Κεφάλαιο 1ο Προβλήματα

1. Τι είναι **πρόβλημα**:
   Με τον όρο πρόβλημα εννοείται μία κατάσταση η οποία χρίζει αντιμετώπισης , απαιτεί λύση, η δε λύση της δεν είναι γνωστή ούτε προφανής.

2. **Κατανόηση προβλήματος**, Εξαρτάται από δύο παράγοντες:
   1. σωστή διατύπωση από την μεριά του δημιουργού.
   2. Σωστή ερμηνεία από της μεριά εκείνου που καλείται να το αντιμετωπίσει.
3. Τι είναι **δομή προβλήματος**:
   Με τον όρο δομή προβλήματος αναφερόμαστε στα συστατικά του μέρη, δηλαδή στα επιμέρους τμήματα που το αποτελούν καθώς και στον τρόπο που αυτά συνδέονται μεταξύ τους.

4. **Ανάλυση προβλήματος**
   Το αρχικό πρόβλημα αναλύεται σε άλλα απλούστερα υποπροβλήματα. Με την σειρά τους τα νέα προβλήματα αναλύονται σε άλλα ακόμη πιο απλά.

5. Ποια τα **στάδια αντιμετώπισης του προβλήματος**:
   1. **Κατανόηση**, όπου απαιτείται η σωστή και πλήρης αποσαφήνιση των δεδομένων και των ζητουμένων.
   2. **Ανάλυση** όπου το αρχικό πρόβλημα διασπάται σε άλλα επιμέρους απλούστερα προβλήματα.
   3. **Επίλυση** όπου υλοποιείται η λύση του προβλήματος μέσω της λύσης των επιμέρους υποπροβλημάτων.

6. Κατηγορίες προβλημάτων σύμφωνα με την **δυνατότητα επίλυσης**:
   1. **Επιλύσιμα** είναι εκείνα τα προβλήματα για τα οποία η λύση τους είναι ήδη γνωστή
   2. **Ανοικτά**: είναι εκείνα τα προβλήματα για τα οποία η λύση τους δεν έχει ακόμα βρεθεί, αλλά παράλληλα δεν έχει αποδειχθεί ότι δεν λύνονται
   3. **Άλυτα** χαρακτηρίζονται εκείνα τα προβλήματα για τα οποία έχουμε φτάσει στην παραδοχή ότι δεν λύνονται.

7. Κατηγορίες προβλημάτων σύμφωνα με το **βαθμό δόμησης** των λύσεων τους:
   1. **Δομημένα** χαρακτηρίζονται εκείνα τα προβλήματα των οποίων η λύση προέρχεται από μία αυτοματοποιημένη διαδικασία
   2. **Ημιδομημένα** ονομάζονται εκείνα τα προβλήματα των οποίων η λύση επιδιώκεται στα πλαίσια ενός εύρους πιθανών λύσεων, αφήνοντας των ανθρώπινο παράγοντα περιθώρια επιλογής της.
   3. **Αδόμητα** χαρακτηρίζονται εκείνα τα προβλήματα των οποίων οι λύσεις δεν μπορούν να δομηθούν ή δεν έχει διερευνηθεί σε βάθος η δυνατότητα δόμησής τους.

8. Κατηγορίες προβλημάτων με κριτήριο το **είδος επίλυσης**:
   1. **Απόφασης**, όπου η απόφαση που πρόκειται να ληφθεί σαν λύση του προβλήματος, απαντά σε ένα ερώτημα με πιθανή απάντηση ναι ή όχι
   2. **Υπολογιστικά** όπου το πρόβλημα που τίθεται απαιτεί διενέργεια υπολογισμών για να δοθεί μια απάντηση στο πρόβλημα
   3. **Βελτιστοποίησης**, όπου το πρόβλημα επιζητά το βέλτιστο αποτέλεσμα για τα συγκεκριμένα δεδομένα που διαθέτει.

9. Ποιοι είναι οι **λόγοι** που **αναθέτουμε ένα πρόβλημα στον υπολογιστή**
1. Η πολυπλοκότητα των υπολογισμών
2. Η επαναληπτικότητα των διαδικασιών
3. Η ταχύτητα εκτέλεσης των πράξεων
4. Το μεγάλο πλήθος δεδομένων

10. Ποιες είναι οι **βασικές λειτουργίες** που μπορεί να εκτελέσει ο **υπολογιστής**:
1. **Πρόσθεση** η οποία αποτελεί την βασική αριθμητική πράξη, δεδομένου ότι και οι υπόλοιπες πράξεις μπορούν να αντιμετωπισθούν ως πρόσθεση
2. **Σύγκριση** η οποία συνιστά την βασική λειτουργία για τη επιτέλεση όλων των λογικών πράξεων
3. **Μεταφορά δεδομένων** λειτουργεία που προηγείται και έπεται της επεξεργασίας δεδομένων

Κεφάλαιο 2ο Αλγόριθμοι

1. Αλγόριθμος Ορισμός: Είναι μία πεπερασμένη σειρά ενεργειών, αυστηρά καθορισμένων και εκτελέσιμων σε πεπερασμένο χρόνο, που στοχεύουν στην επίλυση ενός προβλήματος. Η σειρά – αλληλουχία των ενεργειών δεν είναι μοναδική.
Η έννοια του αλγορίθμου δεν συνδέεται αποκλειστικά με έννοιες της πληροφορικής.

2. Απαραίτητα **κριτήρια πλήρη αλγόριθμου** (πρέπει να τα ικανοποιεί κάθε αλγόριθμος):
1. **Είσοδος:** Καμία , μία ή περισσότερες τιμές δεδομένων πρέπει να δίνονται ως είσοδος. 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 την βοήθεια άλλων απλών εντολών.
2. **Έξοδος:** Ο αλγόριθμος πρέπει να δημιουργεί τουλάχιστον ένα αποτέλεσμα προς τον χρήστη ή προς ένα άλλο αλγόριθμο.
3. **Καθοριστικότητα**: Κάθε εντολή πρέπει να καθορίζεται χωρίς αμφιβολία για τον τρόπο εκτέλεσής της. Πχ μία εντολή ζ □ x / ψ πρέπει να λαμβάνει υπ' όψιν της το γεγονός ότι μπορεί το ψ να είναι μηδέν.
4. **Περατότητα:** Ο αλγόριθμος πρέπει να τελειώνει μετά από πεπερασμένα βήματα. Αν δεν τελειώνει μετά από ένα συγκεκριμένο αριθμό βημάτων δεν είναι αλγόριθμος αλλά υπολογιστική διαδικασία.
5. **Αποτελεσματικότητα:** Κάθε εντολή ενός αλγορίθμου πρέπει να είναι απλή και εκτελέσιμη (δεν αρκεί να έχει οριστεί).

3. Υπό ποια πρίσματα η Πληροφορική επιστήμη μελετά τους αλγορίθμους;
1. ***Υλικού (hardware) :*** Η ταχύτητα εκτέλεσης ενός αλγορίθμου επηρεάζεται από τις διάφορες τεχνολογίες υλικού.
2. ***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.
3. ***Θεωρητική (theoretical) :*** Υπάρχει πράγματι ή όχι κάποιος αποδοτικός αλγόριθμος για την επίλυση ενός προβλήματος; Η απάντηση απαιτεί μεγάλη θεωρητική κατάρτιση.
4. ***Αναλυτική (analytical)*** : Μελετώνται οι υπολογιστικοί πόροι (computer resources) που απαιτούνται από έναν αλγόριθμο,

4. Περιγραφή και αναπαράσταση αλγορίθμων

1. **Ελεύθερο κείμενο** (Free text): Αποτελεί τον πιο ανεπεξέργαστο και αδόμητο τρόπο παρουσίασης. Υπάρχει κίνδυνος να οδηγήσει σε μη εκτελέσιμη παρουσίαση, παραβιάζοντας το τελευταίο χαρακτηριστικό των αλγορίθμων το κριτήριο της αποτελεσματικότητας.
2. **Διαγραμματικές τεχνικές** (diagramming techniques): Αποτελούν ένα γραφικό τρόπο αναπαράστασης. Μια τεχνική (από τις πιο παλιές και γνωστές) είναι το διάγραμμα ροής (Flow Chart). Η χρήση διαγρ. τεχνικών δεν είναι η καλύτερη λύση, γι' αυτό εμφανίζονται όλο και σπανιότερα στην βιβλιογραφία και στην πράξη.
3. **Φυσική γλώσσα κατά βήματα** (natural language): Μοιάζει με το Ελεύθερο κείμενο απλά είναι κατά βήματα. Κίνδυνος να παραβιαστεί το κριτήριο της καθοριστικότητας.
4. **Κωδικοποίηση** (coding): Δηλαδή με ένα πρόγραμμα γραμμένο είτε με μία ψευδογλώσσα είτε σε κάποιο προγραμματιστικό περιβάλλον που όταν εκτελεσθεί θα δώσει τα ίδια αποτελέσματα με τον αλγόριθμο.

5. Διάγραμμα ροής

Είναι στην ουσία μία διαγραμματική τεχνική {βλέπε παραπάνω}. Αποτελείται από ένα σύνολο γεωμετρικών σχημάτων, όπου το καθένα δηλώνει μία συγκεκριμένη ενέργεια ή λειτουργία. Τα σχήματα ενώνονται μεταξύ τους με βέλη που δηλώνουν την σειρά εκτέλεσης των ενεργειών αυτών. Τα κυριότερα σχήματα είναι τα εξής:

1. **Έλλειψη**: Η αρχή και το τέλος του αλγορίθμου.
2. **Ρόμβος**: Μία συνθήκη με δύο εξόδους ανάλογα με την τιμή της. {αληθής - ψευδής}
3. **Ορθογώνιο**: Η εκτέλεση μίας ή περισσοτέρων πράξεων.
4. **Πλάγιο παρ/μο**: Η είσοδος και η έξοδος στοιχείων του αλγορίθμου.

6. Δομή ακολουθίας

Ονομάζεται και σειριακή ή ακολουθιακή δομή. Αποτελείται από ένα σύνολο εντολών που τοποθετούνται η μία κάτω από την άλλη. Χρησιμοποιείται (από μόνη της) για την επίλυση **πολύ απλών προβλημάτων** όπου η σειρά εκτέλεσης ενός συνόλου ενεργειών είναι δεδομένη. Χρησιμοποιείται ευρύτατα σε συνδυασμό με άλλες δομές (επιλογής, επανάληψης).

7. Δομή επιλογής

Αποτελείται από ένα σύνολο εντολών που **εκτελούνται κατά περίπτωση**. Η διαδικασία της επιλογής περιλαμβάνει τον έλεγχο κάποιας συνθήκης με δύο δυνατές τιμές (αληθής, ψευδής) και στη συνέχεια την απόφαση εκτέλεσης κάποιας εντολής ανάλογα με τη συνθήκη.

8. Δομή επανάληψης

Η δομή επανάληψης εφαρμόζεται στις περιπτώσεις όπου **μία ακολουθία εντολών πρέπει να γίνει περισσότερες από μία φορές**. Εφαρμόζεται σε ένα σύνολο περιπτώσεων που έχουν κάτι κοινό. Οι επαναληπτικές διαδικασίες έχουν τρεις μορφές και συχνά εμπεριέχουν συνθήκες επιλογών.

9. Ολίσθηση

ολίσθηση προς τα αριστερά ισοδυναμεί με διπλασιασμό του αριθμού.
ολίσθηση προς τα δεξιά ισοδυναμεί με υποδιπλασιασμό του αριθμού (div2).

10. Πολλαπλασιασμός αλά ρωσικά

Η πράξη του πολλαπλασιασμού δύο αριθμών δεν εκτελείται από τον υπολογιστή με τον τρόπο που την εκτελούμε εμείς. Έστω ότι θέλω να πολλαπλασιάσω τους αριθμούς 45 και 19 Οι

αριθμοί που πρέπει να πολλαπλασιαστούν γράφονται δίπλα δίπλα και ο πρώτος διπλασιάζεται ενώ ο δεύτερος υποδιπλασιάζεται. Η διαδικασία αυτή επαναλαμβάνεται ώσπου ο δεύτερος να γίνει μονάδα :

45	**19**	*45*
90	**9**	*90*
180	4	
360	2	
720	**1**	*720*

Τελικά το ζητούμενο γινόμενο ισούται με το άθροισμα των στοιχείων της πρώτης στήλης όπου αντίστοιχα στην δεύτερη στήλη υπάρχει περιττός αριθμός. Στο παράδειγμα τα στοιχεία αυτά παρουσιάζονται στην τρίτη στήλη. Δηλαδή είναι 45+90+720=855 {βλέπε σχολικό για τον αντίστοιχο αλγόριθμο}

11. Γιατί ο **υπολογιστής εκτελεί τον πολλαπλασιασμό αλά ρωσικά**;
Υλοποιείται πιο απλά και γρήγορα με την βοήθεια κυκλωμάτων. Σε αντίθεση με την γνωστή μας διαδικασία πολλαπλασιασμού απαιτεί μόνο διπλασιασμούς και υποδιπλασιασμούς. Σε επίπεδο λοιπόν κυκλωμάτων ο διπλασιασμός μπορεί να γίνει με μία εντολή ολίσθησης προς τα αριστερά ενώ ο υποδιπλασιασμός με ολίσθηση προς τα δεξιά.

Κεφάλαιο 3ο Δομές Δεδομένων

1. Δομή δεδομένων
Είναι ένα σύνολο αποθηκευμένων δεδομένων που υφίστανται επεξεργασία από ένα σύνολο λειτουργιών. Κάθε μορφή δομής δεδομένων αποτελείται από ένα σύνολο κόμβων (nodes). *{ένας πίνακας 100 θέσεων έχει 100 κόμβους }*

2. Βασικές λειτουργίες (πράξεις) των δομών δεδομένων

1. **Προσπέλαση:** πρόσβαση σε ένα κόμβο με σκοπό να εξεταστεί ή να αλλάξει το περιεχόμενό του.
2. **Εισαγωγή:** προσθήκη νέων κόμβων σε μία δομή.
3. **Διαγραφή:** (αντίθετο της εισαγωγής), δηλαδή αφαίρεση ενός κόμβου από μία δομή.
4. **Αναζήτηση:** γίνεται προσπέλαση των κόμβων μίας δομής προκειμένου να εντοπιστούν ένας ή περισσότεροι που έχουν μία δεδομένη ιδιότητα.
5. **Ταξινόμηση:** όπου οι κόμβοι διατάσσονται κατά αύξουσα ή φθίνουσα σειρά.
6. **Αντιγραφή:** όλοι ή μερικοί από τους κόμβους μίας δομής αντιγράφονται σε μία άλλη.
7. **Συγχώνευση:** δύο ή περισσότερες δομές ενώνονται σε μία δομή.
8. **Διαχωρισμός:** (αντίστροφα από την συγχώνευση)

Επειδή οι πίνακες είναι στατικές δομές δεδομένων και το μέγεθος των κόμβων τους δεν μπορεί να αλλάξει κατα την διάρκεια του προγράμματος, οι λειτουργίες της διαγραφής και εισαγωγής κόμβων δεν έχουν νόημα.

3. Αλγόριθμοι + Δομές δεδομένων = Προγράμματα

4. Κατηγορίες δομών δεδομένων

Οι δομές διακρίνονται σε δύο μεγάλες κατηγορίες : Δυναμικές και στατικές

1. **Δυναμικές δομές:**
   - i. Οι δομές αυτές **δεν έχουν σταθερό μέγεθος** αλλά ο αριθμός των κόμβων τους μεγαλώνει καθώς εισάγονται νέα δεδομένα , και μικραίνει καθώς διαγράφονται.
   - ii. **Δεν αποθηκεύονται σε συνεχόμενες θέσεις μνήμης** αλλά στηρίζονται στην τεχνική της **δυναμικής παραχώρησης μνήμης** (DMA : Dynamic Memory Allocation).

Όλες οι σύγχρονες γλώσσες προγραμματισμού τις υποστηρίζουν.

2. **Στατικές δομές (πίνακες, ουρά στοίβα):**
   - i. **Σταθερό μέγεθος**. Το ακριβές μέγεθός τους καθορίζεται κατά την στιγμή του προγραμματισμού τους και όχι κατά την στιγμή της εκτέλεσης του προγράμματος.
   - ii. Οι κόμβοι τους αποθηκεύονται σε **συνεχόμενες θέσεις μνήμης**.

Στην πράξη οι στατικές δομές υλοποιούνται με πίνακες .

5. Αναζήτηση

{ Υπάρχουν αρκετοί αλγόριθμοι (μέθοδοι) αναζήτησης σε πίνακα. Η επιλογή του πιο κατάλληλου εξαρτάται από δύο παράγοντες:
Αν ο πίνακας είναι ταξινομημένος ή όχι.
Αν όλα τα στοιχεία του πίνακα είναι διαφορετικά μεταξύ τους ή όχι.
Τα στοιχεία του πίνακα μπορεί να είναι οποιουδήποτε τύπου: αριθμητικά (ακέραιοι-πραγματικοί), αλφαριθμητικά (χαρακτήρες) ακόμη και λογικά.

6. Σειριακή (sequential) – γραμμική (linear) αναζήτηση

Είναι η πιο απλή αλλά και η πιο αναποτελεσματική μέθοδος αναζήτησης σε πίνακα.

7. Πότε δικαιολογείται η χρήση της σειριακής αναζήτησης:

1. Όταν ο πίνακας είναι μη ταξινομημένος,
2. Όταν ο πίνακας είναι μικρού μεγέθους (ν<=20),
3. Όταν η αναζήτηση γίνεται σπάνια.

8. Ταξινόμηση

Δοθέντων των στοιχείων $α_1$, $α_2$, … $α_n$ η ταξινόμηση ορίζεται ως μετάθεση της θέσης των στοιχείων ώστε να τοποθετηθούν σε μια σειρά $α_{κ1}$, $α_{κ2}$, … $α_{κn}$ έτσι ώστε: Αν δοθεί συνάρτηση διάταξης (ordering function) F να ισχύει : $F(α_{κ1}) \le F(α_{κ2}) \le … \le F(α_{κn})$.

9. Ταξινόμηση ευθείας ανταλλαγής-φυσαλίδας

Η μέθοδος της φυσαλίδα βασίζεται στην σύγκριση και ανταλλαγή ζευγών γειτονικών στοιχείων μέχρις ότου διαταχθούν όλα τα στοιχεία.
Η ταξινόμηση φυσαλίδας είναι η πιο απλή και πιο αργή μέθοδος ταξινόμησης.
Θυμάμαι ότι πρώτα ταξινομώ και μετά αναζητώ. Φαντάσου να ψάχνουμε μια λέξη σε αταξινόμητο λεξικό. Η αναζήτηση έπεται της ταξινόμησης δηλαδή.

10. Δομές Δεδομένων Δευτερεύουσας Μνήμης

Σε μεγάλες πρακτικές εμπορικές/επιστημονικές εφαρμογές, το **μέγεθος της κύριας μνήμης δεν επαρκεί** για την αποθήκευση των δεδομένων. Στην περίπτωση αυτή χρησιμοποιούνται ειδικές δομές για την αποθήκευση των δεδομένων στη δευτερεύουσα

μνήμη, δηλαδή κυρίως στο **μαγνητικό δίσκο**. Οι ειδικές αυτές δομές ονομάζονται αρχεία (files).

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

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.

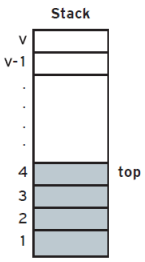
11. Τι ονομάζεται στοίβα;

Στοίβα (stack), ονομάζεται μια **δομή δεδομένων** το σύνολο των στοιχείων της οποίας είναι διατεταγμένο με τέτοιο τρόπο, ώστε τα **στοιχεία που βρίσκονται στην κορυφή** της στοίβας λαμβάνονται **πρώτα**, ενώ αυτά που βρίσκονται στο **βάθος** της στοίβας λαμβάνονται **τελευταία**. Η παραπάνω μέθοδος ονομάζεται Τελευταίο Μέσα, Πρώτο Έξω ή LIFO (=**Last In First Out**).

12. Ποιες είναι οι κύριες λειτουργίες σε μια στοίβα;

1. Η **ώθηση** (**push**) στοιχείου στην κορυφή της στοίβας. Στη διαδικασία της ώθησης ελέγχουμε αν η στοίβα είναι γεμάτη. Στην περίπτωση που προσπαθήσουμε να «προσθέσουμε» ένα στοιχείο σε μια ήδη γεμάτη στοίβα, έχουμε υπερχείλιση (overflow) της στοίβας.
2. Η **απώθηση** (**pop**) στοιχείου από τη στοίβα. Στη διαδικασία της απώθησης ελέγχουμε αν υπάρχει ένα τουλάχιστον στοιχείο στη στοίβα. Στην περίπτωση που προσπαθήσουμε να «αφαιρέσουμε» ένα στοιχείο από μία κενή στοίβα, έχουμε υποχείλιση (underflow) της στοίβας.

13. Πως **υλοποιείται η στοίβα** με χρήση μονοδιάστατου **πίνανα**;



Χρησιμοποιούμε μια μεταβλητή (top), που δείχνει το στοιχείο που τοποθετήθηκε τελευταίο στην κορυφή της στοίβας, η οποία έχει τιμή 0, όταν η στοίβα είναι άδεια.
Η **ώθηση** ενός νέου στοιχείου στη στοίβα (εισαγωγή στοιχείου στον πίνακα) γίνεται πάντα στην κορυφή της.
Συγκεκριμένα, η μεταβλητή top αυξάνεται κατά ένα: top ← top+1
και στη συνέχεια γίνεται η ώθηση του στοιχείου.
Η **απώθηση** ενός στοιχείου από τη στοίβα (εξαγωγή από τον πίνακα) γίνεται πάντα από την κορυφή της στοίβας. Συγκεκριμένα, εξάγεται το στοιχείο που δείχνει η μεταβλητή top και στη συνέχεια η μεταβλητή top μειώνεται κατά ένα: top ← top-1

14. Τι ονομάζεται ουρά;

Ουρά (Queue), ονομάζεται μια **δομή δεδομένων** το σύνολο των στοιχείων της οποίας είναι διατεταγμένο με τέτοιο τρόπο, ώστε τα **στοιχεία που τοποθετήθηκαν πρώτα στην ουρά να λαμβάνονται επίσης πρώτα**. Η παραπάνω μέθοδος ονομάζεται Πρώτο Μέσα, Πρώτο Έξω ή FIFO **(=First In First Out**).

15. Ποιες είναι οι κύριες λειτουργίες σε μια ουρά;
   1. Η **εισαγωγή** (enqueue) στοιχείου στο πίσω άκρο της ουράς.
   2. Η **εξαγωγή** (dequeue) στοιχείου από το εμπρός άκρο της ουράς.

16. Πως υλοποιείται η ουρά με χρήση μονοδιάστατου πίνανα;
Χρησιμοποιούμε δύο μεταβλητές, την **front** (ή εμπρός) που δείχνει τη **θέση του 1ου στοιχείου** της ουράς και την **rear** (ή πίσω) που δείχνει τη **θέση του τελευταίου στοιχείου**. Ως αρχικές τιμές των μεταβλητών rear και front θεωρούμε το μηδέν.
Η **εισαγωγή** ενός νέου στοιχείου γίνεται από το πίσω άκρο της ουράς και η τιμή της μεταβλητής rear αλλάζει ως εξής: rear ← rear+1
Κατά την εισαγωγή, πρώτα αυξάνουμε τον δείκτη rear κατά ένα και μετά εισάγουμε το στοιχείο στον πίνακα. Αυτό υπό την προϋπόθεση ότι υπάρχει χώρος (rear < max). Αν το στοιχείου που βάζουμε είναι το πρώτο (rear=1) τότε θέτουμε και front ← 1
Η εξαγωγή ενός στοιχείου γίνεται από το εμπρός άκρο της ουράς και η τιμή της μεταβλητής front αλλάζει ως εξής: front ← front +1
Κατά την εξαγωγή ενός στοιχείου, αυξάνεται ο δείκτης front κατά ένα (δείχνει στην επόμενη θέση του πίνακα) χωρίς στην πραγματικότητα να γίνεται καμία παρέμβαση στα περιεχόμενα του πίνακα (χωρίς να διαγράφεται κάποιο στοιχείο). Αυτό υπό την προϋπόθεση ότι υπάρχει στοιχείο (front <= rear και front > 0).
Όταν γίνεται εξαγωγή του τελευταίου στοιχείου της ουρά (front = rear) οι δείκτες παίρνουν πάλι τις αρχικές τιμές (0).

Κεφάλαιο 4 διαίρει και βασίλευε

Τι είναι η μέθοδος σχεδίασης αλγορίθμων **«Διαίρει και Βασίλευε» ;**

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

Ο μέγιστος αριθμός των συγκρίσεων (επαναλήψεων) που απαιτούνται για την εύρεση ενός στοιχείου σε ένα σύνολο «n» ταξινομημένων στοιχείων δίνεται από το ακέραιο μέρος του [log2(n)+1]

Κεφάλαιο 6 γλώσσες προγραμματισμού

1. Τι είναι οι φυσικές και τι είναι οι τεχνητές γλώσσες:

Οι φυσικές γλώσσες πχ ελληνικά είναι γλώσσες που χρησιμοποιούνται για την επικοινωνία ανθρώπου με άνθρωπο. Οι τεχνητές γλώσσες πχ java είναι γλώσσες που χρησιμοποιούνται για την επικοινωνία ανθρώπου με υπολογιστή.

2. Στοιχεία γλωσσών προγραμματισμού. **Ομοιότητες φυσικών-Τεχνητών γλωσσών**

Μια γλώσσα προγραμματισμού (όπως και οι φυσικές γλώσσες) προσδιορίζεται από τέσσερα στοιχεία:

a. Το αλφάβητο.
Είναι το σύνολο των στοιχείων που αποτελεί την γλώσσα. (Πχ η ελληνική γλώσσα περιέχει τα εξής στοιχεία: 48 χαρακτήρες, τα σημεία στίξης καθώς και τα ψηφία)

b. Το λεξιλόγιο.
Είναι το υποσύνολο όλων των ακολουθιών που δημιουργούνται από το αλφάβητο και είναι δεκτές από την γλώσσα. Πχ η ακολουθία ΑΒΓΑ είναι δεκτή ενώ η ΑΒΓΒΑ δεν είναι.

c. Την γραμματική.
Η γραμματική αποτελείται από το τυπικό και το συντακτικό.

i. Τυπικό: είναι το σύνολο των κανόνων που καθορίζουν αν μία λέξη είναι αποδεκτή. Πχ Η λέξη «ΓΡΑΨΕ» είναι αποδεκτή αλλά η «ΓΡΑΨΕΣ» όχι.

ii. Συντακτικό: Είναι το σύνολο των κανόνων που καθορίζει αν η διάταξη και η σύνδεση των λέξεων σε μία πρόταση είναι σωστή.

d. Την σημασιολογία.
Είναι το σύνολο των κανόνων που καθορίζει το νόημα των λέξεων άρα και το νόημα των προτάσεων που δημιουργούνται. Πχ Απογειώθηκε το ντροπαλό κατσαβίδι. (δεν έχει νόημα..) Στις γλώσσες προγραμματισμού ο δημιουργός τους αποφασίζει για την σημασιολογία των λέξεων της γλώσσας.

3. Διαφορά φυσικών – τεχνητών γλωσσών

Μια βασική διαφορά είναι η δυνατότητα εξέλιξης:
Οι φυσικές γλώσσες εξελίσσονται συνεχώς και δημιουργούνται νέες λέξεις, αλλάζουν οι κανόνες γραμματικής/ συντακτικού με την πάροδο του χρόνου. Αυτό γίνεται γιατί χρησιμοποιούνται για την επικοινωνία μεταξύ των ανθρώπων.
Οι γλώσσες προγραμματισμού χρησιμοποιούνται για την επικοινωνία ανθρώπου Η/Υ και χαρακτηρίζονται από στασιμότητα. Ωστόσο και αυτές εξελίσσονται από τους δημιουργούς τους για να διορθώσουν αδυναμίες τους ή να καλύψουν μεγαλύτερο εύρος εφαρμογών αλλά και να ακολουθήσουν τις νέες εξελίξεις.

4. Να περιγράψετε την τεχνική της ιεραρχικής σχεδίασης προγράμματος.

Η τεχνική της ιεραρχικής σχεδίασης (ή αλλιώς τεχνική από επάνω προς τα κάτω ή top-down), χρησιμοποιεί την στρατηγική της συνεχούς διαίρεσης του προβλήματος σε υποπροβλήματα τα οποία είναι εύκολο να επιλυθούν οδηγώντας στην επίλυση του αρχικού προβλήματος. Περιλαμβάνει:

a. Τον καθορισμό των βασικών λειτουργιών ενός προγράμματος σε ανώτερο επίπεδο,

b. Την διάσπαση των λειτουργιών αυτών σε όλο και μικρότερες λειτουργίες μέχρι το τελευταίο επίπεδο όπου οι λειτουργίες είναι πολύ απλές.

Υλοποιείται με τμηματικό προγραμματισμό.

5. Δομημένος προγραμματισμός

Δεν είναι απλώς ένα είδος προγραμματισμού αλλά μια μεθοδολογία σύνταξης προγραμμάτων.

6. Βασικές αρχές του δομημένου προγραμματισμού:

1. Όλα τα προγράμματα μπορούν να γραφούν χρησιμοποιώντας **μόνο τις τρεις** παρακάτω λογικές δομές καθώς και συνδυασμών τους. Δομή ακολουθίας. Δομή επιλογής. Δομή επανάληψης.
2. Κάθε πρόγραμμα όπως και κάθε ενότητα προγράμματος έχει **μία είσοδο και μόνο μία έξοδο.**

Αν και ο δομημένος προγραμματισμός αρχικά εμφανίστηκε ως μία προσπάθεια περιορισμού της εντολής GOTO σήμερα αποτελεί την βασική μεθοδολογία προγραμματισμού. Ο δομημένος προγραμματισμός βοηθάει την ανάλυση ενός προγράμματος σε τμήματα έτσι περιέχει τόσο την ιεραρχική σχεδίαση όσο και τον τμηματικό προγραμματισμό.

7. Γιατί η εντολή GOTO κρίνεται ακατάλληλη

Η χρήση της εντολής αλλάζει την ροή ενός προγράμματος Η αλλαγή αυτή της ροής κάνει τα προγράμματα δύσκολα στην παρακολούθηση την κατανόηση και την συντήρηση.

8. Πλεονεκτήματα δομημένου προγραμματισμού

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

9. Πηγαίο πρόγραμμα (source)

Το αρχικό πρόγραμμα που γράφει ο προγραμματιστής λέγεται πηγαίο πρόγραμμα. Η συγγραφή του γίνεται με τον συντάκτη {*βλέπε παρακάτω*}. Το πηγαίο πρόγραμμα δεν είναι κατανοητό από τον υπολογιστή. (εκτός αν είναι γραμμένο απ' ευθείας σε γλώσσα μηχανής)

10. Λάθη στο πηγαίο πρόγραμμα

Τα λάθη σε ένα πρόγραμμα διακρίνονται σε λογικά και συντακτικά.

a. **Λογικά** λάθη.
   Οφείλονται σε σφάλματα κατά την υλοποίηση του αλγορίθμου.
   {*πχ ΜΟ ← x+ψ / 3 (ήθελε παρενθέσεις) …*}
   Εντοπίζονται παρά μόνο κατά την εκτέλεση του προγράμματος.
   Είναι τα πλέον σοβαρά και δύσκολα στην διόρθωση.
b. **Συντακτικά** λάθη.
   Οφείλονται σε αναγραμματισμούς γραμμάτων εντολών, παράληψη δήλωσης δεδομένων κλπ. {*πχ Γάρψε αυτί για Γράψε*}
   Εντοπίζονται από τον μεταγλωττιστή ή τον διερμηνευτή
   Πρέπει να διορθωθούν ώστε να δημιουργηθεί το εκτελέσιμο πρόγραμμα.

11. Συντάκτης

Ο συντάκτης είναι ουσιαστικά ένας μικρός επεξεργαστής κειμένου. (Σαν το Word - Wordpad κλπ.) Η συγγραφή του πηγαίου προγράμματος και η διόρθωση των λαθών του πηγαίου γίνεται με την βοήθεια του συντάκτη.

12. Μεταγλωττιστής

Δέχεται σαν είσοδο ένα πρόγραμμα γραμμένο σε μία γλώσσα προγραμματισμού (υψηλού επιπέδου).

Ανιχνεύει τα τυχόν συντακτικά λάθη. Αν βρεθούν λάθη ο προγραμματιστής τα διορθώνει (με τον συντάκτη) και υποβάλλει το πρόγραμμα ξανά προς μεταγλώττιση μέχρι να παραχθεί το σωστό.

Αν δεν υπάρχουν λάθη και μόνο τότε, παράγει το **αντικείμενο πρόγραμμα,** το οποίο είναι ισοδύναμο με το πηγαίο αλλά εκφρασμένο πλέον σε γλώσσα μηχανής. Αυτό είναι πλέον τελείως ανεξάρτητο από το αρχικό πρόγραμμα, αλλά δεν είναι ακόμη εκτελέσιμο:

Η διαδικασία μέσω της οποίας καταλήγουμε στο εκτελέσιμο πρόγραμμα (γλώσσα μηχανής) είναι συνοπτικά:

Πηγαίο πρόγραμμα →μεταγλωττιστής →αντικείμενο πρόγραμμα → συνδέτης-φορτωτής → Εκτελέσιμο πρόγραμμα.

13. Συνδέτης φορτωτής

Το αντικείμενο πρόγραμμα που παράγει ο μεταγλωττιστής είναι μεν σε μορφή κατανοητή από τον υπολογιστή (γλώσσα μηχανής) αλλά δεν είναι εκτελέσιμο. Χρειάζεται να συμπληρωθεί και να συνδεθεί με άλλα τμήματα προγράμματος απαραίτητα για την εκτέλεσή του. (Τα τμήματα αυτά είτε τα γράφει ο προγραμματιστής (υποπρογράμματα), είτε βρίσκονται στις βιβλιοθήκες της γλώσσας που χρησιμοποιεί.) Το πρόγραμμα που κάνει την σύνδεση αυτή ονομάζεται συνδέτης- φορτωτής .

14. Διερμηνευτής

Ο διερμηνευτής δέχεται ως είσοδο ένα πρόγραμμα γραμμένο σε γλώσσα υψηλού επιπέδου και εκτελεί μία μία τις εντολές του αρχικού προγράμματος.
Η διαδικασία εκτέλεσης είναι η εξής:
Διαβάζει μία από τις εντολές του αρχικού προγράμματος,
ανιχνεύει τα (τυχόν) συντακτικά λάθη κάθε της εντολής,
την μεταφράζει σε γλώσσα μηχανής
την εκτελεί.
Κατόπιν διαβάζει την επόμενη εντολή και επαναλαμβάνει την ίδια διαδικασία !

15. Ομοιότητες – Διαφορές διερμηνευτή-μεταγλωττιστή

Ομοιότητες:

- **a.** Και οι δύο μεταφράζουν το πηγαίο πρόγραμμα (υψηλού επιπέδου) σε γλώσσα μηχανής.
- **b.** Και οι δύο ανιχνεύουν τα συντακτικά λάθη.

Διαφορές:

- **c.** Ο μεταγλωττιστής μεταγλωττίζει όλο το πρόγραμμα και με την βοήθεια του συνδέτη – φορτωτή παράγεται το εκτελέσιμο.
- **d.** Ο διερμηνευτής εκτελεί μία μία τις εντολές και δεν χρειάζεται συνδέτη-φορτωτή

16. Διαφορές ως απόρροια των παραπάνω είναι:
Ο διερμηνευτής αφού εκτελεί τις εντολές μία μία έχει το πλεονέκτημα της άμεσης διόρθωσης των λαθών. Για τον λόγο αυτό χρησιμοποιείται συνήθως κατά την συγγραφή-διόρθωση ενός προγράμματος.
Η εκτέλεση ενός προγράμματος με τον διερμηνευτή είναι πιο αργή, γιατί για να εκτελεστεί το πρόγραμμα, πρέπει κάθε φορά να ξαναγίνεται η διερμηνεία από την αρχή, ενώ ο μεταγλωττιστής παράγει μια φορά το αντικείμενο πρόγραμμα και δεν χρειάζεται ξανά μεταγλώττιση αφού είναι σχεδόν εκτελέσιμο (θυμήσου τον συνδέτη)
Για να εκτελεστεί ένα προγράμμα με τον διερμηνευτή είναι απαραίτητη η παρουσία του πηγαίου προγράμματος ενώ με τον μεταγλωττιστή μόνο την πρώτη φορά.

17. (Σύγχρονα) προγραμματιστικά περιβάλλοντα

Για την δημιουργία - εκτέλεση ενός προγράμματος απαιτούνται τουλάχιστον 3 προγράμματα:
Συντάκτης – Μεταγλωττιστής - Συνδέτης

Κεφάλαια 7ο και 8ο προγράμματα

1. **Κανόνες ονομασίας μεταβλητών**
   1. Τα ονόματα αυτά μπορούν να αποτελούνται από Γράμματα (Κεφαλαία και μικρά ελληνικού αλφαβήτου (Α-Ω , α-ω), Κεφαλαία και μικρά αγγλικού αλφαβήτου (A-Z, a-z)), Ψηφία: 0-1 και _ (κάτω παύλα)
   2. Απαγορεύεται η χρήση δεσμευμένων λέξεων ως ονόματα μεταβλητών (πχ γράψε διάβασε)
   3. Τα ονόματα μεταβλητών υποχρεωτικά αρχίζουν από γράμμα (όχι ψηφίο ή underscore)

2. Τελεσταίοι (operands)
Είναι οι Σταθερές(constants) και οι Μεταβλητές(variables)

3. Σταθερές (ή αλλιώς συμβολικές σταθερές)
Είναι προκαθορισμένες τιμές που δεν μεταβάλλονται κατά την διάρκεια εκτέλεσης του προγράμματος. Μπορεί να είναι τύπου ακέραιες, πραγματικές, λογικές, χαρακτήρες.

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

5. Αριθμητικές εκφράσεις.
Όταν μία τιμή προέρχεται από αριθμητικό υπολογισμό τότε αναφερόμαστε σε αριθμητικές εκφράσεις.

6. Λογική έκφραση
Μία λογική έκφραση είναι μία έκφραση που έχει λογική τιμή δηλαδή αληθής ή ψευδής.

7. Σύνθετες λογικές εκφράσεις
Σχηματίζονται από απλές λογικές εκφράσεις με την χρήση των λογικών τελεστών: Όχι, Και, Η. Η ιεραρχία τους είναι Όχι , Και , Η , δηλαδή Ο.Κ.Η.

8. Τελεστές (operators)
Είναι σύμβολα που χρησιμοποιούνται στις διάφορες πράξεις. Διακρίνονται σε τρεις κατηγορίες:

Αριθμητικοί:	**Λογικοί**:	**Συγκριτικοί**:
^	Οχι (άρνησης)	= (ίσο)
* / div mod	Και (σύζευξης)	<> (διάφορο)
+ -	Η (διάζευξης)	>= (μεγαλύτερο ή ίσο) κλπ.

9. Εκφράσεις (expressions)
Όταν μία τιμή προκύπτει από υπολογισμό τότε αναφερόμαστε σε εκφράσεις. Οι τελεστές μαζί με τους τελεσταίους διαμορφώνουν τις εκφράσεις.

10. Δομή απλής επιλογής (Αν..τοτε)

```
Αν συνθήκη τότε
  Εντολή-1
  ..
  Εντολή-ν
τελος_αν
```

```
Υπάρχει και η εντολή

Αν συνθήκη τότε εντολή

Πχ αν χ>2 τότε κ←1

Χωρίς τέλος_αν δηλαδή
```

Λειτουργία :Αν η συνθήκη ισχύει τότε εκτελούνται οι εντολές που βρίσκονται ανάμεσα στο τότε και στο τέλος_αν, αλλιώς αγνοούνται.Η εκτέλεση συνεχίζεται με την εντολή μετά το τέλ ος_αν

11. Δομή σύνθετης επιλογής (Αν..τοτε..αλλιώς)

Αν συνθήκη τότε
  Εντολή-1
  ..
  Εντολή-ν
Αλλιώς
  Εντολή-1
  ..
  Εντολή-ν
τελος_αν

Λειτουργία: Αν η συνθήκη ισχύει τότε εκτελούνται οι εντολές που βρίσκονται ανάμεσα στο τότε και στο αλλιώς, αλλιώς εκτελούνται οι εντολές που βρίσκονται ανάμεσα στο αλλιώς και στο τέλος_αν. Η εκτέλεση συνεχίζεται την εντολή μετά το τέλος_αν.

12. Δομή πολλαπλής επιλογής (Αν..τοτε..αλλιώς_αν)

Αν συνθήκη-1 τότε
  Εντολή-1
  ..
  Εντολή-ν
Αλλιώς_αν συνθήκη-2 τότε
  Εντολή-1
  ..
  Εντολή-ν
Αλλιώς
  Εντολή-1
  ..
  Εντολή-ν
τελος_αν

Λειτουργία Εκτελούνται οι εντολές που βρίσκονται στο αντίστοιχο τμήμα όταν η συνθήκη είναι αληθής. Η εκτέλεση συνεχίζεται την εντολή μετά το τέλος_αν.

13. Εμφωλευμένα αν

Ονομάζονται δύο ή περισσότερες εντολές Αν που περιέχονται η μία μέσα στην άλλη.
Η χρήση εμφωλευμένων αν οδηγεί συνήθως σε πολύπλοκες δομές που αυξάνουν την πιθανότητα λάθους και κάνουν το πρόγραμμα δυσνόητο.

14. Δομή επανάληψης

Η δομή επανάληψης έχει τρεις εντολές: Για, όσο και μέχρις_ότου.
Όλες ελέγχονται από μία συνθήκη ή οποία καθορίζει την έξοδο από τον βρόχο.

15. Εντολή Οσο..επανάλαβε

Όσο συνθήκη επανάλαβε
Εντολή-1
Εντολή-2
..
Εντολή-ν
Τελος_επανάληψης

Λειτουργία: Ελέγχεται η συνθήκη και αν είναι αληθής εκτελούνται οι εντολές που βρίσκονται ανάμεσα στις όσο και τέλος_επανάληψης. Στην συνέχεια εκτελείται πάλι η συνθήκη και αν ισχύει εκτελούνται πάλι οι ίδιες εντολές. Όταν η συνθήκη γίνει ψευδής τότε σταματά η επανάληψη και εκτελείται η εντολή μετά το τέλος_επανάληψης

Παρατηρήσεις:

- **a.** Με την δομή ΟΣΟ μπορούν να εκφραστούν όλες οι άλλες δομές επανάληψης
- **b.** Χαρακτηριστικό της είναι ότι ο αριθμός των επαναλήψεων δεν είναι γνωστός.
- **c.** Για να σταματήσει η επανάληψη πρέπει υποχρεωτικά μέσα στον βρόχο να υπάρχει μία εντολή η οποία κάνει την συνθήκη ψευδή.

16. Δομή μέχρις_οτου

Αρχή_επανάληψης
Εντολή-1
Εντολή-2
..
Εντολή-ν
Μέχρις_οτου συνθήκη

Λειτουργία: Εκτελούνται οι εντολές μεταξύ των αρχή_επανάληψης και μέχρις_οτου.Στη συνέχεια ελέγχεται η συνθήκη και αν είναι ψευδής τότε εκτελούνται πάλι οι εντολές. Όταν η συνθήκη βρεθεί αληθής τότε σταματά η επανάληψη και εκτελείται η εντολή μετά το μέχρις_ότου.

Παρατηρήσεις: εκτελείται τουλάχιστον μία φορά. Η επανάληψη συνεχίζει όταν η συνθήκη γίνει ψευδής. Χρήσιμη για έλεγχο εγκυρότητας, μενού επιλογής και ασκήσεις που φρενάρουν με ερώτηση αν ο χρήστης θέλει να συνεχίσει ή όχι.

17. Δομή για από μέχρι με_βήμα

Για μεταβλητή από τιμή1 μέχρι τιμή2 με βήμα τιμή3
Εντολή-1
Εντολή-2
..
Εντολή-ν
Τελος_επανάληψης

Λειτουργία: Οι εντολές του βρόχου εκτελούνται για όλες τις τιμές της μεταβλητής από την τιμή1 μέχρι την τιμή2 αυξανόμενες κατά την τιμή3. Αν το βήμα είναι 1 τότε παραλείπεται

18. Εμφωλευμένοι βρόχοι

Ονομάζονται δύο ή περισσότεροι βρόχοι που βρίσκονται ο ένας μέσα στον άλλο.

19. Κανόνες εμφωλευμένων βρόχων

- **a.** Ο εσωτερικός βρόχος πρέπει να βρίσκεται ολόκληρος μέσα στον εξωτερικό, έτσι ο βρόχος που ξεκινάει τελευταίος ολοκληρώνεται πρώτος.
- **b.** Η είσοδος σε ένα βρόχο υποχρεωτικά γίνεται από την αρχή του.
- **c.** Δεν μπορεί να χρησιμοποιηθεί η ίδια μεταβλητή ως μετρητής

Κεφάλαιο 9

1. **Ορισμός πίνακα**
   Είναι ένα σύνολο αντικειμένων ιδίου τύπου τα οποία αναφέρονται με ένα κοινό όνομα. Κάθε ένα από τα αντικείμενα που τον αποτελούν ονομάζεται στοιχείο. Η αναφορά σε ένα στοιχείο το πίνακα γίνεται με το όνομά του πίνακα ακολουθούμενο από **ένα** δείκτη.

2. **Γενικά περί πινάκων**
   Τύπος δεδομένων: Κάθε πίνακας περιέχει υποχρεωτικά δεδομένα του ίδιου τύπου (ακέραια , πραγματικά κλπ)
   Δήλωση πίνακα: Ο τύπος του πίνακα καθώς και ο αριθμός των στοιχείων του δηλώνεται στο τμήμα δηλώσεων μεταβλητών.
   Όνομα: Το όνομα ενός πίνακα καθορίζει μια ομάδα διαδοχικών θέσεων μνήμης.
   Στοιχείο: Κάθε συγκεκριμένη θέση μνήμης καλείται στοιχείο του πίνακα και προσδιορίζεται από την τιμή ενός δείκτη
   Χρήση πινάκων: Η ανάγνωση, επεξεργασία και εμφάνιση των στοιχείων των πινάκων γίνεται πάντοτε από βρόχους. Συνήθως με την χρήση της δομής Για αφού είναι προκαθορισμένο το πλήθος των στοιχείων

3. **μειονεκτήματα πινάκων:**
   1. **Οι πίνακες απαιτούν μνήμη**: Κάθε πίνακας δεσμεύει από την αρχή του προγράμματος πολλές θέσεις μνήμης. Σε ένα μεγάλο και σύνθετο πρόγραμμα η άσκοπη χρήση πινάκων μπορεί να οδηγήσει ακόμη και σε αδυναμία εκτέλεσης του προγράμματος.
   2. **Οι πίνακες περιορίζουν τις δυνατότητες του προγράμματος**: Οι πίνακες είναι στατικές δομές δεδομένων αφού το μέγεθός τους, δηλώνεται στην αρχή του προγράμματος παραμένει υποχρεωτικά σταθερό κατά την εκτέλεση.

4. **Πότε πρέπει να χρησιμοποιούνται;**
   Όταν τα δεδομένα που εισάγονται σε ένα πρόγραμμα πρέπει να διατηρούνται στην μνήμη μέχρι το τέλος της εκτέλεσης.
   Η απόφαση για την χρήση ή όχι πίνακα είναι θέμα εμπειρίας στον προγραμματισμό.
   {Γενικά καλό είναι να αποφεύγονται κυρίως γιατί καταναλώνουν πολύ μνήμη}

5. **Τυπικές επεξεργασίες πινάκων**
   1. **Υπολογισμός αθροισμάτων στοιχείων του πίνακα**. Πολύ συχνά απαιτείται η εύρεση του αθροίσματος των στοιχείων ενός πίνακα ή του αθροίσματος των στοιχείων που έχουν μία ιδιότητα.
   2. **Εύρεση μεγίστου-ελαχίστου**. Αν ο πίνακας δεν είναι ταξινομημένος τότε πρέπει να συγκριθούν τα στοιχεία ένα προς ένα, αλλιώς αν είναι ταξινομημένος το μέγιστο και ελάχιστο βρίσκονται προφανώς στα δύο ακριανά στοιχεία του.
   3. **Ταξινόμηση των στοιχείων του πίνακα**. Ένας αλγόριθμος ταξινόμησης είναι ο αλγόριθμος φυσαλίδας αν και δεν είναι ο αποδοτικότερος.
   4. **Αναζήτηση ενός στοιχείου του πίνακα**. Δύο είναι οι πλέον διαδεδομένοι αλγόριθμοι: Η σειριακή αναζήτηση, Δυαδική αναζήτηση (μόνο σε ταξινομημένους πίνακες)
   5. **Συγχώνευση δύο πινάκων**. Σκοπός της είναι η ένωση δύο ή περισσότερων ταξινομημένων πινάκων σε έναν που είναι και αυτός ταξινομημένος

Κεφάλαιο 10ο

1. **Τμηματικός προγραμματισμός**
   Τμηματικός προγραμματισμός ονομάζεται η τεχνική σχεδίασης και ανάπτυξης των προγραμμάτων ως ένα σύνολο από απλούστερα τμήματα (προγραμμάτων).

2. **Τι είναι υποπρόγραμμα**
   Ονομάζεται ένα τμήμα προγράμματος που επιτελεί **ένα αυτόνομο έργο** και **έχει γραφεί χωριστά** από το υπόλοιπο πρόγραμμα (subprogram).

3. **Βασικές ΙΔΙΟΤΗΤΕΣ-ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ των υποπρογραμμάτων**
   1. **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.
   2. **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.
   3. **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.

4. **Πλεονεκτήματα του τμηματικού προγραμματισμού**
   1. **Διευκολύνει την ανάπτυξη του αλγορίθμου και του αντίστοιχου προγράμματος:** Αντί να αντιμετωπίζουμε το συνολικό πρόβλημα εξετάζουμε και επιλύουμε τα υποπροβλήματα στα οποία αναλύεται, τα οποία είναι πιο απλά! Με τη σταδιακή επίλυση των υποπροβλημάτων και τη δημιουργία των αντιστοίχων υποπρογραμμάτων τελικά επιλύεται το αντίστοιχο 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. Αυτό είναι πολύ σημαντικό χαρακτηριστικό του σωστού προγραμματισμού, αφού ένα μεγάλο πρόγραμμα, 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 κατανοητό.
   4. **Επεκτείνει τις δυνατότητες των γλωσσών προγραμματισμού.** Ένα υποπρόγραμμα 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.

5. **Παράμετροι**
   Μια παράμετρος είναι μια μεταβλητή που επιτρέπει το πέρασμά της τιμής της από ένα τμήμα προγράμματος σε ένα άλλο.

6. **Διαδικασίες**
   Είναι ένας τύπος υποπρογραμμάτων που μπορεί να εκτελέσει όλες τις λειτουργίες ενός προγράμματος

7. **Συναρτήσεις**
   Είναι ένας τύπος υποπρογράμματος που υπολογίζει και επιστρέφει μόνο μια τιμή με το όνομά της.

8. **Η λίστα παραμέτρων στην διαδικασία δεν είναι υποχρεωτική. Δηλαδή μια διαδικασία μπορεί να έχει μια καμία ή περισσότερες παραμέτρους**

9. **Πραγματικές και τυπικές παράμετροι**
   Όλες οι μεταβλητές έχουν ισχύ μόνο για το τμήμα προγράμματος στο οποίο έχουν δηλωθεί.
   Η λίστα των τυπικών παραμέτρων καθορίζει τις παραμέτρους στην δήλωση του υποπρογράμματος. Μερικές γλώσσες προγραμματισμού τις ονομάζουν ορίσματα
   Η λίστα των πραγματικών παραμέτρων καθορίζει τις παραμέτρους στην κλήση του υποπρογράμματος. Μερικές γλώσσες προγραμματισμού τις ονομάζουν απλά παραμέτρους.
   Τα ονόματα των τυπικών παραμέτρων και πραγματικών παραμέτρων μπορούν να είναι οποιαδήποτε. Αυτό δεν σημαίνει ότι αν έχουν το ίδιο όνομα είναι η ίδια μεταβλητή. Απεναντίας είναι υποχρεωτικά διαφορετικές μεταβλητές αφού ανήκουν σε διαφορετικά τμήματα προγράμματος

10. **Η χρήση της στοίβας στην κλήση διαδικασιών.**
    Όταν μια διαδικασία ή συνάρτηση καλείται από το κύριο πρόγραμμα τότε η αμέσως επόμενη εντολή-διεύθυνση του κυρίως προγράμματος (διεύθυνση επιστροφής) αποθηκεύεται από τον μεταφραστή σε μια στοίβα που ονομάζεται στοίβα χρόνου εκτέλεσης.
    Μετά την εκτέλεση της διαδικασίας ή της συνάρτησης η διεύθυνση επιστροφής απωθείται από την στοίβα και έτσι ο έλεγχος του προγράμματος μεταφέρεται και πάλι στο κυρίως πρόγραμμα.
    Η τεχνική αυτή εφαρμόζεται οποτεδήποτε μια διαδικασία ή συνάρτηση καλεί μια άλλη διαδικασία ή συνάρτηση.

11. **Τυπικές και πραγματικές παράμετροι**
    Σχηματικά έχουμε:

Τμήμα κύριου προγράμματος
Διάβασε α,β
Κάλεσε Διαδ(**α,β,Σ**)
Γράψε Σ

Διαδικασία
Διαδικασία Διαδ(**κ,λ,Σ**)
….
Τέλος_Διαδικασίας

Οι μεταβλητές στην λίστα του κυρίου προγράμματος α,β, Σ λέγονται **πραγματικές παράμετροι** ή απλά **παράμετροι**.
Οι μεταβλητές στην λίστα παραμέτρων της διαδικασίας κ,λ, Σ λέγονται **τυπικές παράμετροι** ή **ορίσματα**.

Εμβέλεια μεταβλητών:

Μεταβλητές	Ισχύς	Εμβέλεια
Τοπικές	Ισχύουν μόνο εκεί που δηλώθηκαν	Περιορισμένη
Καθολικές	Έχουν ισχύ σε όλα τα προγράμματα που τις χρησιμοποιούν	απεριόριστη

12. Τι γνωρίζετε για την απεριόριστη εμβέλεια μεταβλητών; Ποια τα μειονεκτήματά της;
Σύμφωνα με αυτήν την αρχή, όλες οι μεταβλητές και όλες οι σταθερές είναι γνωστές και μπορούν να χρησιμοποιηθούν σε οποιοδήποτε τμήμα προγράμματος, άσχετα που δηλώθηκαν. Δηλαδή οι μεταβλητές είναι καθολικές.
Μειονεκτήματα:
1. Καταστρατηγεί την αρχή της αυτονομίας των υποπρογραμμάτων.
2. Δημιουργεί πολλά προβλήματα και τελικά είναι αδύνατη η χρήση της για μεγάλα προγράμματα τα οποία χρησιμοποιούν πολλά υποπρογράμματα.

13. Τι γνωρίζετε για την περιορισμένη εμβέλεια; Ποια τα πλεονεκτήματά της;
Η περιορισμένη εμβέλεια (αυτή που χρησιμοποιείτε όλη την χρονιά στις ασκήσεις σας) υποχρεώνει όλες τις μεταβλητές που χρησιμοποιούνται σε ένα τμήμα προγράμματος να δηλώνονται σε αυτό το τμήμα. Ισχύουν δηλαδή μόνο για το πρόγραμμα-υποπρόγραμμα στο οποίο δηλώθηκαν.
Πλεονεκτήματα:
1. Προσφέρει απόλυτη αυτονομία.
2. Δίνει την δυνατότητα να χρησιμοποιούμε οποιοδήποτε όνομα μεταβλητής θέλουμε χωρίς να μας ενδιαφέρει αν χρησιμοποιήθηκε σε άλλο υποπρόγραμμα.

14. Τι γνωρίζετε για την μερικώς περιορισμένη εμβέλεια;

Σύμφωνα με αυτή την αρχή άλλες μεταβλητές είναι τοπικές και άλλες καθολικές. Κάθε γλώσσα προγραμματισμού έχει τους δικούς της κανόνες και μηχανισμούς για τον τρόπο και τις προϋποθέσεις που ορίζονται οι μεταβλητές ως τοπικές ή καθολικές. Η μερικώς περιορισμένη εμβέλεια προσφέρει μερικά πλεονεκτήματα στον πεπειραμένο προγραμματιστή, αλλά για τον αρχάριο περιπλέκει το πρόγραμμα δυσκολεύοντας την ανάπτυξή του

Κεφάλαιο 13 εκσφαλμάτωση

1. **Ποιες είναι οι βασικές κατηγορίες λαθών που είναι δυνατό να παρουσιαστούν στα προγράμματα;**
   1. Λάθη **κατά την υλοποίηση (Συντακτικά λάθη)**
   2. Λάθη **κατά την εκτέλεση** (Λάθη που οδηγούν σε αντικανονικό τερματισμό του προγράμματος)
   3. **Λογικά λάθη** (Λογικά λάθη που παράγουν λανθασμένα αποτελέσματα)

2. **Που οφείλονται συνήθως τα λάθη κατά την υλοποίηση και πότε γίνονται αντιληπτά;**
   Τα λάθη κατά το χρόνο υλοποίησης προκαλούνται κυρίως από λανθασμένη σύνταξη εντολών προγράμματος. Τέτοια λάθη μπορεί να είναι η λανθασμένη συγγραφή μιας δεσμευμένης λέξης της γλώσσας προγραμματισμού ή η χρήση μιας δομής ελέγχου χωρίς την εντολή τερματισμού της. Ανιχνεύονται από τον μεταγλωττιστή, ο οποίος εμφανίζει προς τον προγραμματιστή κάποιο προειδοποιητικό μήνυμα και δεν επιτρέπεται η εκτέλεσή του προγράμματος μέχρι να το διορθώσει ο προγραμματιστής.
3. **Που οφείλονται συνήθως τα λάθη κατά την εκτέλεση και πότε γίνονται αντιληπτά;**
   Τα λάθη που προκαλούν τον αντικανονικό τερματισμό της εφαρμογής και το κρέμασμα του συστήματος εμφανίζονται σε πραγματικό περιβάλλον εκτέλεσης. Τέτοια λάθη είναι δυνατό να προκληθούν από την προσπάθεια διαίρεσης ενός αριθμού με το μηδέν, την κλήση μιας διαδικασίας με δεδομένα που δεν μπορεί να χειριστεί, όπως η αναζήτηση διαγραμμένων αρχείων, η υπερχείλιση μιας αριθμητικής μεταβλητής ή από δυσλειτουργία του υλικού μέρους του υπολογιστή, όπως η καταστροφή του σκληρού δίσκου του συστήματος, ο τερματισμός μιας σύνδεσης δικτύου και η αποσύνδεση του εκτυπωτή.
4. **Που οφείλονται συνήθως τα λογικά λάθη και πότε γίνονται αντιληπτά;**
   Τα λογικά λάθη είναι συνήθως λάθη σχεδιασμού και δεν προκαλούν τη διακοπή της εκτέλεσης του προγράμματος. Ενώ ο μεταγλωττιστής της γλώσσας προγραμματισμού δεν ανιχνεύει κανένα συντακτικό λάθος και κατά την εκτέλεση του προγράμματος δεν παρουσιάζονται ανεπιθύμητες καταστάσεις σφαλμάτων, τελικά δεν παράγονται τα επιθυμητά αποτελέσματα. Η ανίχνευση τέτοιων λαθών δεν είναι δυνατό να πραγματοποιηθεί από κάποιο εργαλείο του υπολογιστή και διαπιστώνονται μόνο με τη διαδικασία ελέγχου (testing) και την ανάλυση των αποτελεσμάτων των προγραμμάτων.
5. **Τι ονομάζεται εκσφαλμάτωση και ποιος ο στόχος της;**
   Η διαδικασία ελέγχου, εντοπισμού και διόρθωσης των σφαλμάτων ενός προγράμματος καλείται εκσφαλμάτωση (debugging). Στόχος της διαδικασίας εκσφαλμάτωσης είναι ο εντοπισμός των σημείων του προγράμματος που προκαλούν προβλήματα στη λειτουργία του.
6. **Ποιος ο ρόλος των σχολίων στην εκσφαλμάτωση;**
   Η εισαγωγή γραμμών με σχόλια σε ένα πρόγραμμα υποβοηθά σημαντικά την εκσφαλμάτωση. Χρησιμοποιούνται προκειμένου να διαχωρίζονται οι επεξηγηματικές φράσεις από τις λέξεις-κλειδιά του αλγορίθμου.
7. Ποιος ο ρόλος των **σεναρίων ελέγχου** (test case);
   Ένα σενάριο ελέγχου (test case) περιγράφει τα δεδομένα εισόδου ολόκληρου του προγράμματος ή τμήματος του προγράμματος (διαδικασία, συνάρτηση) και τα αναμενόμενα αποτελέσματα. Τα σενάρια ελέγχου εκτελούνται, είτε σε πραγματικό περιβάλλον προγραμματισμού είτε εικονικά με δημιουργία πίνακα τιμών των μεταβλητών. Σε περίπτωση αποκλίσεων μεταξύ των αναμενόμενων και των πραγματικών αποτελεσμάτων, υπάρχει λάθος το οποίο πρέπει να εντοπιστεί και να διορθωθεί.
8. Πώς εφαρμόζεται **η τεχνική ελέγχου μαύρου κουτιού**;

Μια δημοφιλής τεχνική ελέγχου είναι ο έλεγχος μαύρου κουτιού (black-box testing).Ονομάζεται έτσι επειδή τα δεδομένα εισόδου στα σενάρια ελέγχου προκύπτουν από τις προδιαγραφές του προγράμματος, αγνοώντας εντελώς τον κώδικα. Δηλαδή το πρόγραμμα μοιάζει σαν να βρίσκεται μέσα σε ένα μαύρο κουτί που κρύβει το περιεχόμενό του.

9. Ποιος ο ρόλος των αντιπροσωπευτικές τιμές στις τιμές εισόδου;
Ιδανικά θα θέλαμε να ελέγξουμε όλες τις τιμές εισόδου και όλα τα πιθανά αποτελέσματα. Αυτό όμως είναι αδύνατο. Γι' αυτό προσπαθούμε να βρούμε αντιπροσωπευτικές τιμές για τα δεδομένα εισόδου που θα παράγουν αντιπροσωπευτικά αποτελέσματα. Το πρώτο βήμα είναι η δημιουργία ισοδύναμων διαστημάτων τιμών (equivalence partitioning) για τα δεδομένα εισόδου. Τα διαστήματα θεωρούνται ισοδύναμα, καθώς αν δεν υπάρχουν λάθη, τότε όλες οι τιμές ενός διαστήματος εισόδου θα παράγουν τιμές που θα ανήκουν στο ίδιο διάστημα αποτελεσμάτων.
Μετά τον καθορισμό των διαστημάτων πρέπει να επιλεγούν τιμές για τα σενάρια ελέγχου που να καλύπτουν όλα τα διαστήματα. Αφού τα διαστήματα είναι ισοδύναμα, μπορεί να επιλεγεί οποιαδήποτε τιμή από κάθε διάστημα. Μια καλύτερη στρατηγική είναι να γίνει έλεγχος των ακραίων τιμών κάθε διαστήματος (boundary value analysis), καθώς η εμπειρία έχει δείξει ότι τα περισσότερα λάθη γίνονται σε αυτά τα σημεία. Αυτό είναι λογικό, αν σκεφτούμε ότι τα διαστήματα τιμών θα υλοποιηθούν με κάποια μορφή δομής επιλογής, οπότε μπορεί να υπάρχουν λάθη στις λογικές συνθήκες

10. Πώς γίνεται ο έλεγχος στα υποπρογράμματα;
Η τεχνική που περιγράφηκε ανωτέρω, μπορεί να εφαρμοστεί και σε υποπρογράμματα. Αυτό είναι συνηθισμένο σε μεγάλα προγράμματα που αποτελούνται από χιλιάδες γραμμές κώδικα τα οποία είναι οργανωμένα σε υποπρογράμματα. Σε αυτές τις περιπτώσεις πρώτα ελέγχεται κάθε υποπρόγραμμα μεμονωμένα. Αφού διαπιστωθεί η ορθή λειτουργία του καθενός, μόνο τότε πραγματοποιείται έλεγχος ολόκληρου του προγράμματος.

11. Πώς γίνεται ο έλεγχος αν έχουμε πολλές τιμές εισόδου;
Αν οι είσοδοι είναι περισσότερες, τότε κάθε μία πρέπει να ελεγχθεί ανεξάρτητα από τις άλλες. Για να γίνει αυτό, δημιουργούνται σενάρια ελέγχου όπου μία από τις εισόδους λαμβάνει όλες τις ακραίες τιμές ενώ οι υπόλοιπες διατηρούνται σταθερές δίνοντας μια οποιαδήποτε έγκυρη τιμή. Αυτό επαναλαμβάνεται για όλες τις εισόδους. Γίνεται κατανοητό ότι η αύξηση των δεδομένων εισόδου οδηγεί σε μεγάλη αύξηση των σεναρίων ελέγχου, ενώ η διαδικασία αρχίζει να γίνεται περίπλοκη.