
3. **Τι περιλαμβάνει το στάδιο της κατανόησης:** Τον εντοπισμό των δεδομένων και των ζητουμένων (αλλιώς, καθορισμός απαιτήσεων).
4. **Από ποιούς βασικούς παράγοντες επηρεάζεται η κατανόηση:** από τη διατύπωση και την ερμηνεία.
5. **Πότε θεωρείται μια διατύπωση καλή:** όταν είναι σαφής, σε σωστή γλώσσα και με σωστή ορολογία, παρέχει όλα τα απαραίτητα στοιχεία, δίνει ορθά στοιχεία.
6. **Τι προϋποθέτει μια καλή ερμηνεία:** ο λύτης χρειάζεται να έχει γνώσεις, εμπειρία, και να δίνει την απαραίτητη προσοχή στη διατύπωση.
7. **Τι περιλαμβάνει το στάδιο της ανάλυσης:** Τη διάσπαση του αρχικού προβλήματος σε όσο γίνεται πιο στοιχειώδη υποπροβλήματα (και στα υποπροβλήματά τους)
8. **Τι αποκαλύπτει το στάδιο της ανάλυσης:** τη δομή του προβλήματος
9. **Τι ονομάζουμε δομή του προβλήματος:** τα επιμέρους υποπροβλήματά του και τις σχέσεις με τις οποίες αυτά συνδέονται μεταξύ τους.
10. **Τι περιλαμβάνει το στάδιο της επίλυσης:** το συνδυασμό των λύσεων των επιμέρους υποπροβλημάτων, ώστε να οδηγηθούμε στη λύση του αρχικού προβλήματος
11. **Τι ονομάζουμε δεδομένο:** Όποιο στοιχείο μπορεί να γίνει αντιληπτό από έναν τουλάχιστον παρατηρητή με μια από τις αισθήσεις του.
12. **Σχόλιο για τα δεδομένα (data):** είναι η αφαιρετική αναπαράσταση της πραγματικότητας και συνεπώς μία απλοποιημένη όψη της.
13. **Τι ονομάζουμε πληροφορία:** Όποιο γνωσιακό στοιχείο μπορούμε να αντλήσουμε από τα δεδομένα, κατόπιν επεξεργασίας.
14. **Τι ονομάζουμε επεξεργασία δεδομένων:** Ο μηχανισμός εξαγωγής πληροφοριών από τα δεδομένα *(μπορεί να περιλαμβάνει αριθμητικές πράξεις, συγκρίσεις, λογικές συσχετίσεις μεταξύ των δεδομένων)*.
15. **Αναφέρετε δυο μηχανισμούς επεξεργασίας δεδομένων:** Ο υπολογιστής και ο ανθρώπινος εγκέφαλος
16. **Τι ονομάζουμε χώρο του προβλήματος:** Τα προβλήματα μπορεί να είναι κοινωνικά, εθνικά, οικονομικά, ψυχολογικά, οικογενειακά, πολιτικά, επιστημονικά κ.ά.
17. **Τι ονομάζουμε καθορισμό απαιτήσεων:** Το να προσδιοριστούν με ακρίβεια τα δεδομένα και η σημασία τους, ποιό είναι το μέγεθος του προβλήματος, και να καταγραφούν με λεπτομέρεια ποιά είναι τα ζητούμενα και σε ποιά μορφή πρέπει να παρουσιαστούν.
18. **Τι είναι η διαγραμματική αναπαράσταση:** ένας τρόπος απεικόνισης της δομής ενός προβλήματος (ή ενός αντικειμένου, ή ενός οργανισμού).
19. **Ένα παράδειγμα:**



20. **Πώς αλλιώς μπορεί να περιγραφεί η δομή ενός προβλήματος:** φραστικά

Διαίρει & Βασίλευε:

21. **Συγγενή προβλήματα:** προβλήματα που μπορούν να αναλυθούν με παρόμοιο τρόπο και να αντιμετωπισθούν με παρόμοιες μεθόδους και τεχνικές

22. **Μια μέθοδος ανάλυσης και επίλυσης προβλημάτων είναι χρήσιμη γιατί:**
    - παρέχει ένα γενικό πρότυπο κατάλληλο για την επίλυση προβλημάτων ευρείας κλίμακας,
    - μπορει να αναπαρασταθει με κοινές δομές δεδομένων και ελέγχου (που υποστηρίζονται από όλες τις σύγχρονες γλώσσες προγραμματισμού),
    - παρέχει τη δυνατότητα καταγραφής των χρονικών και "χωρικών" απαιτήσεων της μεθόδου επίλυσης, έτσι ώστε να μπορεί να γίνει επακριβής εκτίμηση των αποτελεσμάτων.
23. **Η ανάλυση ενός προβλήματος σε ένα σύγχρονο υπολογιστικό περιβάλλον περιλαμβάνει:**
    1. την καταγραφή της υπάρχουσας πληροφορίας για το πρόβλημα,
    2. την αναγνώριση των ιδιαιτεροτήτων του προβλήματος,
    3. την αποτύπωση των συνθηκών και προϋποθέσεων υλοποίησής του
    4. την πρόταση επίλυσης με χρήση κάποιας μεθόδου, και
    5. την τελική επίλυση με χρήση υπολογιστικών συστημάτων.
24. **Διαίρει & Βασίλευε (divide & conquer):** Μέθοδος επίλυσης προβλημάτων που βασίζεται στην υποδιαίρεση του προβλήματος σε υποπροβλήματα (και την υποδιαίρεσή τους σε ακόμα μικρότερα) ίδια με το αρχικό, αλλά μικρότερα σε μέγεθος (αναφέρονται σε λιγότερα/ευκολότερα δεδομένα). Το αρχικό πρόβλημα επιλύεται από την λύση ενός υποπροβλήματος ή το συνδυασμό των λύσεων περισσότερων υποπροβλημάτων. Η προσέγγιση αυτή ονομάζεται «**από πάνω προς τα κάτω**» (top-down).
25. **Πόσο αποδοτική είναι η μέθοδος Δ&Β:** Συνήθως για προβλήματα με Ν δεδομένα, ένας αλγόριθμος Δ&Β απαιτεί στη χειρότερη περίπτωση $\log_2(n)$ επαναλήψεις, που σημαίνει πολύ καλή αποδοτικότητα.
26. **Ποιά είναι τα βήματα γενικά της μεθόδου:**
    1. Δίνεται για επίλυση ένα στιγμιότυπο ενός προβλήματος.
    2. Το στιγμιότυπο του προβλήματος υποδιαιρείται σε υπο-στιγμιότυπα του ίδιου προβλήματος.
    3. Δίνεται ανεξάρτητη λύση σε κάθε ένα υπο-στιγμιότυπο.
    4. Συνδυάζονται όλες ή μερικές λύσεις που βρέθηκαν για τα υπο-στιγμιότυπα, έτσι ώστε να δοθεί η συνολική λύση του προβλήματος.

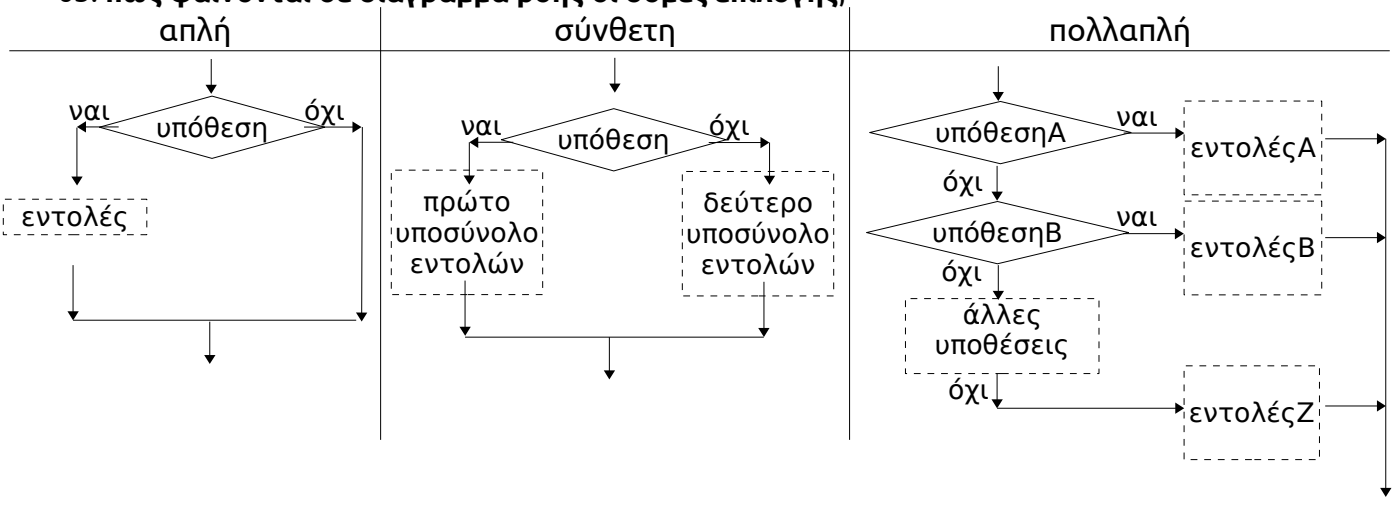
Αλγόριθμοι:

27. **ποιός είναι ο ορισμός του αλγορίθμου;** είναι μία πεπερασμένη σειρά ενεργειών, αυστηρά καθορισμένων και εκτελέσιμων σε πεπερασμένο χρόνο, με σκοπό την επίλυση ενός προβλήματος
28. **Ποιές είναι οι 4 σκοπιές από τις οποίες η Πληροφορική μελετά τους αλγορίθμους:** υλικού, γλωσσών προγραμματισμού, θεωρητική, αναλυτική
29. **Τι σημαίνει η μελέτη από τη σκοπιά του υλικού:** Η ταχύτητα εκτέλεσης ενός αλγορίθμου επηρεάζεται από τις διάφορες τεχνολογίες υλικού, δηλαδή από τον τρόπο που είναι δομημένα σε μία ενιαία αρχιτεκτονική τα διάφορα συστατικά του υπολογιστή (επεξεργαστές, μνήμες, ταχύτητες).
30. **Τι σημαίνει η μελέτη από τη σκοπιά των γλωσσών προγραμματισμού:** Ανάλογα με τη γλώσσα προγραμματισμού αλλάζει η δομή του αλγορίθμου, ο αριθμός των εντολών, η ταχύτητα μετάφρασης, οι διαθέσιμες στον προγραμματιστή τεχνικές και εντολές.
31. **Τι σημαίνει η μελέτη από τη θεωρητική σκοπιά:** Εξετάζεται αν πράγματι υπάρχει ή όχι κάποιος αποδοτικός αλγόριθμος για την επίλυση ενός προβλήματος.
32. **Τι σημαίνει η μελέτη από την αναλυτική σκοπιά:** Μελετώνται οι υπολογιστικοί πόροι 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.
33. **ποιά είναι τα πέντε χαρακτηριστικά/κριτήρια κάθε αλγορίθμου;** είσοδος, έξοδος, περατότητα, καθοριστικότητα, αποτελεσματικότητα
34. **τι σημαίνει το καθένα από τα πέντε χαρακτηριστικά**;
    - **είσοδος** είναι τα δεδομένα που χρησιμοποιεί ο αλγόριθμος (που μπορεί να δίνονται από τον χρήστη, από το πρόβλημα (εκφώνηση), να παράγονται μέσα στον αλγόριθμο, ή να προέρχονται από έναν άλλο αλγόριθμο)
    - **έξοδος** είναι τα αποτελέσματα τα οποία μπορεί να υπολογίζει ή να εμφανίζει ο αλγόριθμος
    - **περατότητα** είναι η ιδιότητα του αλγορίθμου να ολοκληρώνεται σε συγκεκριμένο χρονικό διάστημα, να μην εκτελεί επ' άπειρο τις εντολές του
    - **καθοριστικότητα** έχει όταν κάθε εντολή είναι σαφής ως προς το τι κάνει και πως εκτελείται
    - **αποτελεσματικότητα** έχει όταν κάθε εντολή είναι απλή και εκτελέσιμη, έχοντας προβλέψει όλα τα ενδεχόμενα
35. **ξέρω ένα παράδειγμα αλγορίθμου που να τού λείπει κάθε ένα από αυτά τα χαρακτηριστικά**;

- να λείπει η είσοδος είναι απίθανο,
- να λείπει η έξοδος μπορεί αν δεν υπάρχει καμία εντολή γράψε, ούτε εκχώρηση τιμής, ούτε παραπομπή σε υποπρόγραμμα
- να λείπει η περατότητα μπορεί αν σε μία ΓΙΑ έχει βήμα 0, ή σε μία ΟΣΟ ή ΜΕΧΡΙΣ_ΟΤΟΥ η ελεγχόμενη συνθήκη είναι πάντα ΑΛΗΘΗΣ ή ΨΕΥΔΗΣ αντίστοιχα
- να λείπει η καθοριστικότητα μπορεί αν υπάρχει εντολή που δε γνωρίζουμε ποιό θα είναι το αποτέλεσμά της πχ «αύξησε x», «α ← 5/0»
- να λείπει η αποτελεσματικότητα μπορεί όταν δεν γίνονται οι απαραίτητοι έλεγχοι εγκυρότητας, ή δεν ελέγχονται όλες οι συνθήκες

36. **πώς ονομάζεται ο «αλγόριθμος» που φτιάξαμε, όταν δεν υπάρχει περατότητα**; υπολογιστική διαδικασία
37. **ποιές είναι οι τρεις αλγοριθμικές δομές**; δομή ακολουθίας, δομή επιλογής, δομή επανάληψης (τις οποίες συναντάμε και εμφωλευμένες τη μία μέσα στην άλλη)
38. **ποιές είναι οι άλλες αλγοριθμικές συνιστώσες / τι άλλο συναντά κανείς σε έναν αλγόριθμο / ποιά είναι τα συστατικά των αλγορίθμων**; μεταβλητές, σταθερές, τελεστές, συναρτήσεις, παραστάσεις (εκφράσεις), εντολές, σχόλια, δηλώσεις αρχής-τέλους, δηλώσεις δεδομένων-αποτελεσμάτων
39. **από τί αποτελείται μια παράσταση / έκφραση**; από τελεστές, τελεστέους, μεταβλητές, σταθερές, συναρτήσεις
40. **τι είδη παραστάσεων υπάρχουν**; αριθμητικές παραστάσεις (όπου κάνεις πράξεις και καταλήγεις σε μία αριθμητική τιμή πχ «2+8») και λογικές παραστάσεις (όπου κάνεις πράξεις και καταλήγεις σε μία λογική τιμή πχ «14 > 21»). Αυτές οι παραστάσεις μπορεί να παρουσιάζονται και σύνθετες, δηλαδή να έχουν πολλούς τελεστές μαζί, και να αποτελούνται από επιμέρους απλούστερες παραστάσεις (πχ «(21+α)/(α^2+β^2+1)» , «x>0 και ψ>0»).
41. **ποιοί είναι οι τέσσερεις τρόποι αναπαράστασης των αλγορίθμων**; ελεύθερο κείμενο, φυσική γλώσσα σε βήματα, διαγραμματικές τεχνικές (διάγραμμα ροής), κωδικοποίηση (σε ψευδογλώσσα, ή γλώσσα προγραμματισμού)
42. **ξέρω να ξεχωρίζω αυτούς τους τρόπους αναπαράστασης με μια ματιά**; το ελεύθερο κείμενο ξεχωρίζει λόγω έλλειψης κανόνων, η φυσική γλώσσα έχει αριθμημένα τα βήματα, τα διαγράμματα ροής έχουν συγκεκριμένα σχήματα, η κωδικοποίηση έχει συγκεκριμένες εντολές, λεξιλόγιο και κανόνες
43. **ποιά χαρακτηριστικά των αλγορίθμων κινδυνεύουν να χαθούν στο ελεύθερο κείμενο**; η καθοριστικότητα και η αποτελεσματικότητα
44. **ποιά χαρακτηριστικά των αλγορίθμων κινδυνεύουν να χαθούν στη φυσική γλώσσα**; η καθοριστικότητα
45. **ποιά είναι η γενική μορφή σύνταξης ενός αλγορίθμου σε κωδικοποιημένη μορφή**;
    αλγόριθμος *όνομα*
    τμήμα-εντολών
    τέλος *όνομα*
46. **πως γράφουμε ένα σχόλιο:** ξεκινώντας μια γραμμή με το σύμβολο «!» μπορούμε να γράψουμε ένα σχόλιο που να περιγράφει με λόγια τί ακριβώς κάνουμε
47. **πότε και πού δηλώνω «Δεδομένα // ... //» και «Αποτελέσματα // ... //»**: μόνο σε αλγορίθμους, όχι σε προγράμματα, αντίστοιχα στην πρώτη εντολή μπορεί να δηλώσεις δεδομένα, αν η εκφώνηση λέει ότι δίνονται και δε ζητάει να διαβάζονται, ενώ στην τελευταία εντολή μπορεί να δηλώσεις αποτελέσματα, αν η εκφώνηση δε ζητάει νωρίτερα να εμφανίζονται
48. **ποιά σχήματα χρησιμοποιούνται στα διαγράμματα ροής**; έλλειψη (που δηλώνει την αρχή ή το τέλος), ορθογώνιο παραλληλόγραμμο (για τις εντολές εκχώρησης τιμής), πλάγιο παραλληλόγραμμο (για τις εντολές εισόδου-εξόδου), ρόμβος (για τον έλεγχο συνθηκών)
49. **πως συντάσσεται μια εντολή εισόδου**;
    διάβασε *μεταβλητές-χωρισμένες-με-κόμμα*
50. **πως λειτουργεί μια εντολή εισόδου**; διακόπτεται η εκτέλεση του προγράμματος, και ο υπολογιστής περιμένει από το χρήστη να πληκτρολογήσει μία τιμή, για κάθε μεταβλητή που τοποθετήσαμε δίπλα στην εντολή διάβασε, να πατήσει enter, και μετά συνεχίζει την εκτέλεση του προγράμματος
51. **πως συντάσσεται μια εντολή εξόδου**;
    γράψε *σταθερές-και-μεταβλητες-χωρισμένες-με-κόμμα*
52. **πως λειτουργεί μια εντολή εξόδου**; εμφανίζει στην οθόνη μία-μία τις τιμές των σταθερών και των μεταβλητών που τοποθετήσαμε δίπλα στην εντολή γράψε, με τη σειρά που τις τοποθετήσαμε
53. **πως συντάσσεται μια εντολή ανάθεσης (εκχώρησης) τιμής**;
    μεταβλητή ← *παράσταση*

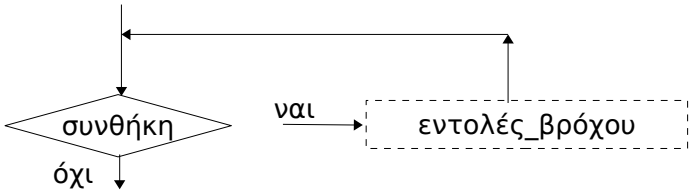
54. **πως λειτουργεί μια εντολή ανάθεσης (εκχώρησης) τιμής**; εκτελούνται πρώτα οι πράξεις που υπάρχουν στην *παράσταση* (αν υπάρχουν), σκολουθώντας την ιεραρχία των πράξεων, από αυτές θα προκύψει μία τιμή, και η τιμή αυτή αποθηκεύεται στη *μεταβλητή*
55. **πως συντάσσεται η δομή απλής επιλογής**;
    αν *συνθήκη* **τότε**
    εντολές
    τέλος_αν
56. **πως λειτουργεί η δομή απλής επιλογής**; ελέγχεται η *συνθήκη*. Αν είναι ΑΛΗΘΗΣ, εκτελούνται οι *εντολές*. Αν είναι ΨΕΥΔΗΣ δεν εκτελούνται
57. **πως συντάσσεται η δομή σύνθετης επιλογής**;
    αν *συνθήκη* **τότε**
    πρώτο-υποσύνολο-εντολών
    αλλιώς
    δεύτερο-υποσύνολο-εντολών
    τέλος_αν
58. **πως λειτουργεί η δομή σύνθετης επιλογής**; ελέγχεται η *συνθήκη*. Αν είναι ΑΛΗΘΗΣ εκτελείται μόνο το *πρώτο υποσύνολο εντολών*. Αν είναι ΨΕΥΔΗΣ εκτελείται μόνο το *δεύτερο υποσύνολο εντολών*
59. **πως συντάσσεται η δομή πολλαπλής επιλογής**;
    αν (*υπόθεσηΑ*) **τότε**
    εντολέςΑ
    αλλιώς_αν (*υπόθεσηΒ*) **τότε**
    εντολέςΒ
    ...
    αλλιώς
    εντολέςΖ
    τέλος_αν
60. **πως λειτουργεί η δομή πολλαπλής επιλογής**; ελέγχεται πρώτα η *υπόθεσηΑ*. Αν είναι ΑΛΗΘΗΣ εκτελούνται μόνο οι *εντολέςΑ*. Μόνο αν είναι ΨΕΥΔΗΣ ελέγχεται η *υπόθεσηΒ*. Αν είναι η *υπόθεσηΒ* ΑΛΗΘΗΣ εκτελούνται μόνο οι *εντολέςΒ*. ... κοκ... Αν όλες οι ελεγχόμενες υποθέσεις διαδοχικά είναι ΨΕΥΔΗΣ, εκτελούνται οι *εντολέςΖ*
61. **πώς συντάσσεται η δομή πολλαπλής επιλογής με ΕΠΙΛΕΞΕ**:
    ΕΠΙΛΕΞΕ *(παράσταση)*
    ΠΕΡΙΠΤΩΣΗ *(λίστα ή διάστημα τιμών τιμών της παράστασης)*
    εντολέςΑ
    ΠΕΡΙΠΤΩΣΗ *(λίστα ή διάστημα τιμών τιμών της παράστασης)*
    εντολέςΒ
    ...
    ΠΕΡΙΠΤΩΣΗ ΑΛΛΙΩΣ
    εντολέςΓ
    ΤΕΛΟΣ_ΕΠΙΛΟΓΩΝ
62. **πότε να προτιμήσουμε την ΕΠΙΛΕΞΕ αντί της ΑΝ**: Σε προβλήματα όπου θέλουμε να εξετάσουμε περιπτώσεις με πολλές διακριτές τιμές, ή διάφορα διαδοχικά διαστήματα, θεωρείται πιο συμπαγής τρόπος κωδικοποίησης με τη δομή ΕΠΙΛΕΞΕ αντί της ΑΝ.
63. **πώς φαίνονται σε διάγραμμα ροής οι δομές επιλογής**;



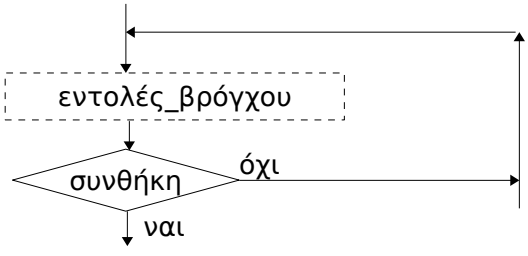
64. **πότε πρέπει να προσέχω τη σειρά με την οποία διατυπώνω τις υποθέσεις σε μία δομή πολλαπλής επιλογής**; όταν οι ελεγχόμενες περιπτώσεις δεν είναι αλληλοαποκλειόμενες (πχ η μία λέει «x=ψ» και η άλλη λέει «x=0 και ψ=0»)
65. **τι είναι η εμφώλευση**; όταν μέσα στις εντολές που εκτελούνται σε μία δομή, περιλαμβάνεται μία άλλη δομή, εκείνη που είναι εσωτερικά λέμε ότι είναι εμφωλευμένη
66. **πώς μετατρέπεις την πολλαπλή σε σύνθετες**; με διαδοχικές εμφωλεύσεις σύνθετων, όπως στο ακόλουθο παράδειγμα:
67. **πώς μετατρέπεις την πολλαπλή σε απλές**; με πολλές διαδοχικές απλές, όπως στο ακόλουθο παράδειγμα:
68. **πώς μετατρέπεις τη σύνθετη σε απλές**; με δύο απλές

πολλαπλή	σύνθετες	πολλές διαδοχικές απλές
αν συνθ1 **τότε** εντολές1 **αλλιώς_αν** συνθ2 **τότε** εντολές2 **αλλιώς_αν** συνθ3 **τότε** εντολές3 **αλλιώς** εντολές4 **τέλος_αν**	**αν** συνθ1 **τότε** εντολές1 **αλλιώς** **αν** συνθ2 **τότε** εντολές2 **αλλιώς** **αν** συνθ3 **τότε** εντολές3 **αλλιώς** εντολές4 **τέλος_αν** **τέλος_αν** **τέλος_αν**	**αν** συνθ1 **τότε** εντολές1 **τέλος_αν** **αν** συνθ2 **και οχι** συνθ1 **τότε** εντολές2 **τέλος_αν** **αν** συνθ3 **και οχι** συνθ1 **και οχι** συνθ2 **τότε** εντολές3 **τέλος_αν** **αν όχι** συνθ1 **και οχι** συνθ2 **και οχι** συνθ3 **τότε** εντολές4 **τέλος_αν**

69. **τι είναι βρόχος**; η δομή επανάληψης αλλιώς ονομάζεται βρόχος
70. **τί είναι ατέρμονας βρόχος**; η δομή επανάληψης που δεν τερματίζεται, δεν έχει περατότητα
71. **πως συντάσσεται γενικά η ΟΣΟ**;
    όσο (*συνθήκη*) **επανάλαβε**
    εντολές_βρόχου
    τέλος_επανάληψης
72. **πως λειτουργεί η ΟΣΟ**; πρώτα ελέγχεται η *συνθήκη*, αν είναι ΑΛΗΘΗΣ εκτελούνται οι *εντολές βρόχου* και μετά ξαναελέγχεται η *συνθήκη*. Αυτό επαναλαμβάνεται ώσπου η *συνθήκη* γίνει ΨΕΥΔΗΣ.
73. **πώς φαίνεται η ΟΣΟ σε διάγραμμα ροής**;



74. **πως συντάσσεται γενικά η ΜΕΧΡΙΣ_ΟΤΟΥ**;
    αρχή_επανάληψης
    εντολές_βρόχου
    μέχρις_ότου (*συνθήκη ελέγχου*)
75. **πώς λειτουργεί η ΜΕΧΡΙΣ_ΟΤΟΥ**; εκτελούνται οι *εντολές βρόχου*, και μετά ελέγχεται η *συνθήκη*. Αν είναι ΨΕΥΔΗΣ ξαναεκτελούνται οι *εντολές βρόχου*. Αυτό επαναλαμβάνεται ώσπου η *συνθήκη* γίνει ΑΛΗΘΗΣ.
    πώς φαίνεται η ΜΕΧΡΙΣ_ΟΤΟΥ σε διάγραμμα ροής;



76. **πως συντάσσεται γενικά η ΓΙΑ**;
    για μ **από** τ1 **μέχρι** τ2 **με βήμα** τ3
    εντολές_βρόχου
    τέλος_επανάληψης

77. **πως λειτουργεί η ΓΙΑ**;
είναι το ισοδύναμο της:

```
μ ← τ1
όσο μ <= τ2 επανάλαβε
  εντολές-βρόχου
  μ ← μ+τ3
τέλος_επανάληψης
```

η ανίσωση θα είναι μ>=τ2 όταν τ3<0

78. **πώς φαίνεται η ΓΙΑ σε διάγραμμα ροής**; τη μετατρέπεις σε ΟΣΟ και κάνεις το διάγραμμα
79. **πώς μπορώ να υπολογίσω τον αριθμό των επαναλήψεων που εκτελεί μία ΓΙΑ**;
    - με τ3 > 0 και τ1 > τ2, εκτελούνται 0 επαναλήψεις
    - με τ3 < 0 και τ1 < τ2, εκτελούνται 0 επαναλήψεις
    - με τ3 = 0, εκτελούνται άπειρες επαναλήψεις
    - διαφορετικά το πλήθος Ν των επαναλήψεων δίνεται από τη σχέση N = A_M(A_T(τ2-τ1) / τ3) + 1
80. **ποιές είναι οι διαφορές μεταξύ των δομών επανάληψης ΟΣΟ, ΜΕΧΡΙΣ_ΟΤΟΥ**;
    - στην **όσο** πρώτα ελέγχεται η *συνθήκη* και μετά ακολουθούν οι *εντολές βρόχου*, ενώ στη **μέχρις_ότου** πρώτα εκτελούνται οι *εντολές βρόχου* και μετά ελέγχεται η *συνθήκη*
    - άρα στην **όσο** μπορεί να μην εκτελεστεί καμία επανάληψη, ενώ στη **μέχρις_οτου** θα εκτελεστεί τουλάχιστον μία
    - στην **όσο** εκτελούνται επαναλήψεις όταν η ελεγχόμενη *συνθήκη* είναι ΑΛΗΘΗΣ, ενώ στη **μέχρις_ότου** όταν είναι ΨΕΥΔΗΣ
81. **τι πρέπει να υπάρχει πάντα μέσα στις εντολές βρόχου της ΟΣΟ και της ΜΕΧΡΙΣ_ΟΤΟΥ**; μια εντολή, που θα αλλάζει κάποια από τις μεταβλητές που υπάρχουν στην ελεγχόμενη συνθήκη, αλλιώς δεν υπάρχει περατότητα
82. **πότε ταιριάζει καλύτερα να χρησιμοποιηθεί η καθεμία από τις τρεις δομές επανάληψης**; η **για** μόνο σε προβλήματα γνωστού αριθμού επαναλήψεων, η **μέχρις_ότου** σε προβλήματα αγνώστου αριθμού επαναλήψεων όπου όμως τουλάχιστον μία επανάληψη απαιτείται, η **όσο** σε προβλήματα αγνώστου αριθμού επαναλήψεων όπου όμως μπορεί να μην εκτελεστεί καμία επανάληψη
83. **Οι ΟΣΟ και ΜΕΧΡΙΣ_ΟΤΟΥ μπορούν να χρησιμοποιηθούν και σε προβλήματα γνωστού αριθμού επαναλήψεων**; πάντα
84. **γίνονται μετατροπές από τη μία μορφή δομής επανάληψης στην άλλη**; η **για** πάντα μετατρέπεται στις άλλες δύο μορφές, το αντίστροφο όχι πάντα. Η **όσο** και η **μέχρις_ότου** μεταξύ τους πάντα μπορούν να μετατραπούν
85. **τί να προσέξω όταν μετατρέψω το ΟΣΟ σε ΜΕΧΡΙΣ_ΟΤΟΥ**; σίγουρα πρέπει να αντιστραφεί η *συνθήκη*. Μετά, η **όσο** μπορεί να μην κάνει καμία επανάληψη, ενώ η **μέχρις_ότου** κάνει εξ ορισμού μία, γι'αυτό μπορεί να χρειάζεται μία έξτρα δομή επιλογής, όπως στο παράδειγμα:

```
ΟΣΟ συνθήκη ΕΠΑΝΑΛΑΒΕ
  εντολες_βροχου
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

```
ΑΝ συνθηκη ΤΟΤΕ
  ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
    εντολες_βροχου
  ΜΕΧΡΙΣ_ΟΤΟΥ ΟΧΙ συνθηκη
ΤΕΛΟΣ_ΑΝ
```

86. **τί να προσέξω όταν μετατρέψω το ΜΕΧΡΙΣ_ΟΤΟΥ σε ΟΣΟ**; σίγουρα πρέπει να αντιστραφεί η *συνθήκη*. Μετά, επειδή η **μέχρις_ότου** κάνει μια επανάληψη ενώ η **όσο** μπορεί να μην κάνει καμία, μία απλή λύση για να είναι σίγουρο ότι οι εντολές του βρόχου θα εκτελεστούν τουλάχιστον μία φορά, είναι να τις γράψουμε και έξω από το βρόχο, όπως στο παράδειγμα:

```
ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
  εντολες_βρόχου
ΜΕΧΡΙΣ_ΟΤΟΥ συνθήκη
```

```
εντολές_βρόχου
ΟΣΟ ΟΧΙ συνθήκη ΕΠΑΝΑΛΑΒΕ
  εντολές_βρόχου
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

87. **τί να προσέξω όταν μετατρέψω το ΓΙΑ σε ΌΣΟ (ή ΜΕΧΡΙΣ_ΟΤΟΥ)**; το πρόσημο του βήματος(τ3) είναι αυτό που καθορίζει τη φορά της ανίσωσης

Γενικός κανόνας με τ3 > 0

```
ΓΙΑ x ΑΠΟ τ1 ΜΕΧΡΙ τ2 ΜΕ ΒΗΜΑ τ3
  εντολές
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

```
x ← τ1
ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
  εντολές
  x ← x + τ3
```

Γενικός κανόνας με τ3 < 0

```
ΓΙΑ x ΑΠΟ τ1 ΜΕΧΡΙ τ2 ΜΕ ΒΗΜΑ τ3
  εντολές
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

```
x ← τ1
ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
  εντολές
  x ← x + τ3
```

ΜΕΧΡΙΣ_ΟΤΟΥ x > τ2 x ← τ1 **ΟΣΟ** x <= τ2 **ΕΠΑΝΑΛΑΒΕ** εντολές x ← x + τ3 **ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ**	**ΜΕΧΡΙΣ_ΟΤΟΥ** x < τ2 x ← τ1 **ΟΣΟ** x >= τ2 **ΕΠΑΝΑΛΑΒΕ** εντολές x ← x + τ3 **ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ**

88. **τί να προσέξω όταν μετατρέψω το ΌΣΟ (ή το ΜΕΧΡΙΣ_ΟΤΟΥ) σε ΓΙΑ**; στη **για** η τελευταία εντολή που εκτελείται σε κάθε επανάληψη, είναι η εντολή που αλλάζει τη μεταβλητή ανάλογα με το βήμα. Στην **όσο** ή τη **μέχρις_ότου**, αυτή η εντολή μπορεί να είναι τοποθετημένη οπουδήποτε μέσα στο βρόχο
89. **τι πρέπει να προσέχω κατά την εμφώλευση δομών επανάληψης**;
    - ο εσωτερικός βρόχος να βρίσκεται ολόκληρος μέσα στον εξωτερικό,
    - ο βρόχος που ξεκινά τελευταίος (ο εμφωλευμένος) να ολοκληρώνεται πρώτος,
    - η είσοδος σε κάθε βρόχο να γίνεται από την αρχή του, και
    - να μη χρησιμοποιούμε την ίδια μεταβλητή ως μετρητή δύο (ή περισσότερων) βρόχων.
90. **τι ονομάζεται «τιμή φρουρός»**; μια αυθαίρετη/παράλογη τιμή που διαλέγει ο προγραμματιστής ώστε μόλις παρουσιαστεί στον αλγόριθμο -συνήθως μόλις πληκτρολογηθεί από το χρήστη- να τερματίζονται κάποιες επαναλήψεις
91. **τι είναι η (αριστερή και η δεξιά) ολίσθηση**; αριστερή ολίσθηση είναι η πράξη του διπλασιασμού (* 2), ενώ δεξιά ολίσθηση είναι η πράξη του υποδιπλασιασμού (div 2)
92. **τι είναι ο πολλαπλασιασμός αλά ρωσικά**; μία μέθοδος που ανάγει κάθε πολλαπλασιασμό σε πρόσθεση και ολίσθηση, που είναι πολύ εύκολες για τον υπολογιστή
93. **τί είναι οι τελεστές**; τα σύμβολα που χρησιμοποιούμε για να δείξουμε ποιά είναι η τελούμενη πράξη
94. **τί είναι οι τελεστέοι**; τα σύμβολα στα οποία εκτελείται μία πράξη (πχ η πράξη της πρόσθεσης έχει δύο προσθετέους). Συνήθως είναι μεταβλητές, σταθερές, ή συναρτήσεις
95. **σε ποιές τρεις κατηγορίες χωριζονται οι τελεστές**; αριθμητικοί, συγκριτικοί, λογικοί
96. **ποιά είναι η προτεραιότητα/ ιεραρχία μεταξύ των τριών κατηγοριών τελεστών**; η προαναφερόμενη. Πρώτα εκτελούνται αριθμητικές πράξεις, μετά συγκριτικές, και τελευταίες οι λογικές. Όταν υπάρχουν παρενθέσεις, προηγούνται οι παρενθέσεις.
97. **ποιοί είναι οι τελεστές της κάθε κατηγορίας**;

αριθμητικοί	(1) ^	(2) * / div mod	(3) + -
συγκριτικοί	(4) >	< =	<= >= <>
λογικοί	(5) όχι (άρνηση)	(6) και (σύζευξη)	(7) ή (διάζευξη)

98. **ποιά είναι η προτεραιότητα μεταξύ των τελεστών κάθε κατηγορίας**; όπως δείχνουν οι αριθμοί στο πινακάκι. Υψηλότερη προτεραιότητα έχει το (1), χαμηλότερη το (7). Όταν έχεις μόνο τελεστές ίδιας προτεραιότητας, οι πράξεις εκτελούνται από αριστερά προς τα δεξιά. Όταν υπάρχουν παρενθέσεις, παίρνουν πάντα προτεραιότητα
99. **με ποιές πράξεις βρίσκω συγκεκριμένα ψηφία ενός ακέραιου αριθμού Χ**;

μονάδες	δεκάδες	εκατοντάδες	χιλιάδες	κ.ο.κ
x div **1** mod 10	x div **10** mod 10	x div **100** mod 10	x div **1000** mod 10	

100. **με ποιά πράξη βρίσκω τα τελευταία από δεξιά ψηφία ενός ακεραίου Χ**;

1 τελευταίο	2 τελευταία	3 τελευταία	4 τελευταία	κ.ο.κ
x mod **10**	x mod **100**	x mod **1000**	x mod **1000**	

101. **με ποιά πράξη βρίσκω ψηφία ενός ακεραίου Χ, εξαιρώντας τα τελευταία από δεξιά**;

χωρίς 1 τελευταίο	χωρίς 2 τελευταία	χωρίς 3 τελευταία	χωρίς 4 τελευταία	κ.ο.κ
x div **10**	x div **100**	x div **1000**	x div **1000**	

102. **με ποιά συνθήκη ελέγχω αν ο ακέραιος Χ είναι πολλαπλάσιο του ακεραίου Υ**; X mod Y = 0
103. **ποιό είναι το αποτέλεσμα του τελεστή ΟΧΙ**; μετατρέπει μια ΑΛΗΘΗ σχέση σε ΨΕΥΔΗ, και μία ΨΕΥΔΗ σε ΑΛΗΘΗ
104. **ποιά είναι τα αποτελέσματα του ΚΑΙ και του Η**; όπως φαίνονται στο πινακάκι:

υπόθεση1	υπόθεση2	**σύζευξη**υπόθεση1 **ΚΑΙ** υπόθεση2	**Διάζευξη** υπόθεση1 **Η** υπόθεση2
αληθής	*αληθής*	*αληθής*	*αληθής*
αληθής	*ψευδής*	*ψευδής*	*αληθής*
ψευδής	*αληθής*	*ψευδής*	*αληθής*
ψευδής	*ψευδής*	*ψευδής*	*ψευδής*

105. **μπορεί να διαβαστεί από το πληκτρολόγιο η τιμή μιας λογικής μεταβλητής**; όχι. Από το πληκτρολόγιο διαβάζονται αριθμητικές τιμές, ή χαρακτήρες
106. **πως αλλιώς ονομάζονται οι λογικές μεταβλητές**; επειδή παίρνουν μόνο τις τιμές ΑΛΗΘΗΣ, ΨΕΥΔΗΣ, ονομάζονται και flags (σημαίες) επειδή και μια σημαία μπορεί να είναι ανεβασμένη ή κατεβασμένη
107. **τι σημαίνει η σύγκριση μεταξύ χαρακτήρων ή λέξεων**; σημαίνει ότι συγκρίνουμε τη λεξικογραφική τους διάταξη, η αλλιώς την αλφαβητική τους διάταξη. Μικρότερη θεωρείται εκείνη η λέξη που ξεκινάει από 'Α'.

Προγράμματα:

108. **τι ονομάζεται πρόγραμμα**; είναι ένας αλγόριθμος διατυπωμένος σε γλώσσα κατανοητή από τον υπολογιστή (γλώσσα προγραμματισμού) και που επεξεργάζεται δομές δεδομένων κατανοητές από τον υπολογιστή
109. **σχόλιο για τον προγραμματισμό:** Ο Η/Υ στην πραγματικότητα ξέρει μόνο να μεταφέρει δεδομένα απο/προς τη μνήμη του, να κάνει βασικές αριθμητικές πράξεις και συγκρίσεις. Η τεράστια ταχύτητά τος και ο προγραμματισμός με μια γλώσσα προγραμματισμού είναι αυτός που κάνει τον Η/Υ να φαίνεται «έξυπνος».
110. **εμείς ποιά γλώσσα προγραμματισμού μαθαίνουμε**; τη ΓΛΩΣΣΑ
111. **ποιά είναι η τυπική σύνταξη ενός προγράμματος στη ΓΛΩΣΣΑ**;

    πρόγραμμα *όνομα-προγράμματος*
    τμήμα-δήλωσης-σταθερών
    τμήμα-δήλωσης-μεταβλητών
    αρχή
    τμήμα-εντολών
    τέλος_προγράμματος
112. **Σχόλιο για τις εντολές**: άλλες είναι **εκτελεστέες** και άλλες είναι **δηλωτικές**
113. **τι είναι οι μεταβλητές**; μεταβλητή είναι μία παράμετρος του προβλήματος, η οποία μπορεί να αλλάζει καθώς το αντιμετωπίζουμε ή κάθε φορά που το αντιμετωπίζουμε. Τις μεταβλητές τις συμβολίζουμε με ονόματα που εμείς επιλέγουμε, ενώ πρακτικά για τον υπολογιστή κάθε μεταβλητή συμβολίζει τη θέση στη μνήμη του, όπου αποθηκεύεται η τιμή της μεταβλητής
114. **τι είναι οι (συμβολικές)σταθερές**; σταθερά είναι μία παράμετρος του προβλήματος, η οποία δεν αλλάζει ούτε καθώς το αντιμετωπίζουμε, ούτε κάθε φορά που το αντιμετωπίζουμε. Τις σταθερές τις συμβολίζουμε με ονόματα που εμείς επιλέγουμε, ενώ πρακτικά για τον υπολογιστή κάθε σταθερά συμβολίζει τη θέση στη μνήμη του, όπου αποθηκεύεται η τιμή της σταθεράς
115. **γιατί υπάρχει το τμήμα δήλωσης μεταβλητών**; για να γνωρίζει το πρόγραμμα πόσο χώρο από τη μνήμη του υπολογιστή θα χρειαστεί να δεσμεύσει, για να αποθηκεύσει στη συνέχεια τις τιμές των μεταβλητών που θα επεξεργαστεί. Επίσης, ανάλογα με την κατηγορία στην οποία ανήκει μια μεταβλητή, περιορίζονται και οι επεξεργασίες στις οποίες μπορεί να χρησιμοποιηθεί αργότερα (πχ πραγματική δε μπαίνει στο div, λογική δε μπαίνει στο διάβασε)
116. **γιατί υπάρχει το τμήμα δήλωσης σταθερών**; γιατί όταν έχουμε δηλώσει κάτι ως σταθερό, αλλά προσπαθήσουμε -κατά λάθος- να το αλλάξουμε, ο υπολογιστής δε θα μας αφήσει, ή γιατί μπορεί η χρήση σταθερών να κάνει τον κώδικά μας πιο ευανάγνωστο.
117. **ποιά είναι τα είδη των μεταβλητών (ή ποιοί είναι οι τύποι των δεδομένων)**; ακέραιες, πραγματικές, χαρακτήρες (ονομάζονται και αλφαριθμητικές), λογικές
118. **ποιές μεταβλητές πρέπει οπωσδήποτε να δηλωθούν ως ακέραιες**; εκείνες που χρησιμοποιούμε δεξιά-αριστερά από το div και το mod, και εκείνες που χρησιμεύουν ως δείκτες πινάκων
119. **τι τιμές παίρνει κάθε είδος**; οι ακέραιες μεταβλητές παίρνουν αριθμητικές τιμές χωρίς υποδιαστολή, οι πραγματικές μεταβλητές παίρνουν αριθμητικές τιμές με υποδιαστολή (το 2.0 θεωρείται πραγματική τιμή, όχι ακέραια), οι αλφαριθμητικές μεταβλητές παίρνουν οποιαδήποτε τιμή αλλά εντός εισαγωγικών, οι λογικές μεταβλητές παίρνουν ως τιμές μόνο τις λογικές σταθερές ΑΛΗΘΗΣ, ΨΕΥΔΗΣ (χωρίς εισαγωγικά)
120. **ποιοί κανόνες / περιορισμοί ισχύουν για τα ονόματα που μπορούμε να δώσουμε (σε μεταβλητές, σταθερές, προγράμματα, υποπρογράμματα)**; πρέπει να ξεκινάνε με γράμμα, εκτός από γράμματα επιτρέπεται να περιλαμβάνουν αριθμητικά ψηφία ή την κάτω παύλα, ενώ δε μπορεί να χρησιμοποιηθεί ως όνομα μια ήδη δεσμευμένη λέξη
121. **τι είναι οι δεσμευμένες λέξεις**; είναι οι λέξεις που ήδη χρησιμοποιούνται στα πλαίσια της γλώσσας προγραμματισμού για να συμβολίσουν κάτι άλλο (ΔΙΑΒΑΣΕ, ΓΡΑΨΕ, ΑΝ, ΤΟΤΕ, ΑΛΛΙΩΣ, ΤΕΛΟΣ_ΑΝ, ΑΛΛΙΩΣ_ΑΝ, ΟΣΟ, ΕΠΑΝΑΛΑΒΕ, ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ, ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ, ΜΕΧΡΙΣ_ΟΤΟΥ, ΓΙΑ, ΑΠΟ, ΜΕΧΡΙ, ΜΕ, ΒΗΜΑ, ΠΡΟΓΡΑΜΜΑ, ΣΤΑΘΕΡΕΣ,

ΜΕΤΑΒΛΗΤΕΣ, ΑΚΕΡΑΙΕΣ, ΠΡΑΓΜΑΤΙΚΕΣ, ΧΑΡΑΚΤΗΡΕΣ, ΛΟΓΙΚΕΣ, ΑΛΗΘΗΣ, ΨΕΥΔΗΣ, ΑΡΧΗ, ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ, ΔΙΑΔΙΚΑΣΙΑ, ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ, ΣΥΝΑΡΤΗΣΗ, ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ, ΚΑΛΕΣΕ, ΑΚΕΡΑΙΑ, ΠΡΑΓΜΑΤΙΚΗ, ΧΑΡΑΚΤΗΡΑΣ, ΛΟΓΙΚΗ, DIV, MOD, Α_Τ, Α_Μ, Τ_Ρ, ΗΜ, ΣΥΝ, ΕΦ, Ε, ΛΟΓ)

Δομές δεδομένων:

122. **τι ονομάζεται δομή δεδομένων**; είναι ένα σύνολο αποθηκευμένων δεδομένων που υποβάλουμε σε επεξεργασία από ένα σύνολο λειτουργιών
123. **τι λέει η εξίσωση του Wirth**: Επειδή υπάρχει μεγάλη εξάρτηση μεταξύ μιας δομής δεδομένων και των αλγορίθμων με τους οποίους γίνονται οι επεξεργασίες της, έχει διατυπωθεί η εξίσωση: **Αλγόριθμοι + Δομές Δεδομένων = Προγράμματα** *(εξίσωση του Wirth)*
124. **από τί αποτελείται γενικά μία δομή δεδομένων**; από κόμβους, στους οποίους αποθηκεύονται τα επιμέρους δεδομένα
125. **σε τι διαφέρουν οι στατικές με τις δυναμικές δομές δεδομένων**; οι στατικές έχουν αμετάβλητη χωρητικότητα που καθορίζεται από τον δημιουργό του προγράμματος και καταλαμβάνουν συνεχόμενες θέσεις στη μνήμη του υπολογιστή, ενώ οι δυναμικές έχουν μεταβλητή χωρητικότητα που καθορίζεται από τον χρήστη ενός προγράμματος και καταλαμβάνουν διάσπαρτες θέσεις στη μνήμη του υπολογιστή
126. **τι είναι οι τεχνικές δυναμικής παραχώρησης μνήμης**; χρησιμεύουν στην επεξεργασία των δυναμικών δομών δεδομένων, ουσιαστικά είναι εντολές με τις οποίες ζητάμε να δεσμευθεί ή να αποδεσμευθεί μνήμη
127. **σε τί διαφέρει η κύρια με τη δευτερεύουσα μνήμη**; η κύρια μνήμη (ram) είναι πολύ ταχύτερη, αλλά χάνει τα δεδομένα της όταν δεν έχει ρεύμα, η δευτερεύουσα (πχ σκληρός δίσκος) είναι πιο αργή, αλλά διατηρεί αποθηκευμένα τα δεδομένα και χωρίς να έχει ρεύμα
128. **ποιές είναι οι οκτώ επεξεργασίες επί των δομών δεδομένων γενικά**; προσπέλαση, εισαγωγή, διαγραφή, αντιγραφή, ταξινόμηση, αναζήτηση, συγχώνευση, διαχωρισμός
129. **ποιά είναι η ερμηνεία/λειτουργία τους**;
    - **προσπέλαση** σημαίνει να έχεις πρόσβαση σε κάποιο κόμβο για να δεις ή να αλλάξεις τα περιεχόμενα δεδομένα του,
    - **εισαγωγή** σημαίνει να προσθέσεις στη δομή δεδομένων ένα νέο κόμβο με δεδομένα,
    - **διαγραφή** (αντίστροφο της εισαγωγής) σημαίνει να αφαιρέσεις από τη δομή δεδομένων έναν κόμβο,
    - **αντιγραφή** σημαίνει τα περιεχόμενα ενός κόμβου να αποθηκευθούν και σε μια άλλη δομή δεδομένων,
    - **ταξινόμηση** σημαίνει οι κόμβοι της δομής δεδομένων να μπουν σε αύξουσα (αλφαβητική) ή φθίνουσα διάταξη,
    - **αναζήτηση** σημαίνει να εξεταστούν οι κόμβοι μιας δομής ώστε να εντοπιστεί κάποιος με μια ιδιότητα,
    - **συγχώνευση** σημαίνει να ενώσουνε δύο ή περισσότερες δομές σε μία,
    - **διαχωρισμός** (αντίστροφο της συγχώνευσης) σημαίνει να επιλεχθούν κάποιοι κόμβοι της δομής δεδομένων ώστε να αποθηκευθούν σε μία άλλη.
130. **εφαρμόζονται και οι οκτώ σε όλες τις δομές δεδομένων**; Σπάνια. Το ποιές επεξεργασίες εφαρμόζονται εξαρτάται από τα ίδια τα δεδομένα (πχ γιατί να ταξινομήσεις αλφαβητικά τις λέξεις μιας πρότασης;). Επίσης μπορεί μια δομή δεδομένων να είναι αποδοτικότερη σε κάποια επεξεργασία μόνο. Γι' αυτό υπάρχουν και διάφορες δομές δεδομένων
131. **ποιές επεξεργασίες δεν γίνονται σε στατικές δομές**; η εισαγωγή και η διαγραφή, γιατί θα σήμαιναν αύξηση ή μείωση της χωρητικότητας
132. **ποιές δομές δεδομένων γνωρίζετε**; αρχεία, πίνακες, στοίβες, ουρές, λίστες, δένδρα, γράφοι
133. **Θεωρία Πληροφοριών (Information Theory):** Η μέτρηση, η κωδικοποίηση, η μετάδοση της πληροφορίας αποτελεί αντικείμενο μελέτης ενός ιδιαίτερου κλάδου, της Θεωρίας Πληροφοριών, που είναι ένα ιδιαίτερα σημαντικό πεδίο της Πληροφορικής
134. **Ποιές είναι οι 4 σκοπιές από τις οποίες μελετά η πληροφορική τα δεδομένα:** Υλικού, Γλωσσών προγραμματισμού, Δομών δεδομένων, Ανάλυσης δεδομένων
135. **Τι σημαίνει η μελέτη των δεδομένων από τη σκοπιά του υλικού:** πώς μπορεί να κωδικοποιηθεί ένα δεδομένο για να αποθηκευθεί στο υλικό, δηλαδή στην κύρια ή στις δευτερεύουσες μνήμες του υπολογιστή.
136. **Τι σημαίνει η μελέτη των δεδομένων από τη σκοπιά των γλωσσών προγραμματισμού:** ποιά γλώσσα προγραμματισμού υποστηρίζει τους τύπους των δεδομένων που θέλουμε να χρησιμοποιήσουμε, και την αποδοτικότερη χρήση και αποθήκευσή τους.

137. **Τι σημαίνει η μελέτη των δεδομένων από τη σκοπιά των Δομών Δεδομένων:** μελετά ποιά είναι τα επιμέρους δεδομένα και πώς αυτά μπορεί να συσχετιστούν και να ομαδοποιηθούν (σε μία δομή δεδομένων), ώστε μαζί να περιγράφουν αυτό που πρέπει να επεξεργαστούμε στο κάθε πρόβλημα.
138. **Τι σημαίνει η μελέτη των δεδομένων από τη σκοπιά της Ανάλυσης Δεδομένων**: πώς μπορούν να καταγράφονται αποδοτικά και να συσχετίζονται μεταξύ τους τα διαφορετικά δεδομένα, ώστε να εξάγουμε από αυτά πληροφορίες, με τις οποίες αναπαριστούμε τη γνώση μας για απτά, πραγματικά γεγονότα, πρόσωπα ή αντικείμενα. Με την ανάλυση δεδομένων ασχολούνται οι τεχνολογίες των Βάσεων Δεδομένων, της Μοντελοποίησης Δεδομένων και της Αναπαράστασης Γνώσης.

Πίνακες:

139. **τι είναι ο πίνακας**; είναι μία στατική δομή δεδομένων, με δεδομένα ίδιου τύπου. Σε ένα πίνακα τα στοιχεία του όλα συμβολίζονται με το ίδιο όνομα αλλά τα ξεχωρίζουμε με τη βοήθεια ενός (ή παραπάνω) αριθμητικού δείκτη
140. **τι σημαίνει στατική:** ότι το μέγεθός του προκαθορίζεται από τον προγραμματιστή την ώρα δημιουργίας του προγράμματος, δε μπορεί να μεταβληθεί κατά τη χρήση του προγράμματος, ενώ ο απαιτούμενος χώρος στη μνήμη θα είναι συνεχόμενος.
141. **από τι αποτελείται ο πίνακας**; από κελιά, σε καθένα από τα οποία αποθηκεύεται μία τιμή
142. **τι δείχνει ο «δείκτης» ενός πίνακα**; δείχνει σε ποιό κελί (σε ποιά θέση) του πίνακα αναφερόμαστε
143. **ποιά πλεονεκτήματα έχει η χρήση των πινάκων**; θετικό είναι ότι μπορούμε να αποθηκεύσουμε έναν όγκο δεδομένων για να τα ξαναχρησιμοποιήσουμε αργότερα, ενώ η επεξεργασία τους γίνεται εύκολα με τυποποιημένες μεθοδολογίες με δομές επανάληψης
144. **και ποιά μειονεκτήματα έχει η χρήση πινάκων**; αρνητικό είναι ότι λόγω στατικότητας, ένας πίνακας μικρού μεγέθους περιορίζει τις δυνατότητες ενός προγράμματος, ενώ ένας πίνακας μεγάλου μεγέθους οδηγεί σε σπατάλη της μνήμης του υπολογιστή. Θεωρητικά, η άσκοπη χρήση πινάκων μπορεί να οδηγήσει ακόμα και σε αδυναμία εκτέλεσης ενός προγράμματος, λόγως έλλειψης μνήμης
145. **πότε πρέπει να χρησιμοποιούμε πίνακες**; όταν χρειάζεται τα δεδομένα μας να διατηρηθούν στη μνήμη, ώστε παρακάτω να τα ξαναεπεξεργαστούμε
146. **πότε μπορούμε να χρησιμοποιούμε πίνακες**; σε όποιο πρόβλημα είναι από την αρχή γνωστό το ακριβές πλήθος (ή το μέγιστο πλήθος) των δεδομένων που θα χρειαστεί να επεξεργαστούμε
147. **ποιές είναι οι πέντε τυπικές επεξεργασίες πινάκων**; υπολογισμός αθροίσματος, εντοπισμός μέγιστης/ελάχιστης τιμής, ταξινόμηση, αναζήτηση, συγχώνευση
148. **υπάρχει περιορισμός στις διαστάσεις (στους δείκτες) ενός πίνακα**; θεωρητικά όχι, μπορώ να φτιάξω ν-διάστατους πίνακες, χρησιμοποιώντας περισσότερους δείκτες (και αντίστοιχα περισσότερες δομές επανάληψης)
149. **ποιοί πίνακες ονομάζονται τετραγωνικοί**; οι δισδιάστατοι πίνακες όπου το πλήθος των γραμμών τους ισούται με το πλήθος των στηλών τους
150. **ποιοί πίνακες ονομάζονται στήλες**; οι μονοδιάστατοι πίνακες που τα κελιά τους τα σχεδιάζουμε το ένα κάτω απ'το άλλο
151. **ποιοί πίνακες ονομάζονται γραμμές**; οι μονοδιάστατοι πίνακες που τα κελιά τους τα σχεδιάζουμε το ένα δίπλα στο άλλο
152. **ποιοί πίνακες ονομάζονται αραιοί**; οι πίνακες οι οποίοι είναι κατά 80% περίπου κενοί
153. **ποιοί πίνακες ονομάζονται παράλληλοι**; παράλληλοι είναι οι πίνακες που έχουν μία τουλάχιστον διάσταση κοινή, και υπάρχει μεταξύ τους μία λογική συσχέτιση
154. **τι τιμές επιτρέπεται να παίρνουν οι δείκτες των πινάκων**; αποκλειστικά ακέραιες τιμές (και οι δείκτες να είναι δηλωμένες ακέραιες μεταβλητές), ανάμεσα στο 1 και στο μέγεθος της διάστασης του πίνακα
155. **σε ένα δισδιάστατο πίνακα τί δείχνει ο πρώτος δείκτης και τί ο δεύτερος**; ο πρώτος υποδηλώνει πάντα τη γραμμή στην οποία αναφερόμαστε, και ο δεύτερος υποδηλώνει τη στήλη
156. **ποιά είναι η σχέση που χαρακτηρίζει τις συντεταγμένες των κελιών που ανήκουν στην κύρια διαγώνιο ενός τετράγωνου πίνακα NxN**; για ένα κελί Α[x,ψ] που ανήκει στην κύρια διαγώνιο, ισχύει «x=ψ»
157. **ποιά είναι η σχέση που χαρακτηρίζει τις συντεταγμένες των κελιών που ανήκουν στη δευτερεύουσα διαγώνιο ενός τετράγωνου πίνακα NxN**; για ένα κελί Α[x,ψ] που ανήκει στη δευτερεύουσα διαγώνιο ισχύει «x+ψ=N+1»

158. **τι διαφορά έχει η συγχώνευση πινάκων από τη συγχώνευση δομών δεδομένων γενικά**; στους πίνακες λέμε ότι συγχωνεύονται ταξινομημένοι πίνακες, και ο πίνακας που δημιουγείται από τη συγχώνευση είναι κι αυτός ταξινομημένος, ενώ γενικά στις δομές δεδομένων η ταξινόμηση δεν εμπλέκεται με την συγχώνευση

159. **πώς λειτουργεί η σειριακή (γραμμική) αναζήτηση μια τιμής Τ σε ένα πίνακα Π μεγέθους Ν**; περνάμε ένα-ένα τα κελιά, μέχρι να εντοπίσουμε την τιμή Τ, και μόλις την εντοπίσουμε σταματάμε

160. **ποιά είναι η γενική μορφή αυτής της αναζήτησης**;

τρόπος βιβλίου
```
x ← 1
found ← ψευδής
όσο x<=N και found=ψευδής επανάλαβε
 αν Π[x] = Τ τότε
  found ← αληθής
 αλλιώς
  x ← x+1
 τέλος_αν
τέλος_επανάληψης
```

απλούστερος τρόπος
```
x ← 0
αρχή_επανάληψης
 x ← x+1
μέχρις_ότου Π[x]=Τ ή x=N
```

161. **πώς λειτουργεί η δυαδική αναζήτηση μιας τιμής Τ σε ένα πίνακα Π μεγέθους Ν (ταξινομημένο σε αύξουσα σειρά)**; συγκρίνουμε την αναζητούμενη τιμή, με τη μεσαία τιμή του εξεταζόμενου πίνακα. Ανάλογα με το αποτέλεσμα της σύγκρισης, αποφασίζουμε αν θα επαναλάβουμε την αναζήτηση στο κομμάτι πριν από τη μέση, ή στο κομμάτι μετά από τη μέση, ή αν θα σταματήσουμε

162. **ποιά είναι η γενική μορφή αυτής της αναζήτησης**;

```
left ← 1
right ← N
found ← ψευδής
όσο found=ψευδής και left<= right επανάλαβε
 mid ← (left+right) div 2
 αν Π[mid] = Τ τότε
  found ← αληθής
 αλλιώς_αν Π[mid] < Τ τότε
  left ← mid+1
 αλλιώς
  right ← mid-1
 τέλος_αν
τέλος_επανάληψης
```

163. **πότε χρησιμοποιούμε τη σειριακή αναζήτηση**; όταν η δομή δεδομένων είναι μικρού μεγέθους, όταν κάνουμε σπάνια αναζήτηση, και -αναγκαστικά- όταν τα δεδομένα είναι αταξινόμητα

164. **η ταξινόμηση εξυπηρετεί την αναζήτηση**; ναι, λέγεται ότι ο σκοπός της ταξινόμησης είναι να διευκολύνει την αναζήτηση

165. **πώς λειτουργεί η ταξινόμηση με τη μέθοδο της φυσαλίδας (μέθοδος ευθείας ανταλλαγής)**; βασίζεται στη σύγκριση γειτονικών στοιχείων και στην αντιμετάθεσή τους όταν δε βρίσκονται στη σωστή σειρά. Τα γειτονικά ζεύγη συγκρίνονται όλα, ξεκινώντας από το τελευταίο κάθε φορά. Και κάθε φορά που γίνεται αυτό, ένα στοιχείο μετακινείται προς τη σωστή θέση.

166. **ποιά είναι η γενική μορφή ταξινόμησης ενός πίνακα Π μεγέθους Ν σε αύξουσα σειρά με τη μέθοδο της φυσαλίδας**;

```
για ψ από 2 μέχρι Ν
 για x από Ν μέχρι ψ με βήμα -1
  αν Π[x] < Π[x-1] τότε
   αντιμετάθεσε Π[x],Π[x-1]
  τέλος_αν
 τέλος_επανάληψης
τέλος_επανάληψης
```

167. **πώς λειτουργεί η «έξυπνη» φυσαλίδα**; με τη βοήθεια ενός μετρητή, ή μιας λογικής μεταβλητής, εξετάζουμε αν γίνανε ή δε γίνανε αντιμεταθέσεις. Αν δε γίνανε αντιμεταθέσεις, ο αλγόριθμος τερματίζεται, γιατί σημαίνει ότι τα στοιχεία είναι ήδη ταξινομημένα

168. **ποιά είναι η γενική μορφή της παραλλαγής της «έξυπνης» φυσαλίδας**;

```
ψ ← 2
αρχή_επανάληψης
   έγινεΑντιμετάθεση ← ψευδής
   για x από Ν μέχρι ψ με βήμα -1
    αν Π[x] < Π[x-1] τότε
      αντιμετάθεσε Π[x],Π[x-1]
      έγινεΑντιμετάθεση ← αληθής
    τέλος_αν
  τέλος_επανάληψης
  ψ ← ψ+1
μέχρις_ότου ψ>Ν ή έγινεΑντιμετάθεση=ψευδής
```

169. **θεωρείται καλή μέθοδος ταξινόμησης, η μέθοδος της φυσαλίδας**; είναι ο πιο απλός, αλλά ταυτόχρονα ο πιο αργός τρόπος ταξινόμησης
170. **υπάρχουν άλλες μέθοδοι ταξινόμησης**; αναφέρονται η μέθοδος της επιλογής, η μέθοδος της εισαγωγής, η «γρήγορη ταξινόμηση» και διαλέγουμε κάποιον ανάλογα με το είδος των στοιχείων, ανάλογα με το πλήθος των στοιχείων ή ανάλογα με την αρχική τους διάταξη
171. **πώς λειτουργεί η ταξινόμηση με τη μέθοδο της επιλογής**; περνάμε τα στοιχεία ώστε να βρούμε το 1ο μικρότερο (μεγαλύτερο) και να το βάλουμε στην 1η θέση, να βρούμε το 2ο μικρότερο (μεγαλύτερο) και να το βάλουμε στη 2η θέση, κοκ
172. **ποιά είναι η γενική μορφή ταξινόμησης ενός πίνακα Π μεγέθους Ν σε αύξουσα σειρά με τη μέθοδο της επιλογής**;

```
για ψ από 1 μέχρι Ν-1
 θέση ← ψ
 ελαχ ← Π[ψ]
 για x από ψ+1 μέχρι Ν
  αν Π[x] < ελαχ τότε
   ελαχ ← Π[x]
   θέση ← x
  τέλος_αν
 τέλος_επανάληψης
 αντιμετάθεσε Π[ψ], Π[θέση]
τέλος_επανάληψης
```

173. **ποιές εντολές ισοδυναμούν με την εντολή «αντιμετάθεσε α,β»**

```
γ ← α
α ← β
β ← γ
```

Στοίβες:

174. **τι είναι η στοίβα:** δομή δεδομένων που χρησιμεύει όταν θέλουμε να αποθηκεύσουμε στοιχεία για να τα επαναφέρουμε με την αντίστροφη σειρά από τη σειρά αποθήκευσης
175. **πώς ονομάζεται αυτή η λειτουργία:** LIFO (last in first out)
176. **ποιές είναι οι δύο λειτουργίες που γίνονται σε μία στοίβα:** ώθηση (push), απώθηση (pop)
177. **τι γίνεται κατά την ώθηση:** πρώτα ελέγχεται μήπως η στοίβα είναι ήδη γεμάτη (έλεγχος υπερχείλισης), αν δεν είναι, το νέο στοιχείο ωθείται στην κορυφή της
178. **τι γίνεται κατά την απώθηση:** πρώτα ελέγχεται μήπως η στοίβα είναι ήδη άδεια (έλεγχος υποχείλισης), αν δεν είναι, απωθείται ένα στοιχείο από την κορυφή της
179. **πώς υλοποιείται συνήθως μία στοίβα:** ως μονοδιάστατος πίνακας (θα μπορούσε και ως λίστα)
180. **πόσους δείκτες χρειαζόμαστε για την υλοποίηση μίας στοίβας:** έναν δείκτη που τυπικά ονομάζεται **top**
181. **τί μας δείχνει αυτός ο δείκτης**: δείχνει σε ποιό σημείο της στοίβας έχει φτάσει η κορυφή της (ή πιο απλά, πόσα στοιχεία είναι μέσα)
182. **Παράδειγμα κωδικοποίησης ώθησης και απώθησης**:

Θεωρώντας ότι η στοίβα είναι ένας μονοδιάστατος πίνακας Α με Ν κελιά, και θέλουμε να **ωθήσουμε** στην κορυφή της την τιμή Χ:	Θεωρώντας ότι η στοίβα είναι ένας μονοδιάστατος πίνακας Α, και η τιμή που θα **απωθηθεί** από την κορυφή της θα ονομαστεί Ψ:

```
ΑΝ top < N ΤΟΤΕ
  top ← top + 1
  A[top]← X
  ΓΡΑΨΕ 'ωθήθηκε η νέα τιμή'
  overflow ← ψευδής
ΑΛΛΙΩΣ
  ΓΡΑΨΕ 'δεν ωθήθηκε η νέα τιμή'
  overflow ← αληθής
ΤΕΛΟΣ_ΑΝ
```

```
ΑΝ top >=1 ΤΟΤΕ
  Ψ ← A[top]
  top ← top - 1
  underflow ← ψευδής
  ΓΡΑΨΕ 'απωθήθηκε η τιμή', Ψ
ΑΛΛΙΩΣ
  ΓΡΑΨΕ 'δεν απωθήθηκε καμία τιμή'
  underflow ← αληθής
ΤΕΛΟΣ_ΑΝ
```

183. **Σχόλιο για τη στοίβα μέσω πίνακα:** όταν γίνεται απώθηση, στην πραγματικότητα το στοιχείο που απωθείται παραμένει μέσα στον πίνακα, απλά είναι σαν να μην υπάρχει, επειδή εμείς κάνουμε προσπέλαση μόνο στη θέση top, η οποία θα έχει πλέον αλλάξει.

Ουρές:

184. **τι είναι η ουρά:** δομή δεδομένων που χρησιμεύει όταν θέλουμε να αποθηκεύσουμε στοιχεία για να τα επαναφέρουμε με την ίδια σειρά από τη σειρά αποθήκευσης
185. **πώς ονομάζεται αυτή η λειτουργία:** FIFO (first in first out)
186. **ποιές είναι οι δύο λειτουργίες που γίνονται σε μία ουρά:** εισαγωγή (enqueue), εξαγωγή (dequeue)
187. **τι γίνεται κατά την εισαγωγή:** πρώτα ελέγχεται μήπως η ουρά είναι ήδη γεμάτη, αν δεν είναι, το νέο στοιχείο τοποθετείται στο πίσω μέρος της ουράς
188. **τι γίνεται κατά την εξαγωγή:** πρώτα ελέγχεται μήπως η ουρά είναι ήδη άδεια, αν δεν είναι, εξάγεται ένα στοιχείο από το εμπρός μέρος της ουράς
189. **πώς υλοποιείται συνήθως μία ουρά:** ως μονοδιάστατος πίνακας (θα μπορούσε και ως λίστα)
190. **πόσους δείκτες χρειαζόμαστε για την υλοποίηση μίας ουράς:** δύο δείκτες που τυπικά ονομάζονται **front** και **rear**
191. **τί μας δείχνουν αυτοί οι δείκτες**: ο **front** δείχνει σε ποιά θέση βρίσκεται το πρώτο στοιχείο της ουράς, δηλαδή εκείνο που έχει σειρά να φύγει, ο **rear** δείχνει σε ποιά θέση βρίσκεται το τελευταίο στοιχείο της ουράς, δηλαδή εκείνο που μπήκε τελευταίο
192. **Παράδειγμα κωδικοποίησης εισαγωγής και εξαγωγής**:

θέλουμε να **εισάγουμε** την τιμή X σε μια ουρά μεγέθους N:

```
ΑΝ rear = N ΤΟΤΕ
  ΓΡΑΨΕ 'δεν υπάρχει χώρος'
ΑΛΛΙΩΣ_ΑΝ rear = 0 ΚΑΙ front=0 ΤΟΤΕ
  rear ← 1
  front ← 1
  A[rear] ← X
  ΓΡΑΨΕ 'μπήκε το', X
ΑΛΛΙΩΣ
  rear ← rear + 1
  A[rear] ← X
  ΓΡΑΨΕ 'μπήκε το', X
ΤΕΛΟΣ_ΑΝ
```

θέλουμε να βάλουμε στο X την τιμή που **εξάγουμε** από την ουρά:

```
ΑΝ rear = 0 ΚΑΙ front=0 ΤΟΤΕ
  ΓΡΑΨΕ 'η ουρά είναι άδεια'
ΑΛΛΙΩΣ_ΑΝ rear = front ΤΟΤΕ
  X ← A[front]
  ΓΡΑΨΕ 'εξάγεται η τιμή', X
  rear ← 0
  front ← 0
ΑΛΛΙΩΣ
  X ← A[front]
  ΓΡΑΨΕ 'εξάγεται η τιμή', X
  front ← front + 1
ΤΕΛΟΣ_ΑΝ
```

193. **Υπάρχουν εναλλακτικές κωδικοποιήσεις**: Ναι, η παραπάνω είναι η πιο απλοϊκή, όπου η ουρά θεωρείται γεμάτη όταν ο δείκτης rear φτάσει στην τελευταία θέση. Εναλλακτικά θα μπορούσε:
    - σε κάθε εξαγωγή να μετακινεί όλα τα στοιχεία της ουράς μία θέση πιο μπροστά,
    - ή όταν ο δείκτης rear έφτανε στην τελευταία θέση, να μετακινούσε όλα τα στοιχεία front-1 θέσεις πιο μπροστά
    - ή αντί να μηδενίζει rear,front όταν η ουρά είναι άδεια, να άφηνε το rear να είναι μικρότερο από το front
194. **Σχόλιο για την ουρά μέσω πίνακα:** όταν γίνεται εξαγωγή, στην πραγματικότητα το στοιχείο που εξαγεται παραμένει μέσα στον πίνακα, απλά είναι σαν να μην υπάρχει, επειδή εμείς κάνουμε προσπέλαση μόνο στη θέση front, η οποία θα έχει πλέον αλλάξει.

Αρχεία:

195. **τι ονομάζεται αρχείο**; αρχείο είναι κάθε δομή δεδομένων που αποθηκεύεται στη δευτερεύουσα μνήμη.
196. **σε τί χωρίζεται**; Αποτελείται από επιμέρους εγγραφές, οι οποίες με τη σειρά τους αποτελούνται από πεδία
197. **ποιά πεδία είναι «κλειδιά»**; κλειδί ονομάζεται ένα πεδίο όταν χρησιμεύει στην ταυτοποίηση μιας εγγραφής (μπορεί να υπάρχει πρωτεύον, δευτερεύον κλπ)

Λίστες:

198. **τί είναι η λίστα**: μία δυναμική, γραμμική δομή δεδομένων
199. **από τι αποτελείται**: από κόμβους
200. **από τί αποτελείται ένας κόμβος**: κατά ένα μέρος από τα δεδομένα που έχουμε αποθηκεύσει σε αυτόν, και κατά το άλλο μέρος από δείκτες (έναν ή δύο)
201. **τί δείχνει ένας τέτοιος δείκτης**: δείχνει σε ποιά διεύθυνση της μνήμης του υπολογιστή είναι αποθηκευμένος ένας άλλος κόμβος
202. **ποιά είδη λίστας γνωρίζετε**; την απλά συνδεδεμένη λίστα (που συνήθως αναφέρεται ως λίστα) και τη διπλά συνδεδεμένη λίστα
203. **σε μία απλά συνδεδεμένη λίστα, τί δείχνει ο δείκτης κάθε κόμβου**; δείχνει πού βρίσκεται ο επόμενος κόμβος της λίστας
204. **σε μία διπλά συνδεδεμένη λίστα, τί δείχνουν οι δείκτες κάθε κόμβου**; ο ένας δείχνει πού βρίσκεται ο επόμενος κόμβος κι ο άλλος δείχνει πού βρίσκεται ο προηγούμενος κόμβος της λίστας
205. **κι όταν δεν υπάρχει επόμενος ή προηγούμενος κόμβος**; ο αντιστοιχος δείκτης δε δείχνει πουθενά (είναι null pointer)
206. **πώς αποκτούμε πρόσβαση σε μία απλά συνδεδεμένη λίστα**; με τη βοήθεια ενός δείκτη που συνήθως ονομάζεται κεφαλή (head) ο οποίος δείχνει πού βρίσκεται ο πρώτος κόμβος της λίστας
207. **πώς αποκτούμε πρόσβαση σε μία διπλά συνδεδεμένη λίστα**; με τη βοήθεια δύο δεικτών που συνήθως ονομάζονται κεφαλή (head) και ουρά (tail), που ο ένας δείχνει πού βρίσκεται ο πρώτος κόμβος της λίστας και ο άλλος δείχνει πού βρίσκεται ο τελευταίος κόμβος της λίστας
208. **Ποιές είναι οι βασικές πράξεις των συνδεδεμένων λιστών**;
    - Εισαγωγή κόμβου στη λίστα (στην αρχή, στο τέλος της λίστας ή ενδιάμεσα).
    - Διαγραφή κόμβου από τη λίστα (από την αρχή, το τέλος της λίστας ή ενδιάμεσα).
    - Έλεγχος για το αν η λίστα είναι κενή.
    - Αναζήτηση κόμβου για την εύρεση συγκεκριμένου στοιχείου.
    - Διάσχιση της λίστας και προσπέλαση των στοιχείων της.
209. **τι πλεονεκτήματα έχουν οι λίστες έναντι των πινάκων**;
    - το δυναμικό τους μέγεθος,
    - η ευκολία εισαγωγής και διαγραφής από οποιοδήποτε μέρος της λίστας
    - η μη αναγκαιότητα δήλωσης του μεγέθους τους
210. **τι μειονεκτήματα έχουν οι λίστες έναντι των πινάκων**;
    - Η τυχαία πρόσβαση στη λίστα δεν επιτρέπεται. Για να προσπελάσεις ένα συγκεκριμένο κόμβο πρέπει νωρίτερα να περάσεις από όλους τους προηγούμενους (ή επόμενους)
    - είναι λιγότερο αποτελεσματικές στη χρήση της μνήμης, αφού απαιτούν έξτρα μνήμη για την αποθήκευση των δεικτών

Δένδρα:

211. **Τί είναι το δένδρο**: μια δομή δεδομένων που αναπαριστά κάποια ιεραρχική σχέση μεταξύ των αποθηκευμένων δεδομένων
212. **Από τί αποτελείται**: ουσιαστικά από δεδομένα και δείκτες, αλλά εδώ αυτά πάλι ονομάζονται αντίστοιχα κόμβοι και ακμές
213. **Τι ονομάζουμε ρίζα**: τον αρχικό κόμβο του δένδρου, δεν έχει γονέα, από αυτόν ξεκινάει κάθε διαδρομή και κάθε πρόσβαση στο δένδρο
214. **Τι ονομάζουμε γονέα και τι παιδί**: γονέας είναι ο κόμβος από τον οποίο ξεκινά μία ακμή και παιδί είναι ο κόμβος στον οποίο καταλήγει μία ακμή. Ο γονέας τοποθετείται σε επίπεδο υψηλότερο από το παιδί. Ένα παιδί έχει μόνο έναν γονέα, ενώ ένας γονέας μπορεί να έχει πολλά παιδιά.

215. **Τι ονομάζουμε διαδρομή για έναν κόμβο**: μια ακολουθία διαδοχικών ακμών που συνδέουν αυτόν τον κόμβο με τη ρίζα του δέντρου
216. **Τι ονομάζουμε πρόγονο και τι απόγονο ενός κόμβου x**: πρόγονος είναι κάποιος κόμβος από τον οποίο ξεκινά μια διαδρομή που καταλήγει στον x, ενώ απόγονος είναι κάθε κόμβος που βρίσκεται σε κάποια διαδρομή που ξεκινά από τον x
217. **Τι ονομάζουμε υποδένδρο**: είναι το μικρότερο εκείνο δέντρο που σχηματίζεται όταν ως ρίζα θεωρήσουμε έναν κόμβο του αρχικού δέντρου.
218. **Υπάρχει δένδρο χωρίς ρίζα**: ναι, το κενό δένδρο
219. **Τι ονομάζουμε φύλλα**: εκείνους τους κόμβους που δεν έχουν παιδιά
220. **Τι ονομάζουμε αδέλφια**: τους κόμβους που έχουν τον ίδιο γονέα
221. **Τι ονομάζουμε δυαδικό δένδρο**: το δέντρο που ο κάθε κόμβος μπορεί να έχει το πολύ δύο παιδιά (σε αυτά τα παιδιά τυπικά αναφερόμαστε ως αριστερό-δεξί)
222. **Πότε ονομάζουμε ένα δένδρο διατεταγμένο**: όταν έχει σημασία η διάταξη (σειρά) με την οποία αναφέρονται τα παιδιά ενός κόμβου (από αριστερά προς τα δεξιά)
223. **Τι ονομάζουμε δυαδικό δένδρο αναζήτησης**: είναι το διατεταγμένο δυαδικό δέντρο στο οποίο για κάθε κόμβο του ισχύει ότι το αριστερό του υποδέντρο έχει τιμές μικρότερες από τη δική του, ενώ το δεξιό του υποδέντρο έχει τιμές μεγαλύτερες(ίσες) με τη δική του.
224. **Σε τί πλεονεκτούν τα δυαδικά δέντρα αναζήτησης**: συνδυάζουν τα πλεονεκτήματα των λιστών, όσον αφορά τις πράξεις της εισαγωγής και της διαγραφής, αλλά και τα πλεονεκτήματα των ταξινομημένων πινάκων, όσον αφορά την πράξη της αναζήτησης
225. **Άλλη εφαρμογή των δένδρων**:
    - σε αλγορίθμους μηχανικής μάθησης χρησιμοποιούνται δέντρα απόφασης,
    - σε διερμηνευτές γλωσσών, χρησιμοποιούνται δέντρα συντακτικής ανάλυσης,
    - σε παιχνίδια με τεχνητή νοημοσύνη, χρησιμοποιούνται τα δέντρα παιχνιδιού για την αναπαράσταση πιθανών επόμενων κινήσεων

Γράφοι:

226. **Τί είναι ο γράφος**: Δομή δεδομένων που χρησιμοποιούμε για να αναπαραστήσουμε σύνθετες σχέσεις μεταξύ δεδομένων
227. **Από τί αποτελείται ένας γράφος**: από κόμβους και γραμμές που συνδέουν τους κόμβους. Οι κόμβοι ονομάζονται κορυφές ή σημεία, ενώ οι γραμμές ονομάζονται τόξα ή ακμές
228. **Ένα δένδρο μπορεί να θεωρηθεί γράφος**; Ναι, είναι είδος γράφου στο οποίο δυο κόμβοι συνδέονται μεταξύ τους με μία μόνο διαδρομή
229. **Μία λίστα μπορεί να θεωρηθεί γράφος**; Ναι, είναι είδος γράφου στο οποίο όλοι οι κόμβοι εντάσσονται σε μία διαδρομή
230. **Πώς αναπαρίσταται ένας γράφος σχηματικά**; αναπαρίσταται με κάποιες κορυφές (ή σημεία), οι οποίες συνδέονται μεταξύ τους με κάποιες ακμές (ή τόξα) (μπορεί να υπάρχει κάποια κορυφή που να μη συνδέεται με άλλες)
231. **Τί ακμές μπορεί να έχει ένας γράφος:** κατευθυνόμενες ή μή κατευθυνόμενες
232. **Ποιά είναι η διαφορά**: Μία κατευθυνόμενη ακμή (που σχηματικά συμβολίζεται με βελάκι) μπορείς να κινηθείς μόνο προς τη μία κατεύθυνση (που δείχνει το βελάκι)
233. **πότε είναι ο γράφος κατευθυνόμενος**; όταν όλες οι ακμές του είναι κατευθυνόμενες
234. **πότε είναι ο γράφος μη-κατευθυνόμενος**; όταν όλες οι ακμές του είναι μη-κατευθυνόμενες
235. **Εφαρμογή γράφων**: Χρησιμεύουν σε ποικίλα και σύνθετα προβλήματα, γι' αυτό η Θεωρία Γράφων αποτελεί ξεχωριστό επιστημονικό τομέα

Τεχνικές προγραμματισμού / Τεχνικές σχεδίασης προγραμμάτων

236. **ποιές είναι οι τρεις κλασικές τεχνικές προγραμματισμού**; ιεραρχικός προγραμματισμός (ή αλλιώς ιεραρχική σχεδίαση, ή αλλιώς σχεδίαση από πάνω προς τα κάτω), τμηματικός προγραμματισμός, δομημένος προγραμματισμός
237. **τι προτείνει ο ιεραρχικός προγραμματισμός**; να αναλύουμε τις βασικές λειτουργίες ενός προγράμματος σε όσο γίνεται μικρότερες και πιο στοιχειώδεις λειτουργίες, που είναι ευκολότερο να υλοποιηθούν
238. **τι προτείνει ο τμηματικός προγραμματισμός**; να θεωρούμε το κάθε υποπρόβλημα ως μία ξεχωριστή ενότητα, η λύση του οποίου υλοποιείται χωριστά και μπορεί να χρησιμοποιηθεί σε διαφορετικά προβλήματα/προγράμματα.

239. **τι προτείνει ο δομημένος προγραμματισμός**; να στηρίζουμε τη δημιουργία κάθε προγράμματος/υποπρογράμματος αποκλειστικά στις τρεις βασικές αλγοριθμικές δομές (δομές ακολουθίας, επιλογής, επανάληψης), ενώ απαγορεύει την ελεύθερη χρήση εντολών goto
240. **ποιοί πρότειναν τις αρχές του δομημένου προγραμματισμού**: Οι Bohm και Jacopini το 1964
241. **πότε καθιερώθηκε ο δομημένος προγραμματισμός**; από το 1968, όταν ο Dijkstra δημοσίευσε ένα άρθρο κατά της χρήσης εντολών goto
242. **ποιά είναι η λειτουργία της εντολής goto**; επιτρέπει την αλλαγή της ακολουθιακής σειράς εκτέλεσης των εντολών, ώστε από οποιαδήποτε εντολή θέλουμε να μεταβαίνουμε σε οποιαδήποτε εντολή θέλουμε. Χάρη σε αυτήν υπάρχουν οι γνωστές δομές επιλογής και επανάληψης, αλλά η ανεξέλεγκτη χρήση της οδηγεί σε ακατανόητο κώδικα, δύσκολο στη διόρθωση.
243. **στην πράξη υλοποιούνται και οι τρεις κλασικές τεχνικές ταυτόχρονα**; ναι, μέσω του τμηματικού προγραμματισμού (έχεις εφαρμόσει την ιεραρχική σχεδίαση για να χωρίσεις σε επιμέρους τμήματα το πρόγραμμα, και στην υλοποίηση του καθενός εφαρμόζεις δομημένο προγραμματισμό, χωρίς goto)
244. **ποιά είναι τα πλεονεκτήματα του δομημένου προγραμματισμού**;
    - εύκολη μετατροπή αλγορίθμου σε πρόγραμμα,
    - δημιουργία απλούστερων προγραμμάτων
    - εύκολη διόρθωση / επέκταση,
    - εύκολη ανάγνωση και κατανόηση από τρίτους,
    - εύκολη ανάλυση του προγράμματος σε τμήματα,
    - παρουσία λιγότερων λαθών κατά τη δημιουργία ενός προγράμματος

Γλώσσες:

245. **ποιές διαφορές διακρίνονται μεταξύ φυσικών και τεχνητών γλωσσών**; φυσικές είναι οι γλώσσες που ομιλούνται για τις καθημερινές επικοινωνιακές ανάγκες, ενώ τεχνητές είναι οι γλώσσες που έχουν κατασκευαστεί για κάποιο συγκεκριμένο επικοινωνιακό σκοπό. Ως εκ τούτου, οι φυσικές γλώσσες είναι πλουσιότερες, ενώ εξελίσσονται και γρηγορότερα από τις τεχνητές
246. **ποιά είναι τα χαρακτηριστικά που διαφοροποιούν κάθε γλώσσα από όλες τις άλλες**; αλφάβητο, λεξιλόγιο, γραμματική, σημασιολογία
247. **τι είναι αλφάβητο**; αλφάβητο είναι το σύνολο των συμβόλων που χρησιμοποιούνται σε μία γλώσσα
248. **τί είναι λέξη**; λέξη είναι κάθε αποδεκτή ακολουθία γραμμάτων του αλφαβήτου
249. **τί είναι λεξιλόγιο**; λεξιλόγιο είναι το σύνολο των λέξεων
250. **τί είναι το τυπικό (τυπολογικό, τυπικοί κανόνες)**; τυπικό είναι το σύνολο των κανόνων που καθορίζουν τους διάφορους τύπους στους οποίους θεωρείται σωστή μία λέξη
251. **τί είναι το συντακτικό**; συντακτικό είναι το σύνολο των κανόνων που καθορίζουν τη σειρά στην οποία πρέπει να τοποθετούνται οι λέξεις,
252. **τί είναι γραμματική**; γραμματική είναι το σύνολο των τυπικών και των συντακτικών κανόνων
253. **τί είναι σημασιολογία**; σημασιολογία είναι το σύνολο των κανόνων που προσδιορίζουν το νόημα μιας λέξης ή μιας φράσης

Προγραμματιστικά περιβάλλοντα:

254. **τι είναι το προγραμματιστικό περιβάλλον**; είναι ένα πρόγραμμα που μας διευκολύνει στη δημιουργία νέων προγραμμάτων χωρίς να χρειάζεται να γνωρίζει κανείς τη γλώσσα του υπολογιστή, αλλά μία γλώσσα προγραμματισμού (γλώσσα υψηλού επιπέδου)
255. **τι περιέχεται στις βιβλιοθήκες**; βιβλιοθήκη ονομάζεται ένα σύνολο υποπρογραμμάτων-εντολών, των οποίων οι λειτουργίες έχουν παρόμοιο αντικείμενο (πχ βιβλιοθήκη με εντολές για γραφικά, βιβλιοθήκη με εντολές για έλεγχο του ποντικιού)
256. **ποιά είναι τα επιμέρους τμήματα σε ένα προγραμματιστικό περιβάλλον**; συντάκτης, διερμηνευτής, μεταγλωττιστής, συνδέτης-φορτωτής
257. **ποιά είναι τα απαραίτητα εργαλεία ενός προγραμματιστικού περιβάλλοντος**; συντάκτης, μεταγλωττιστής, συνδέτης-φορτωτής
258. **σε τι χρησιμεύει ο συντάκτης**; είναι ένας ειδικός επεξεργαστής κειμένου, ο οποίος βοηθάει τον προγραμματιστή να γράψει συντακτικά σωστό πηγαίο κώδικα

259. **πώς ονομάζεται το πρόγραμμα που γράφουμε σε έναν συντάκτη**; πηγαίος κώδικας, πηγαίο πρόγραμμα
260. **πώς λειτουργεί ένας διερμηνευτής**; κάνει βήμα-βήμα τη μετάφραση του πηγαίου κώδικα σε ισοδύναμες εντολές στη γλώσσα του υπολογιστή, ενώ παράλληλα τον εκτελεί και δείχνει τα απότελέσματα από την εκτέλεσή του
261. **ποιά πλεονεκτήματα και ποιά μειονεκτήματα έχει**; πλεονέκτημα είναι ότι χάρη στη βήμα-βήμα εκτέλεση, βοηθάει έναν παρατηρητικό προγραμματιστή να εντοπίσει πιθανά σφάλματα στη λογική με την οποία λειτουργεί το πρόγραμμα (άμεση εκτέλεση+διόρθωση). Μειονέκτημα είναι ότι κάθε φορά που πρέπει να εκτελεστεί το πρόγραμμα, χρειάζεται να ξαναμεταφραστεί από την αρχή (χάσιμο χρόνου), απαιτώντας έτσι την ύπαρξη του προγραμματιστικού περιβάλλοντος και του διερμηνευτή για την εκτέλεση ενός προγράμματος (δεν παρέχει ανεξαρτησία)
262. **πώς λειτουργεί ένας μεταγλωττιστής**; μεταφράζει ολόκληρο τον πηγαίο κώδικα και παράγει το αντικείμενο πρόγραμμα, ένα ανεξάρτητο πρόγραμμα σε γλώσσα μηχανής
263. **ποιά πλεονεκτήματα και ποιά μειονεκτήματα έχει**; πλεονέκτημα είναι ότι η μεταγλώττιση γίνεται μία κι έξω, δε χρειάζεται να ξαναγίνει παρά μόνο αν αλλάξει ο πηγαίος κώδικας. Μειονέκτημα είναι ότι για να μπορεί να μεταγλωττιστεί ο πηγαίος κώδικας, θα πρέπει να μην υπάρχει κανένα συντακτικό σφάλμα.
264. **πώς ονομάζεται το πρόγραμμα που προκύπτει μετά τη σύνδεση-φόρτωση**; είναι το εκτελέσιμο πρόγραμμα, σε γλώσσα μηχανής
265. **ποιά είναι η λειτουργία του συνδέτη-φορτωτή**; ενσωματώνει στο δικό μας κώδικα κώδικα έτοιμο από τις βιβλιοθήκες που τυχόν χρησιμοποιούμε, ενώ προσθέτει και εντολές για τη φόρτωση του προγράμματος στη μνήμη όταν εκτελείται

Υποπρογράμματα:

266. **τι ονομάζεται τμηματικός προγραμματισμός**; είναι η τεχνική ανάπτυξης προγραμμάτων ως σύνθεση επιμέρους μικρότερων προγραμμάτων (υποπρογραμμάτων)
267. **τι ονομάζεται υποπρόγραμμα**; είναι ένα μικρό πρόγραμμα που υλοποιεί μια συγκεκριμένη λειτουργία, αυτόνομο και γραμμένο ξεχωριστά από άλλα προγράμματα
268. **ποιά τρία χαρακτηριστικά πρέπει να έχουν τα υποπρογράμματα**; πρέπει να είναι σύντομα σε έκταση (δηλαδή να υλοποιούν μία λειτουργία, όχι πολλές), να έχουν μία είσοδο και μία έξοδο (δηλαδή όταν κληθούν, εκτελούνται από την πρώτη ως την τελευταία τους εντολή -ή αλλιώς, δε βάζουμε goto στον τμηματικό προγραμματισμό-), και να είναι αυτόνομα (δηλαδή μία αλλαγή που θα γίνει σε ένα υποπρόγραμμα, να μην προκαλεί αλλαγές σε άλλα τμήματα)
269. **ποιά πλεονεκτήματα προκύπτουν από την εφαρμογή του τμηματικού προγραμματισμού**; προκύπτει ευκολία στην ανάπτυξη νέων προγραμμάτων (γιατί ένα τμήμα είναι πάντα πιο εύκολο να αντιμετωπιστεί απ' ότι ένα σύνολο), ευκολία στην κατανόηση (αρκεί το όνομα του υποπρογράμματος να περιγράφει τη λειτουργία του) και τη διόρθωση (πάλι γιατί ένα τμήμα είναι μικρότερο από ένα σύνολο), κερδίζεται χρόνος (γιατί το ίδιο υποπρόγραμμα μπορεί να κληθεί σε πολλά προγράμματα χωρίς να χρειαστεί να ξαναγραφτεί), εμπλουτίζεται η γλώσσα προγραμματισμού (κάθε υποπρόγραμμα αποθηκεύεται σε μία βιβλιοθήκη υπάρχουσα ή καινούργια και μπορεί να καλείται, όπως οι 8 ενσωματωμένες συναρτήσεις)
270. **τι μπορεί να υλοποιεί μία συνάρτηση / τί είναι συνάρτηση**; συνάρτηση είναι το υποπρόγραμμα που υλοποιεί υπολογισμούς ώστε να προκύψει ακριβώς ένα τελικό επιστρεφόμενο αποτέλεσμα (μία συνάρτηση δεν επιτρέπεται να έχει εντολές εισόδου, εντολές εξόδου, ούτε κλήσεις σε διαδικασίες)
271. **ποιές είναι οι οκτώ έτοιμες συναρτήσεις**; A_M(), A_T(), E(), ΗΜ(), ΣΥΝ(), ΕΦ(), T_P(), ΛΟΓ()
272. **τι μπορεί να υλοποιεί μια διαδικασία / τί είναι διαδικασία**; διαδικασία είναι το υποπρόγραμμα που μπορεί να υλοποιεί οποιαδήποτε λειτουργία υλοποιεί κι ένα πρόγραμμα
273. **τι είναι οι παράμετροι**: παράμετροι ονομάζονται οι μεταβλητές που χρησιμοποιούνται για την ανταλλαγή τιμών μεταξύ των διαφόρων τμημάτων στον τμηματικό προγραμματισμό (μπαίνουν εντός παρενθέσεων δίπλα στο όνομα του υποπρογράμματος)
274. **ποιές παράμετροι ονομάζονται εισόδου, και ποιές εξόδου**; παράμετρος εισόδου ονομάζεται εκείνη η οποία για ένα υποπρόγραμμα αποτελεί είσοδο/δεδομένο, ενώ παράμετρος εξόδου ονομάζεται εκείνη η οποία για ένα υποπρόγραμμα αποτελεί έξοδο/αποτέλεσμα
275. **πόσες παραμέτρους εισόδου μπορεί/πρέπει να έχει μια συνάρτηση**; τουλάχιστον μία
276. **πόσες παραμέτρους εξόδου μπορεί/πρέπει να έχει μια συνάρτηση**; καμία, στη συνάρτηση το αποτέλεσμα συμβολίζεται με το όνομα της

277. **ποιά εντολή πρέπει να υπάρχει σε μία συνάρτηση μέσα**: εντολή εκχώρησης του αποτελέσματος στο όνομα της συνάρτησης
278. **ποιές εντολές απαγορεύον σε μία συνάρτηση μέσα**: ΓΡΑΨΕ, ΔΙΑΒΑΣΕ, ΚΑΛΕΣΕ
279. **πόσες παραμέτρους εισόδου μπορεί/πρέπει να έχει μία διαδικασία**; όσες θέλουμε (ακόμα και καμμία)
280. **πόσες παραμέτρους εξόδου μπορεί/πρέπει να έχει μία διαδικασία**; όσες θέλουμε (ακόμα και καμμία)
281. **ποιές παράμετροι ονομάζονται πραγματικές, και ποιές τυπικές (ορίσματα)**; πραγματικές είναι οι παράμετροι που χρησιμοποιεί αυτός που καλεί ένα υποπρόγραμμα, ενώ τυπικές (ή ορίσματα) είναι οι παράμετροι που χρησιμοποιεί αυτός που φτιάχνει ένα υποπρόγραμμα
282. **ποιοί κανόνες πρέπει να τηρούνται κατά την αντιστοίχιση των τυπικών με τις πραγματικές παραμέτρους**; αντιστοιχίζονται μία-μία, ανάλογα με τη σειρά που παρουσιάζονται εντός των παρενθέσεων (και όχι ανάλογα με το όνομά τους), πρέπει το πλήθος των τυπικών να συμφωνεί με το πλήθος των πραγματικών, και πρέπει να συμφωνούν και οι τύποι των αντιστοιχιζόμενων μεταβλητών (ακέραια με ακέραια, πίνακας με πίνακα ίδιου μεγέθους)
283. **ποιά είναι η τυπική μορφή σύνταξης μιας διαδικασίας** :

    διαδικασία *όνομα* (*παράμετροι εισόδου-εξόδου*)
    τμήμα δήλωσης σταθερών
    τμήμα δήλωσης μεταβλητών
    αρχή
    εντολές
    τέλος_διαδικασίας
284. **ποιά είναι η τυπική μορφή σύνταξης μιας συνάρτησης**;

    συνάρτηση *όνομα* (*παράμετροι εισόδου*):*τύπος*
    τμήμα δήλωσης σταθερών
    τμήμα δήλωσης μεταβλητών
    αρχή
    εντολές
    όνομα ← *παράσταση*
    τέλος_συνάρτησης
285. **ποιός είναι ο τρόπος κλήσης/χρήσης μιας συνάρτησης**; με απλή αναφορά του ονόματός της, όπου χρειάζεται να χρησιμοποιηθεί η τιμή της (συνήθως πάντως σε εντολή εκχώρησης τιμής)
286. **ποιός είναι ο τρόπος κλήσης/χρήσης μιας διαδικασίας**; κάλεσε όνομα(παράμετροι)
287. **πώς λειτουργεί η «συνεργασία» μεταξύ των τμημάτων όταν καλούν διαδικασία**; όταν ένα τμήμα καλεί μία διαδικασία, διακόπτεται η εκτέλεση των εντολών του τμήματος, περνάνε οι παράμετροι του τμήματος στις παραμέτρους της διαδικασίας, εκτελείται η διαδικασία, επιστρέφουν οι παράμετροι της καλούμενης διαδικασίας πίσω στις παραμέτρους του τμήματος, και συνεχίζεται η εκτέλεση των εντολών του τμήματος
288. **πώς λειτουργεί η «συνεργασία» μεταξύ των τμημάτων όταν καλούν συνάρτηση**; όταν ένα τμήμα καλεί μία συνάρτηση, διακόπτεται η εκτέλεση των εντολών του τμήματος, περνάνε οι παράμετροι του τμήματος στις παραμέτρους της συνάρτησης, εκτελείται η συνάρτηση, επιστρέφει **μόνο** η τιμή της καλούμενης συνάρτησης πίσω στο τμήμα, και συνεχίζεται η εκτέλεση των εντολών του τμήματος
289. **πως βάζουμε ως παράμετρο έναν ολόκληρο πίνακα**; αναφέροντας στη λίστα των παραμέτρων εντός παρενθέσεων μόνο το όνομά του, χωρίς αγκύλες, χωρίς δείκτη, ενώ στο τμήμα δηλώσεων δείχνουμε ότι πρόκειται για πίνακα πχ: διαδικασία τάδε (Α,Β,Γ)
    μεταβλητές
    ακέραιες: Α[100], Β, Γ[40]
290. **πως βάζουμε ως παράμετρο ένα συγκεκριμένο κελί ενός πίνακα**; αντιστοιχίζοντας σε μία μεταβλητή του καλούμενου υποπρογράμματος, ένα συγκεκριμένο κελί πχ: για τη συνάρτηση Α_Τ(x), αντιστοιχίζουμε το Α_Τ(Β[i])
291. **μπορεί ένα υποπρόγραμμα εμφωλευμένα να καλεί ένα άλλο**; μια διαδικασία μπορεί να περιέχει εντολές που καλούν οποιοδήποτε υποπρόγραμμα, μια συνάρτηση μπορεί να περιέχει εντολές που καλούν μόνο συναρτήσεις
292. **τι είναι η στοίβα χρόνου εκτέλεσης**; είναι μία ειδική στοίβα στην οποία ωθείται μια **διεύθυνση επιστροφής**, κάθε φορά που ένα τμήμα καλεί ένα άλλο, ενώ κάθε φορά που ένα υποπρόγραμμα ολοκληρώνεται, απωθείται μια διεύθυνση επιστροφής από τη στοίβα. Έτσι, επιτυγχάνεται η σωστή σειρά εκτέλεσης των επιμέρους τμημάτων.
293. **τι σημαίνει εμβέλεια**: η εμβέλεια/ισχύς καθορίζει το πού μπορεί να χρησιμοποιηθεί μία μεταβλητή (ή μία σταθερά) σε σχέση με το πού έχει δηλωθεί

294. **ποιά είναι τα είδη εμβέλειας που υπάρχουν**; τοπική (περιορισμένη), καθολική (απεριόριστη), μικτή (μερικώς περιορισμένη)
295. **τι σημαίνει τοπική, τι σημαίνει καθολική, και τι σημαίνει μικτή εμβέλεια**;
    - **τοπική** εμβέλεια είναι όταν οι μεταβλητές μπορεί να χρησιμοποιηθούν μόνο μέσα στο τμήμα στο οποίο δηλώθηκαν
    - **καθολική** εμβέλεια είναι όταν οι μεταβλητές μπορούν να χρησιμοποιηθούν σε οποιοδήποτε τμήμα, άσχετα από το πού δηλώθηκαν
    - **μικτή** εμβέλεια είναι όταν υπάρχουν και μεταβλητές τοπικής και μεταβλητές καθολικής εμβέλειας
296. **ποιό είδος εμβέλειας τηρείται στη ΓΛΩΣΣΑ που μαθαίνουμε**: χρησιμοποιούμε μόνο μεταβλητές τοπικής εμβέλειας (γι' αυτό μετά χρειαζόμαστε τις παραμέτρους, για να αντιστοιχίσουμε τις μεταβλητές ενός τμήματος με εκείνες ενός άλλου)
297. **γιατί δε χρησιμοποιούμε καθολική εμβέλεια;** Η ύπαρξη μεταβλητών καθολικής εμβέλειας οδηγεί σε απώλεια της αυτονομίας που θέλουμε να έχουν τα υποπρογράμματα, γιατί η αλλαγή μιας καθολικής μεταβλητής από ένα τμήμα μπορεί να δημιουργήσει απρόβλεπτες αλλαγές σε ένα άλλο τμήμα. Ένας έμπειρος προγραμματιστής μπορεί να επωφεληθεί από τη χρήση καθολικών μεταβλητών, αλλά ένας αρχάριος περισσότερο μπερδεύεται

Εκσφαλμάτωση:

298. **Εκσφαλμάτωση**: Η διαδικασία ελέγχου, εντοπισμού και διόρθωσης των σφαλμάτων ενός προγράμματος.
299. **ποιά είδη σφαλμάτων αναφέρονται**: σφάλματα συντακτικά, σφάλματα λογικής, και σφάλματα χρόνου εκτέλεσης
300. **ποιά είναι τα λάθη που όταν παρουσιάζονται τερματίζεται αντικανονικά η εκτέλεση ενός προγράμματος:** αυτά είναι τα σφάλματα χρόνου εκτέλεσης
301. **συνηθισμένα σφάλματα χρόνου εκτέλεσης**: διαιρέσαμε με 0, επιχειρήσαμε να χρησιμοποιήσουμε την τιμή μιας μεταβλητής χωρίς όμως να έχει οριστεί νωρίτερα, ζητήσαμε τον υπολογισμό της τετραγωνικής ρίζας ενός αρνητικού αριθμού, περιμέναμε να διαβαστεί μία ακέραια μεταβλητή ενώ ο χρήστης πληκτρολόγησε μια πραγματική τιμή, χρησιμοποιήσαμε δείκτη εκτός των ορίων ενός πίνακα.
302. **ποιά είναι τα λάθη που εντοπίζονται από το προγραμματιστικό περιβάλλον κατά τη μετάφραση του προγράμματος**; είναι τα συντακτικά σφάλματα, και σχετίζονται με την παραβίαση των διαφόρων κανόνων που κατά καιρούς αναφέρουμε για τη σύνταξη ενός προγράμματος, ενός υποπρογράμματος, μιας εντολής, μιας δομής. Ένα πρόγραμμα με συντακτικά σφάλματα δεν εκτελείται, γιατί το προγραμματιστικό περιβάλλον δεν μπορεί να το μεταφράσει
303. **συνηθισμένα συντακτικά λάθη**: δεν δήλωσες μια μεταβλητή ή τη δήλωσες σε λάθος κατηγορία, το όνομα της μεταβλητής είναι λάθος, ξέχασες να κλείσεις κάποια παρένθεση, ξέχασες το ΤΕΛΟΣ_ΑΝ, ξέχασες το ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ, έγραψες μια εντολή λάθος
304. **ποιά είναι τα λάθη που εντοπίζονται από τον άνθρωπο κατά την εκτέλεση του προγράμματος**; είναι τα σφάλματα λογικής, που σχετίζονται με τη λανθασμένη υλοποίηση της ζητούμενης λειτουργίας. Ένα πρόγραμμα με σφάλματα λογικής εκτελείται κανονικά, αλλά βγάζει λάθος (ή καθόλου) αποτελέσματα.
305. **Συνηθισμένα λογικά λάθη**: λάθος αρχική τιμή σε κάποια μεταβλητή, παράλειψη εντολών, παράλειψη εξέτασης όλων των απαιτούμενων περιπτώσεων, λανθασμένη διατύπωση αριθμητικών παραστάσεων, λανθασμένη διατύπωση συνθηκών, δομές επανάληψης που δεν τερματίζονται ποτέ, λάθος χρήση ΜΕΧΡΙΣ_ΟΤΟΥ έναντι ΟΣΟ, λάθος σειρά δεικτών σε πολυδιάστατο πίνακα, πέρασμα λάθος παραμέτρων σε υποπρόγραμμα, ανεπιθύμητη αλλαγή παραμέτρων από κάποια διαδικασία...
306. **πώς εντοπίζονται τα σφάλματα λογικής**; κάνουμε δοκιμαστικές εκτελέσεις του προγράμματός μας για να ελέγξουμε αν για συγκεκριμένες τιμές εισόδου, το πρόγραμμά μας εξάγει σωστά αποτελέσματα
307. **τί είναι ένα σενάριο ελέγχου**; η περιγραφή κάθε τέτοιας δοκιμαστικής εκτέλεσης με συγκεκριμένα δεδομένα εισόδου (ολόκληρου του προγράμματος ή τμήματος του προγράμματος) και τα αναμενόμενα αποτελέσματα. Όσο περισσότερες παραμέτρους έχει ένα πρόβλημα, τόσο πιο σύνθετα σενάρια πρέπει να δημιουργήσουμε.

308. **Τι ονομάζουμε Μαύρο Κουτί**: Μια τεχνική εκσφαλμάτωσης, στην οποία τα δεδομένα εισόδου στα σενάρια ελέγχου προκύπτουν από τις προδιαγραφές του προβλήματος/προγράμματος, αγνοώντας εντελώς τον κώδικα, θεωρώντας ότι ο κώδικας είναι σε ένα Μάυρο κουτί.
309. **Ποιά είναι τα βήματα στην εκσφαλμάτωση**: δημιουργούμε πρώτα ισοδύναμα διαστήματα, επιλέγουμε δοκιμαστικές τιμές, διενεργούμε τους ελέγχους
310. **Τι είναι τα ισοδύναμα διαστήματα**: είναι διαστήματα τιμών για κάποια μεταβλητή του προβλήματος, στα οποία αναμένουμε η συμπεριφορά του αλγορίθμου μας να είναι ίδια
311. **Πώς επιλέγουμε δοκιμαστικές τιμές**: Τυχαία επιλέγουμε κάποια τιμή από κάθε ισοδύναμο διάστημα, αλλά και **κάθε ακραία τιμή**, καθώς έχει παρατηρηθεί ότι πολλά σφάλματα παρουσιάζονται στις ακραίες τιμές των διαστημάτων. Επιλέγουμε να κάνουμε δοκιμές και για **άκυρες** τιμές, καθώς δεν είναι βέβαιο ότι ένας αλγοριθμος θα τροφοδοτείται σίγουρα μόνο με έγκυρες τιμές.
312. **Και σε μεγάλα προγράμματα πώς λειτουργεί**; Τα μεγάλα προγράμματα είναι πάντα χωρισμένα σε τμήματα, οπότε εφαρμόζουμε πρώτα τη διαδικασία του ελέγχου στα τμήματα, και εφ' όσον αυτά λειτουργούν σωστά, ελέγχουμε και το συνολικό πρόγραμμα.

Αντικειμενοστραφής Σχεδίαση / Προγραμματισμός:

313. **τι είναι**; μεθοδολογία ανάπτυξης εφαρμογών που στηρίζεται η οποία στηρίζεται σε αυτόνομες προγραμματιστικές οντότητες με δική τους ταυτότητα και συμπεριφορά (τα αντικείμενα).
314. **τι είναι ένα αντικείμενο**; Ένα αυτόνομο τμήμα κώδικα (που αντιστοιχεί σε κάποια φυσική οντότητα ή έννοια) μέσα στο οποίο περιλαμβάνονται τόσο τα **δεδομένα** (αλλιώς **ιδιότητες**) που χρειάζονται για να το περιγράψουν, όσο και οι **μέθοδοι** (ενέργειες) που χρησιμεύουν για την υλοποίηση της συμπεριφοράς του και την επεξεργασία των δεδομένων του.
315. **τι είναι η κλάση**: είναι ο γενικός τύπος ενός αντικειμένου, που καθορίζει τις αρχικές ιδιότητες και τη συμπεριφορά κάθε αντικειμένου που προέρχεται από αυτή. Μια κλάση αποτελεί ένα αφαιρετικό (abstract) στοιχείο (τύπο) και μπορεί να παράγει ένα απεριόριστο πλήθος δομικά ίδιων αντικειμένων (όπως λέμε ότι οι μεταβλητές ανήκουν σε κάποιο τύπο μεταβλητής).
316. **ποιά είναι τα πλεονεκτήματά της αντικειμενοστρεφούς σχεδίασης**: περιγράφει τα προβλήματα με έναν πιο φυσικό και πιο κοντά στην ανθρώπινη αντίληψη του κόσμου, και οδηγεί σε κώδικα που είναι επαναχρησιμοποιήσιμος και απλουστεύει την υλοποίηση σύνθετων προβλημάτων. Ταυτόχρονα, βασίζεται και στην ιεραρχική σχεδίαση και τον δομημένο προγραμματισμό, ενώ εφαρμόζει και τον τμηματικό προγραμματισμό.
317. **ποιά είναι τα χαρακτηριστικά που συναντάμε στον αντικειμενοστραφή προγραμματισμό**; ενθυλάκωση, αφαιρετικότητα, πολυμορφισμός, κληρονομικότητα
318. **τι ονομάζουμε ενθυλάκωση**; Η δυνατότητα ενός αντικειμένου να συνδυάζει εσωτερικά τα δεδομένα και τις μεθόδους χειρισμού αυτών των δεδομένων
319. **τι είναι η αφαιρετικότητα**: Η δυνατότητα δημιουργίας/περιγραφής πολλών διαφορετικών αντικειμένων (στιγμιοτύπων) με βάση την ίδια κλάση, αφού μια κλάση χρησιμεύει για τη γενική (αφηρημένη) περιγραφή όλων.
320. **τι είναι ο πολυμορφισμός:** το ότι μια λειτουργία μπορεί να υλοποιείται με πολλούς διαφορετικούς τρόπους σε πολλές κλάσεις, ενώ έχει το ίδιο όνομα σε όλες αυτές(π.χ. τα τρίγωνα και τα τετράγωνα έχουν μία μέθοδο υπολογισμού του εμβαδού τους, η μέθοδος μπορεί να ονομάζεται το ίδιο και στα δύο σχήματα, αλλά η υλοποίηση είναι αλλιώτικη στο καθένα)
321. **τι είναι η κληρονομικότητα**: Η δυνατότητα δημιουργίας ιεραρχιών αντικειμένων και της εξέλιξης υπάρχοντων αντικειμένων. Μια κλάση μπορεί να περιγραφεί γενικά και στη συνέχεια μέσω αυτής της κλάσης να οριστούν υποκλάσεις αντικειμένων. Η κλάση απόγονος (υποκλάση) κληρονομεί και μπορεί να χρησιμοποιήσει όλα τα δεδομένα (ιδιότητες) και τις μεθόδους που περιέχει η κλάση πρόγονος (υπερκλάση).
322. **τι είναι η υποκλάση (κλάση απόγονος)**; μία κλάση η οποία έχει κληρονομήσει τα δεδομένα και τις μεθόδους μιας άλλης, ενώ μπορεί να έχει κι άλλα δικά της
323. **τι είναι η υπερκλάση (κλάση πρόγονος)**; μία κλάση η οποία κληροδοτεί τα δεδομένα και τις μεθόδους της σε μια άλλη κλάση
324. **τι λέει ο κανόνας «α is_a β»**: χρησιμεύει για να αναγνωρίσουμε κατά πόσο μία κλάση είναι υποκλάση μίας άλλης. Εάν η πρόταση «α is_a β» ευσταθεί, σημαίνει ότι η κλάση α είναι υποκλάση της κλάσης β (πχ «ο σκύλος είναι ένα ζώο», «το γραφείο είναι ένα έπιπλο», αλλά «το μολύβι είναι ένα όχημα»)
325. **Ποιά είναι η μεθοδολογία ανάπτυξης αντικειμενοστραφούς εφαρμογής**:
    1. εντοπίζουμε τα αντικείμενα που συμμετέχουν στο σενάριο του προβλήματος,

2. καταγράφουμε τις ιδιότητες κάθε αντικειμένου
3. περιγράφουμε τις υπηρεσίες που προσφέρει ή τις ενέργειες που υλοποιεί κάθε αντικείμενο (μέθοδοι) προς αξιοποίηση από άλλες, ώστε να αναπτυχθούν οι απαραίτητες συνεργασίες μεταξύ των αντικειμένων για την επίλυση του προβλήματος.

326. **Πως δομείται ένα αντικειμενοστραφές πρόγραμμα:** ως ένα δίκτυο συνεργαζόμενων οντοτήτων που είναι τα αντικείμενα. Κάθε αντικείμενο έχει ένα συγκεκριμένο ρόλο και παρέχει μια υπηρεσία ή εκτελεί μια ενέργεια (μέθοδο) που χρησιμοποιείται από άλλα μέλη του δικτύου, δηλαδή από άλλα αντικείμενα.

327. **Πώς αναπαρίστανται διαγραμματικά τα αντικείμενα**: Σε ένα παραλληλόγραμμο με το όνομα της κλάσης, μετά τις ιδιότητες της κλάσης και τέλος τις μεθόδους της κλάσης. Οι μέθοδοι κατά σύμβαση παρουσιάζονται σα να πρόκειται για υποπρογράμματα με δύο παρενθέσεις δίπλα στο όνομά τους (βλέπε ακόλουθο σχήμα).

328. **Τι δείχνει ένα διάγραμμα κλάσεων**: παρουσιάζει τις διάφορες κλάσεις του προβλήματος, αλλά και τις μεταξύ τους συσχετίσεις, με κάποιο όνομα/περιγραφή για την κάθε συσχέτιση. Επίσης μπορεί να δείχνει τις σχέσεις κληρονομικότητας μεταξύ των διαφόρων κλάσεων. Ένα βελάκι από την υποκλάση δείχνει προς την υπερκλάση (βλέπε σχήμα). Μπορεί σε κάποιο διάγραμμα να παρουσιάσουμε και συγκεκριμένα αντικείμενα της κάθε κλάσης, όχι μόνο γενικά τις κλάσεις.