Πληροφορική

Περίληψη Θεωρίας Ι

(Παλιά ύλη)

Δημήτρης Παπαδάκης (6974600499)
dimitrisp@easylearn.gr

Περιεχόμενα

Κεφάλαιο 1

Πρόβλημα

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

Η κατανόηση του προβλήματος.

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

Η **κατανόηση ενός προβλήματος** απαιτεί τη **σωστή διατύπωση** εκ μέρους του δημιουργού του και τη **σωστή ερμηνεία** από τη μεριά εκείνου που καλείται να το αντιμετωπίσει.

Δεδομένο

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

Πληροφορία

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

Επεξεργασία δεδομένων

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

Δομή ενός προβλήματος

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

Η κατανόηση ενός προβλήματος είναι βασική προϋπόθεση, για να γίνει στη συνέχεια δυνατή η σωστή αποτύπωση της δομής του.

Η **καταγραφή της δομής** ενός προβλήματος σημαίνει αυτόματα ότι έχει αρχίσει η διαδικασία **ανάλυσης του προβλήματος σε άλλα απλούστερα.**

Με τη σειρά τους **τα νέα προβλήματα** μπορούν να **αναλυθούν σε άλλα**, ακόμα πιο απλά.

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

Περιγραφή και αναπαράσταση προβλημάτων

Φραστική περιγραφή, όπου το πρόβλημα και τα επιμέρους προβλήματα στα οποία αναλύεται περιγράφονται με λόγια.

Γραφική απεικόνιση, όπου χρησιμοποιείται συχνά η **διαγραμματική αναπαράσταση.** Σύμφωνα με αυτή:

- Το αρχικό πρόβλημα αναπαριστάται από ένα ορθογώνιο παραλληλόγραμμο
- Κάθε ένα από τα απλούστερα προβλήματα, στα οποία αναλύεται ένα οποιοδήποτε πρόβλημα, αναπαρίσταται επίσης από ένα ορθογώνιο παραλληλόγραμμο.
- Τα παραλληλόγραμμα που αντιστοιχούν στα απλούστερα προβλήματα, στα οποία αναλύεται ένα οποιοδήποτε πρόβλημα, σχηματίζονται ένα επίπεδο χαμηλότερα.

Έτσι σε κάθε κατώτερο επίπεδο, δημιουργείται η γραφική αναπαράσταση των προβλημάτων, στα οποία αναλύονται τα προβλήματα του υψηλότερου επιπέδου.

Καθορισμός απαιτήσεων

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

Στάδια αντιμετώπισης ενός προβλήματος

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

Κεφάλαια 2, 7 και 8

Αλγόριθμος

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

Κριτήρια αλγορίθμου

Είσοδος (input). Καμία, μία ή περισσότερες τιμές πρέπει να δίνονται ως είσοδοι στον αλγόριθμο.

Έξοδος (output). Ο αλγόριθμος πρέπει να δημιουργεί τουλάχιστον μια τιμή δεδομένων ως αποτέλεσμα προς το χρήστη ή προς έναν άλλο αλγόριθμο.

Καθοριστικότητα (definiteness). Κάθε εντολή πρέπει να καθορίζεται χωρίς καμία αμφιβολία για τον τρόπο εκτέλεσής της.

Περατότητα (finiteness). Ο αλγόριθμος πρέπει να τελειώνει μετά από πεπερασμένα βήματα εκτέλεσης των εντολών του.

Αποτελεσματικότητα (effectiveness). Κάθε μεμονωμένη εντολή του αλγορίθμου πρέπει να είναι απλή. Μια εντολή, δηλαδή, δεν αρκεί να έχει ορισθεί αλλά πρέπει να είναι και εκτελέσιμη.

Σπουδαιότητα αλγορίθμων

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

- **Υλικού**

  Η ταχύτητα ενός αλγορίθμου επηρεάζεται από την τεχνολογία του υλικού.

- **Γλωσσών Προγραμματισμού**

  Το είδος της γλώσσας προγραμματισμού αλλάζει τη δομή και τον αριθμό των εντολών ενός αλγορίθμου.

  Μια γλώσσα χαμηλότερου επιπέδου είναι ταχύτερη από μια γλώσσα υψηλότερου επιπέδου.

- **Θεωρητική**

  Το ερώτημα, αν υπάρχει ή όχι, αποδοτικός αλγόριθμος για την επίλυση ενός προβλήματος, είναι σημαντικό γιατί προσδιορίζει τα όρια της λύσης που θα βρεθεί σε σχέση με ένα συγκεκριμένο πρόβλημα.

- **Αναλυτική**

  Μελετώνται οι υπολογιστικοί πόροι (computer resources) που απαιτούνται από έναν αλγόριθμο, όπως για παράδειγμα η μνήμη.

Περιγραφή και αναπαράσταση των αλγορίθμων

- **Ελεύθερο κείμενο**

  Αποτελεί τον πιο ανεπεξέργαστο και αδόμητο τρόπο παρουσίασης αλγορίθμου. Απαιτεί προσοχή γιατί μπορεί να παραβιαστεί το κριτήριο της αποτελεσματικότητας.

- **Διαγραμματικές τεχνικές**

  Συνιστούν γραφικό τρόπο παρουσίασης του αλγορίθμου. Η πιο παλιά και πλέον γνωστή είναι τα διαγράμματα ροής. Δεν χρησιμοποιούνται συχνά.

- **Φυσική γλώσσα κατά βήματα**

  Η τεχνική αυτή απαιτεί προσοχή γιατί μπορεί εύκολα να παραβιαστεί το κριτήριο του καθορισμού.

- **Κωδικοποίηση**

  Με πρόγραμμα γραμμένο σε ψευδογλώσσα ή σε γλώσσα προγραμματισμού που όταν θα εκτελεσθεί θα δώσει τα ίδια αποτελέσματα με τον αλγόριθμο.

Βασικές συνιστώσες – εντολές ενός αλγορίθμου

Τύποι Δεδομένων

- **Ακέραιος**

  Ο τύπος αυτός περιλαμβάνει τους ακέραιους που είναι γνωστοί από τα μαθηματικά. Οι ακέραιοι μπορεί να είναι θετικοί, αρνητικοί ή μηδέν και δεν έχουν δεκαδικά ψηφία.

- **Πραγματικός**

  Ο τύπος αυτός περιλαμβάνει τους πραγματικούς αριθμούς που γνωρίζουμε από τα μαθηματικά. Οι πραγματικοί αριθμοί μπορεί να είναι θετικοί, αρνητικοί ή μηδέν και έχουν δεκαδικά ψηφία.

- **Χαρακτήρας (αλφαριθμητικός)**

  Τα δεδομένα αυτού του τύπου ονομάζονται αλφαριθμητικά, επειδή περιέχουν τόσο αλφαβητικούς όσο και αριθμητικούς χαρακτήρες. Ο τύπος αυτός αναφέρεται τόσο σε έναν όσο και σε μια σειρά χαρακτήρων.

  Οι χαρακτήρες βρίσκονται υποχρεωτικά μέσα σε εισαγωγικά.

  Στην ψευδογλώσσα χρησιμοποιούνται τα διπλά εισαγωγικά, "Μέσος Όρος", ενώ στη γλώσσα προγραμματισμού χρησιμοποιούνται τα απλά εισαγωγικά 'Μέσος Όρος'. Στις εξετάσεις δεν έχει σημασία αν θα χρησιμοποιηθούν μονά ή διπλά εισαγωγικά.

- **Λογικός**

  Ο τύπος αυτός δέχεται μόνο δύο τιμές, **Αληθής** και **Ψευδής**, οι οποίες αντιπροσωπεύουν αληθείς ή ψευδείς συνθήκες.

Σταθερές

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

- **Αριθμητικές** (ακέραιες ή πραγματικές): 123, +5, -1,25
- **Αλφαριθμητικές**: ''τιμή'', ''Κατάσταση αποτελεσμάτων'', ''Κ'', ''1'', ''Αληθής'', ''Διάβασε''
- **Λογικές**: Αληθής, Ψευδής

  Στις λογικές τιμές δεν ορίζεται η διάταξη, δηλαδή οι λογικές τιμές δεν μπορούν να συγκριθούν μόνο ως προς ισότητα, είτε ίσο (=) είτε διάφορο (<>).

Σημείωση: Στη γλώσσα προγραμματισμού θα συναντήσουμε και τις **συμβολικές σταθερές**, οι οποίες αντιστοιχούν ονόματα σε σταθερές τιμές. Οι συμβολικές σταθερές δηλώνονται στην αρχή ενός προγράμματος (στο τμήμα δήλωσης σταθερών).

Παράδειγμα: ΦΠΑ = 0.19, ΠΙ = 3.14159

Μεταβλητές

Μια μεταβλητή είναι ένα γλωσσικό αντικείμενο, που χρησιμοποιείται, για να παραστήσει ένα στοιχείο δεδομένου.

Στη μεταβλητή εκχωρείται μια **τιμή που μπορεί να αλλάζει** κατά τη διάρκεια εκτέλεσης ενός αλγορίθμου.

Ανάλογα με το είδος της τιμής που μπορούν να λάβουν, οι μεταβλητές διακρίνονται σε αριθμητικές (ακέραιες ή πραγματικές), αλφαριθμητικές και λογικές.

- **Προσοχή**: Ο τύπος της μεταβλητής δεν αλλάζει ποτέ κατά την εκτέλεση ενός αλγορίθμου ή προγράμματος. Η τιμή της μόνο μπορεί να αλλάζει.

Τα ονόματα των μεταβλητών, μπορεί να περιέχουν γράμματα πεζά ή κεφαλαία του ελληνικού (Α-Ω, α-ω) ή του λατινικού (A-Z, a-z) αλφαβήτου, ψηφία (0-9) και τον χαρακτήρα κάτω παύλα (_), ενώ πρέπει **υποχρεωτικά να αρχίζουν με γράμμα**.

- **Προσοχή**: Επειδή μερικές λέξεις χρησιμοποιούνται για συγκεκριμένους λόγους, όπως για παράδειγμα οι λέξεις Αλγόριθμος, Διάβασε, αυτές οι λέξεις **δεν μπορούν να χρησιμοποιηθούν** για ονόματα μεταβλητών. Οι λέξεις αυτές αποκαλούνται **δεσμευμένες**.

Αποδεκτά ονόματα μεταβλητών είναι: Α, Όνομα, Τιμή, Τυπική_Απόκλιση, Α100, ΦΠΑ.

Μη αποδεκτά ονόματα μεταβλητών είναι: 100Α, Μέση Τιμή, Κόστος$, ΑΝ.

Τελεστές

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

Αριθμητικοί

δύναμη	^	x^2
διαίρεση	/	8/2 = 4,0
πολλαπλασιασμός	*	
πηλίκο ακέραιας διαίρεσης	DIV	7 DIV 2 = 3
υπόλοιπο ακέραιας διαίρεσης	MOD	7 MOD 2 = 1
πρόσθεση	+	
αφαίρεση	–	

Λογικοί

Οι λογικοί τελεστές, κατά σειρά προτεραιότητας των πράξεων, είναι οι ακόλουθοι τρεις:

Όχι	Άρνηση
Και	Σύζευξη
Η	Διάζευξη

Οι τρεις πράξεις ορίζονται με τον πίνακα αλήθειας:

Πρόταση Α	**Πρόταση Β**	**Α ΚΑΙ Β**	**Α Ή Β**	**ΌΧΙ Α**
Αληθής	Αληθής	Αληθής	Αληθής	Ψευδής
Αληθής	Ψευδής	Ψευδής	Αληθής	Ψευδής
Ψευδής	Αληθής	Ψευδής	Αληθής	Αληθής
Ψευδής	Ψευδής	Ψευδής	Ψευδής	Αληθής

- **Προσοχή**: Εύκολα παρασύρεται κανείς με το Όχι Α.

  Παράδειγμα: **Όχι (x>0)** είναι ισοδύναμο με το **x<=0** (και όχι μόνο το x<0).

  Παράδειγμα: **Όχι (x>0 Και x<20)** είναι ισοδύναμο με το **(x<=0) Ή (x>=20).**

Συγκριτικοί

Ίσο	=	Διάφορο	<>
Μεγαλύτερο	>	Μεγαλύτερο ίσο	>=
Μικρότερο	<	Μικρότερο ίσο	<=

Εκφράσεις

Οι εκφράσεις διαμορφώνονται από τους τελεστέους που είναι σταθερές και μεταβλητές, και από τους τελεστές. Μια έκφραση μπορεί να αποτελείται από μια μόνο μεταβλητή ή σταθερά, μέχρι μια πολύπλοκη μαθηματική έκφραση.

Παραδείγματα εκφράσεων

- X
- 2
- X^2 – T_P(Y/4) * Z MOD 3

Προτεραιότητα πράξεων

Αριθμητικές πράξεις

Προηγούνται οι πράξεις μέσα σε παρενθέσεις και ακολουθούν: ύψωση σε δύναμη (^), πολλαπλασιασμός και διαίρεση (*, /, DIV, MOD), πρόσθεση και αφαίρεση (+, -).

Σε περίπτωση ισοδυνάμων πράξεων, οι πράξεις εκτελούνται από αριστερά προς τα δεξιά.

Παράδειγμα: 6/2*5 = **3** * 5 = 15

Προσοχή στη γραφή κλασμάτων σε γλώσσα προγραμματισμού. Να χρησιμοποιείτε πάντα παρενθέσεις στον αριθμητή και στον παρονομαστή, ειδικά όταν είναι εκφράσεις.

Έκφραση	**Σωστό**	**Λάθος**	
$\frac{x+2}{y}$	(x+2)/y	x+2/y	$x+\frac{2}{y}$
$\frac{x}{2\,y}$	x/(2*y)	x/2*y	$\frac{x}{x}\,y$

Λογικές πράξεις

Προηγούνται οι πράξεις μέσα σε παρενθέσεις και ακολουθούν: η άρνηση (Όχι), η σύζευξη (Και) και η διάζευξη (Η).

Παράδειγμα:

Αληθής **Και Όχι** Αληθής **Η** Αληθής ⇔ (Αληθής **Και** (**Όχι** Αληθής)) **Η** Αληθής

Προτεραιότητα των πράξεων

1. Αριθμητικές πράξεις,
2. Συγκρίσεις
3. Λογικές πράξεις.

Παράδειγμα: έστω X = 3, Y = 10, Z = 2 και Δ = Αληθής

Υ div Χ = (Χ + Ζ) mod 3 Και Όχι (Δ Η Χ*Ζ < 10)

1. Αρχικά αντικαθιστούμε τις τιμές στις μεταβλητές:

10 div **3** = (**3** + **2**) mod 3 Και Όχι (**Αληθής** Η **3*2** < 10)

Έπειτα εκτελούμε τις πράξεις

2. Πρώτα εκτελούνται οι αριθμητικές πράξεις.

10 **div** 3 = (3 **+** 2) **mod** 3 Και Όχι (Αληθής Η 3*****2 < 10)

3 = 5 **mod** 3 Και Όχι (Αληθής Η 6 < 10)

3 = 2 Και Όχι (Αληθής Η 6 < 10)

3. Έπειτα εκτελούνται οι συγκρίσεις

3 **=** 2 Και Όχι (Αληθής Η 6 **<** 10)

Ψευδής Και Όχι (Αληθής Η Αληθής)

4. Τελευταίες εκτελούνται οι λογικές πράξεις

Ψευδής **Και Όχι** (Αληθής **Η** Αληθής)

Ψευδής **Και Όχι** (Αληθής)

Ψευδής **Και** Ψευδής

Ψευδής

Κανόνες εμφωλευμένων βρόχων (επαναλήψεων)

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

Ρώσικος Πολλαπλασιασμός

Η μέθοδος αυτή χρησιμοποιείται πρακτικά στους υπολογιστές, γιατί απαιτεί πολλαπλασιασμό επί δύο, διαίρεση διά δύο και πρόσθεση.

Ο πολλαπλασιασμός επί δύο και η διαίρεση διά δύο μπορούν να υλοποιηθούν ταχύτατα με μία απλή εντολή ολίσθησης, σε αντίθεση με τον πολλαπλασιασμό με οποιοδήποτε ακέραιο που θεωρείται πιο χρονοβόρα διαδικασία.

Η ολίσθηση προς τα αριστερά μετατοπίζει όλα τα ψηφία ενός δυαδικού αριθμού μία θέση προς τα αριστερά, εισάγοντας ένα μηδέν στο τέλος του αριθμού κι έχει ως αποτέλεσμα τον πολλαπλασιασμό του αριθμού επί δύο.

Η ολίσθηση προς τα δεξιά μετατοπίζει όλα τα ψηφία ενός δυαδικού αριθμού μία θέση προς τα δεξιά, αποκόπτοντας το τελευταίο ψηφίο και εισάγοντας ένα μηδέν στην αρχή του αριθμού, με αποτέλεσμα τη διαίρεση του αριθμού με το δύο.

Αλγόριθμος σε Γλώσσα με βήματα

Αλγόριθμος: Πολλαπλασιασμός δύο θετικών ακεραίων (αλά ρωσικά)	
Είσοδος:	Δύο ακέραιοι Μ1 και Μ2, όπου Μ1, Μ2 > 1
Έξοδος:	Το γινόμενο Ρ=Μ1*Μ2
Βήμα 1	Θέσε Ρ=0
Βήμα 2	Αν Μ2>0, τότε πήγαινε στο Βήμα 3, αλλιώς πήγαινε στο Βήμα 7
Βήμα 3	Αν ο Μ2 είναι περιττός, τότε θέσε Ρ=Ρ+Μ1
Βήμα 4	Θέσε Μ1=Μ1*2
Βήμα 5	Θέσε Μ2=Μ2/2 (θεώρησε μόνο το ακέραιο μέρος)
Βήμα 6	Πήγαινε στο Βήμα 2
Βήμα 7	Τύπωσε τον Ρ.

Αλγόριθμος σε ψευδογλώσσα

```
Αλγόριθμος Πολλαπλασιασμός_αλά_Ρωσικά
Δεδομένα // Μ1, Μ2 //
Ρ ← 0
Όσο Μ2 > 0 Επανάλαβε
    Αν Μ2 mod 2 < > 0 Τότε
        Ρ ← Ρ + Μ1
    Τέλος_Αν
    Μ1 ← 2 * Μ1
    Μ2 ← Μ2 div 2
Τέλος_Επανάληψης
Αποτελέσματα // Ρ //
Τέλος Πολλαπλασιασμός_αλά_Ρωσικά
```

Κεφάλαιο 3

Δεδομένα

Τα δεδομένα είναι η **αφαιρετική αναπαράσταση της πραγματικότητας** και συνεπώς μια απλοποιημένη όψη της. Είναι ακατέργαστα γεγονότα και κάθε φορά η επιλογή τους εξαρτάται από τον τύπο του προβλήματος.

Η συλλογή των ακατέργαστων δεδομένων και ο συσχετισμός τους δίνει ως αποτέλεσμα την πληροφορία.

Ο αλγόριθμος είναι το μέσο παραγωγής πληροφορίας από δεδομένα.

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.

Η Πληροφορική ορίζεται ως η επιστήμη που μελετά τα δεδομένα από τις ακόλουθες σκοπιές:

1. **Υλικού**

   Το υλικό επιτρέπει τα δεδομένα ενός προγράμματος να αποθηκεύονται στην κύρια μνήμη και στις περιφερειακές συσκευές του υπολογιστή με διάφορες αναπαραστάσεις.

   Τέτοιες είναι η δυαδική, ο κώδικας ASCII, ο κώδικας EBCDIC, το συμπλήρωμα του 1 ή του 2, κλπ.

2. **Γλωσσών Προγραμματισμού**

   Οι γλώσσες προγραμματισμού υψηλού επιπέδου επιτρέπουν τη χρήση διαφόρων τύπων μεταβλητών για να περιγράψουν ένα δεδομένο.

   Ο μεταγλωττιστής κάθε γλώσσας φροντίζει για την αποδοτικότερη μορφή αποθήκευσης από πλευράς υλικού κάθε μεταβλητής στον υπολογιστή.

3. **Δομών Δεδομένων**

   Δομή δεδομένων είναι ένα σύνολο δεδομένων μαζί με ένα σύνολο επιτρεπτών λειτουργιών σε αυτές. Μια τέτοια δομή είναι η **εγγραφή** που αποτελείται από **πεδία** που αποθηκεύουν **χαρακτηριστικά** διαφορετικού τύπου.

   Μια άλλη μορφή δεδομένων είναι το **αρχείο** που αποτελείται από ένα σύνολο εγγραφών. Μια επιτρεπτή λειτουργία σε ένα αρχείο είναι η **σειριακή προσπέλασή** όλων των εγγραφών του.

4. **Ανάλυσης Δεδομένων**

   Τρόποι καταγραφής και αλληλοσυσχέτισης των δεδομένων μελετώνται έτσι ώστε να αναπαρασταθεί η γνώση για πραγματικά γεγονότα.

   Οι τεχνολογίες των **Βάσεων Δεδομένων**, της **Μοντελοποίησης Δεδομένων** και της **Αναπαράστασης Γνώσης** ανήκουν σε αυτή τη σκοπιά μελέτης των δεδομένων.

Αλγόριθμοι και Δομές Δεδομένων

Τα δεδομένα ενός προβλήματος αποθηκεύονται στον υπολογιστή είτε στην κύρια μνήμη είτε στη δευτερεύουσα μνήμη του. Η αποθήκευση δε γίνεται με τυχαίο τρόπο αλλά συστηματικά, δηλαδή χρησιμοποιώντας μια δομή.

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

Κάθε μορφή δομής δεδομένων αποτελείται από ένα σύνολο **κόμβων**. Οι βασικές λειτουργίες των δομών δεδομένων είναι οι ακόλουθες:

- **Προσπέλαση**, πρόσβαση σε έναν κόμβο με σκοπό να εξετασθεί ή να τροποποιηθεί το περιεχόμενό του.
- **Εισαγωγή**, δηλαδή η προσθήκη νέων κόμβων σε μια υπάρχουσα δομή.
- **Διαγραφή**, δηλαδή η αφαίρεση ενός κόμβου από μία δομή. Αποτελεί το αντίστροφο της εισαγωγής.
- **Αναζήτηση**, κατά την οποία προσπελαύνονται οι κόμβοι μιας δομής προκειμένου να εντοπιστούν ένας ή περισσότεροι κόμβοι που έχουν μια συγκεκριμένη ιδιότητα.
- **Ταξινόμηση**, όπου οι κόμβοι μιας δομής διατάσσονται κατά άυξουσα ή φθίνουσα σειρά.
- **Αντιγραφή**, κατά την οποία όλοι οι κόμβοι ή μερικοί από τους κόμβους μιας δομής αντιγράφονται σε μια άλλη δομή.
- Συγχώνευση, κατά την οποία δύο ή περισσότερες δομές συνενώνονται σε μια ενιαία δομή.
- **Διαχωρισμός**, κατά τον οποίο μία ενιαία δομή διαχωρίζεται σε δύο ή περισσότερες δομές. Αποτελεί την αντίστροφη πράξη της συγχώνευσης.

Στην πράξη σπάνια χρησιμοποιούνται και οι οκτώ λειτουργίες για κάποια δομή.

Συνήθως μια δομή δεδομένων είναι αποδοτικότερη από κάποια άλλη δομή με κριτήριο κάποια λειτουργία.

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

Αλγόριθμοι + Δομές Δεδομένων = Προγράμματα, *(Niklaus Wirth 1976)*

Στατικές και Δυναμικές Δομές Δεδομένων

Οι δομές δεδομένων διακρίνονται σε δύο μεγάλες κατηγορίες: τις στατικές και τις δυναμικές.

1. **Οι δυναμικές δομές δεδομένων** στηρίζονται στην τεχνική της **δυναμικής παραχώρησης μνήμης**.

   α. Δεν έχουν σταθερό μέγεθος, αλλά ο αριθμός (το πλήθος) των κόμβων τους (των στοιχείων που περιέχουν) μεγαλώνει και μικραίνει καθώς στη δομή εισάγονται ή διαγράφονται κάποια δεδομένα.

   β. Τα στοιχεία των δυναμικών δομών δεδομένων δεν αποθηκεύονται σε συνεχόμενες θέσεις μνήμης.

   Σημείωση: Το μέγεθος των δυναμικών δομών δεδομένων καθορίζεται δυναμικά κατά την εκτέλεση του προγράμματος και δεν είναι γνωστό τη στιγμή του προγραμματισμού. Για το λόγο αυτό ο μεταγλωττιστής δεν μπορεί να τοποθετήσει τα στοιχεία μιας δυναμικής δομής σε συνεχόμενες θέσεις μνήμης.

2. **Στις στατικές δομές δεδομένων**

   α. το ακριβές μέγεθος της απαιτουμένης κύριας μνήμης καθορίζεται κατά τη στιγμή του προγραμματισμού τους, δηλαδή κατά τη μεταγλώττιση (μετάφραση) του προγράμματος.

   Το ακριβές μέγεθος της απαιτούμενης μνήμης των στατικών δομών δεδομένων δεν καθορίζεται και δεν μπορεί να αλλάξει κατά τη στιγμή της εκτέλεσης του προγράμματος.

   β. Τα στοιχεία των στατικών δεδομένων αποθηκεύονται σε συνεχόμενες θέσεις μνήμης.

   Στην πράξη οι στατικές δομές δεδομένων υλοποιούνται με πίνακες.

   Οι πίνακες χρησιμεύουν για την αποθήκευση και διαχείριση δύο σπουδαίων δομών, της στοίβας και της ουράς.

Σειριακή Αναζήτηση

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

1. ο πίνακας είναι μη ταξινομημένος,
2. ο πίνακας είναι μικρού μεγέθους (για παράδειγμα μέχρι 20 στοιχείων),
3. η αναζήτηση σε ένα συγκεκριμένο πίνακα γίνεται σπάνια,

Ταξινόμηση

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

Ορισμός

Δοθέντων των στοιχείων a_1, a_2, …, a_n η ταξινόμηση συνίσταται στη μετάθεση της θέσης των στοιχείων, ώστε να τοποθετηθούν σε μία σειρά a_{k1}, a_{k2}, …,a_{kn} έτσι ώστε, δοθείσης μίας συνάρτησης διάταξης f, να ισχύει: $f(a_{k1}) \leq f(a_{k2}) \leq \ldots \leq f(a_{kn})$

Δομές δεδομένων δευτερεύουσας μνήμης

- Το μέγεθος της κύριας μνήμης δεν επαρκεί για την αποθήκευση των δεδομένων.
- Χρησιμοποιούνται ειδικές δομές για την αποθήκευση των δεδομένων στη δευτερεύουσα μνήμη, δηλαδή κυρίως στον μαγνητικό δίσκο.
- Οι ειδικές αυτές δομές ονομάζονται **αρχεία**.
  - Στην περίπτωση του δίσκου, τα δεδομένα δεν χάνονται.
  - 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
- Κάθε εγγραφή αποτελείται:
  - Από ένα ή περισσότερα πεδία που ταυτοποιούν την εγγραφή (κλειδιά)
    - Τα κλειδιά διακρίνονται σε πρωτεύοντα και δευτερεύοντα.
  - Από άλλα πεδία που περιγράφουν διάφορα χαρακτηριστικά της εγγραφής.

Κεφάλαιο 4

Ανάλυση προβλημάτων

Είναι πιθανόν ένα πρόβλημα να μην επιλύεται με μία μόνο λύση αλλά με περισσότερες.

Η ανάλυση ενός προβλήματος σε ένα σύγχρονο υπολογιστικό περιβάλλον περιλαμβάνει:

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

και στη συνέχεια:

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

Κατά την ανάλυση ενός προβλήματος θα πρέπει να δοθεί απάντηση στις ερωτήσεις:

1. Ποια είναι τα δεδομένα και το μέγεθος του προβλήματος;
2. Ποιες είναι οι συνθήκες που πρέπει να πληρούνται για την επίλυση του προβλήματος;
3. Ποια είναι η πλέον αποδοτική μέθοδος επίλυσής τους (σχεδίαση αλγορίθμου);
4. Πώς θα καταγραφεί η λύση σε ένα πρόβλημα (π.χ. σε ψευδογλώσσα);
5. Ποιος είναι ο τρόπος υλοποίησης στο συγκεκριμένο υπολογιστικό σύστημα (π.χ. επιλογή γλώσσας προγραμματισμού);

Η ανάλυση κάθε προβλήματος είναι απαραίτητη, έτσι ώστε:

- να αναζητηθεί η πλέον κατάλληλη μέθοδος
- που να παρέχει τη ζητούμενη λύση,
- όσο γίνεται ταχύτερα
- και με το λιγότερο δυνατό κόστος σε υπολογιστικούς πόρους.

Υπάρχουν «συγγενή» προβλήματα, δηλαδή προβλήματα που μπορούν να αναλυθούν με παρόμοιο τρόπο και να αντιμετωπισθούν με αντίστοιχες μεθόδους και τεχνικές.

Οι μέθοδοι ανάλυσης και επίλυσης των προβλημάτων παρουσιάζουν ιδιαίτερο ενδιαφέρον για τους εξής λόγους:

- παρέχουν ένα γενικό πρότυπο κατάλληλο για την επίλυση προβλημάτων ευρείας κλίμακας,
- μπορούν να αναπαρασταθούν με κοινές δομές δεδομένων και ελέγχου,
- παρέχουν τη δυνατότητα καταγραφής των χρονικών και «χωρικών» απαιτήσεων της μεθόδου επίλυσης, έτσι ώστε να μπορεί να γίνει επακριβής εκτίμηση των αποτελεσμάτων.

Διαίρει και Βασίλευε

- Αποτελεί μια μέθοδο σχεδίασης αλγορίθμων
- Εντάσσονται τεχνικές που υποδιαιρούν ένα πρόβλημα σε μικρότερα υποπροβλήματα

- Αυτά έχουν την ίδια τυποποίηση με το αρχικό πρόβλημα
- Είναι, όμως, μικρότερα σε μέγεθος

Η επίλυση ενός προβλήματος έγκειται στη σταδιακή επίλυση των όσο δυνατόν μικρότερων υποπροβλημάτων, ώστε να προκύψει η συνολική λύση του ευρύτερου προβλήματος.

Τα βήματα με τα οποία μπορεί να αποδοθεί η μέθοδος «Διαίρει και Βασίλευε» είναι:

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 υπο-στιγμιότυπο.
4. Συνδυάζονται όλες οι μερικές λύσεις που βρέθηκαν για τα υπο-στιγμιότυπα, έτσι ώστε να δοθεί η τελική λύση του προβλήματος.

Η έννοια του προγράμματος

Συνοπτική αναφορά βημάτων επίλυσης ενός προβλήματος:

1. Ακριβής προσδιορισμός του προβλήματος
2. Ανάπτυξη αντίστοιχου αλγορίθμου
3. Τροποποίηση/ διατύπωση αλγορίθμου σε μορφή κατανοητή από τον υπολογιστή

Ο προγραμματισμός ασχολείται με το τελευταίο στάδιο επίλυσης προβλημάτων.

Προγραμματισμός είναι η διαδικασία κατά την οποία ένα σύνολο εντολών δίνεται στον υπολογιστή προκειμένου να επιλυθεί ένα πρόβλημα με βάση τον αλγόριθμο που έχει ήδη αναπτυχθεί.

Φυσικές και τεχνητές γλώσσες

- Οι **φυσικές γλώσσες** χρησιμοποιούνται για την επικοινωνία μεταξύ των ανθρώπων ενώ,
- Οι **τεχνητές γλώσσες** χρησιμοποιούνται για την επικοινωνία μεταξύ ανθρώπου και μηχανής. Σε αυτή την κατηγορία ανήκουν οι γλώσσες προγραμματισμού αλλά και αυτές ακολουθούν τις ίδιες αρχές και έννοιες της επιστήμης που μελετά τις φυσικές γλώσσες, της γλωσσολογίας.

Τα στοιχεία τα οποία προσδιορίζουν μία γλώσσα

- **Αλφάβητο**

  Το σύνολο των στοιχείων που χρησιμοποιεί η γλώσσα

- **Λεξιλόγιο**

  Το υποσύνολο των ακολουθιών που προκύπτουν από τα στοιχεία του αλφαβήτου και σχηματίζουν τις λέξεις που είναι αποδεκτές από την γλώσσα

- **Γραμματική**
  - **Τυπικό ή τυπολογικό**

    Το σύνολο των κανόνων που καθορίζουν τις μορφές με τις οποίες μία λέξη είναι αποδεκτή

  - **Συντακτικό**

    Το σύνολο των κανόνων που καθορίζουν τη νομιμότητα της διάταξης και σύνδεσης των λέξεων, προκειμένου να δημιουργηθούν προτάσεις

- **Σημασιολογία**

  Το σύνολο των κανόνων που καθορίζουν την εννοιολογική σημασία των λέξεων της γλώσσας, καθώς και των εκφράσεων και προτάσεων που δημιουργούνται από το συνδυασμό των λέξεων αυτών

Κεφάλαιο 6

Διαφορές μεταξύ Φυσικών και τεχνητών γλωσσών

Οι **φυσικές εξελίσσονται** διαρκώς και προσαρμόζονται στα νέα δεδομένα κάθε εποχής και κοινωνίας

Αντίθετα, οι **τεχνητές** γλώσσες, επειδή ο σκοπός κατασκευής τους είναι συγκεκριμένος, είναι **στάσιμες**.

Στις γλώσσες προγραμματισμού προκύπτουν συνεχώς νέες, βελτιωμένες εκδόσεις τους, ώστε:

- να διορθωθούν λάθη παλαιότερων εκδόσεών
- να καλυφθεί μεγαλύτερο εύρος εφαρμογών, μια και οι ανάγκες είναι ολοένα αυξανόμενες.

Αυτή η μεταβολή γίνεται:

- είτε σε επίπεδο διαλέκτου οπότε αλλάζει το λεξιλόγιο
- είτε σε επίπεδο επέκτασης οπότε η νέα έκδοση έχει τα στοιχεία της προηγούμενης αλλά και αρκετά νέα.

Τεχνικές σχεδίασης προγραμμάτων

Αναπτύσσονται διαρκώς μεθοδολογίες και τεχνικές προγραμματισμού, εύκολες στην κατανόηση και την υλοποίηση. Οι βασικότερες από αυτές είναι:

Ιεραρχική Σχεδίαση Προγράμματος

Είναι η τεχνική που βασίζεται στη συνεχή διαίρεση ενός προβλήματος σε υποπροβλήματα, τα οποία επιλύονται εύκολα και οδηγούν στην επίλυση του αρχικού προβλήματος.

Ονομάζεται και σχεδίαση «από πάνω προς τα κάτω» διότι περιλαμβάνει τον καθορισμό των βασικών λειτουργιών σε ανώτερο επίπεδο και στη συνέχεια τη διάσπαση των λειτουργιών αυτών σε ολοένα μικρότερες, άρα απλούστερες, μέχρι και το τελευταίο επίπεδο, στο οποίο οι λειτουργίες είναι πολύ απλές άρα και εύκολα υλοποιήσιμες.

Τμηματικός προγραμματισμός

Αποτελεί την υλοποίηση της ιεραρχικής σχεδίασης: Μετά την ιεραρχική σχεδίαση, κάθε υποπρόβλημα αποτελεί ανεξάρτητη ενότητα, άρα και η επίλυσή του πραγματοποιείται σε ξεχωριστό τμήμα του προγράμματος

Πλεονεκτήματα

1. Διευκόλυνση στην υλοποίηση(δημιουργία)
2. Μείωση λαθών

3. Συνεχής παρακολούθηση και,
4. Δυνατότητα κατανόησης και συντήρησης του προγράμματος από τρίτους

Δομημένος προγραμματισμός

Ο λόγος ανάπτυξης του δομημένου προγραμματισμού ήταν η ανάγκη ύπαρξης μίας μεθοδολογίας κοινής για την ανάπτυξη των προγραμμάτων αλλά και η ανάγκη περιορισμού της ανεξέλεγκτης χρήσης της εντολής GOTO στα προγράμματα.

Ο δομημένος προγραμματισμός αποτελεί μία μεθοδολογία σύνταξης προγραμμάτων που αποσκοπεί:

- στο να βοηθήσει τον προγραμματιστή στην ανάπτυξη σύνθετων προγραμμάτων,
- στη μείωση των λαθών,
- στη μεγαλύτερη κατανόηση των προγραμμάτων και
- στην ευ μεταβλητότητα των προγραμμάτων (ευκολία στις διορθώσεις και αλλαγές)

Βασικά Χαρακτηριστικά

1. Στηρίζεται στη χρήση τριών δομών: **Ακολουθίας, Επιλογής και, Επανάληψης**
2. Όλα τα προγράμματα μπορούν να υλοποιηθούν χρησιμοποιώντας μόνο αυτές τις δομές ή συνδυασμούς τους.
3. Κάθε πρόγραμμα(και κάθε τμήμα ή ενότητα προγράμματος) έχει μόνο μία είσοδο και μία έξοδο.

Σήμερα, έχει επικρατήσει απόλυτα και αποτελεί τη βασική μεθοδολογία των περισσότερων γλωσσών προγραμματισμού. Βοηθά την ανάλυση ενός προγράμματος σε υποπρογράμματα γι’ αυτό και περιλαμβάνει τόσο την ιεραρχική σχεδίαση όσο και τον τμηματικό προγραμματισμό.

Πλεονεκτήματα του δομημένου προγραμματισμού:

1. Δημιουργία απλούστερων προγραμμάτων
2. Άμεση μεταφορά αλγορίθμων σε προγράμματα
3. Διευκόλυνση ανάλυσης (διαίρεσης) κάθε προγράμματος σε τμήματα (υποπρογράμματα)
4. Μείωση λαθών
5. Διευκόλυνση στην ανάγνωση και κατανόηση ενός προγράμματος από τρίτους
6. Ευκολία στη διόρθωση και συντήρηση προγραμμάτων

Αντικειμενοστραφής Προγραμματισμός

Ένα πρόγραμμα περιγράφει «ενέργειες» που εφαρμόζονται πάνω σε δεδομένα και τα επεξεργάζονται.

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

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

Ο αντικειμενοστραφής προγραμματισμός χρησιμοποιεί την ιεραρχική σχεδίαση, τον τμηματικό προγραμματισμό και ακολουθεί τις αρχές του δομημένου προγραμματισμού.

Προγραμματιστικά περιβάλλοντα

Δύο είναι τα βασικά μεταφραστικά προγράμματα που χρησιμοποιούνται για τη μετάφραση ενός προγράμματος σε γλώσσα μηχανής προκειμένου αυτό να γίνει κατανοητό -κατά συνέπεια- εκτελέσιμο από τον υπολογιστή. **Μεταγλωττιστές** και **διερμηνευτές.**

- **Μεταγλωττιστής**

  Μεταφραστικό πρόγραμμα το οποίο δέχεται σαν είσοδο ένα πρόγραμμα σε γλώσσα υψηλού επιπέδου και παράγει σαν έξοδο ένα ισοδύναμο πρόγραμμα σε γλώσσα μηχανής, το οποίο είναι ανεξάρτητο από το αρχικό και εκτελέσιμο ανά πάσα στιγμή.

- **Διερμηνευτής**

  Μεταφραστικό πρόγραμμα το οποίο διαβάζει μία προς μία τις εντολές του αρχικού προγράμματος σε γλώσσα υψηλού επιπέδου και εκτελεί για καθεμία μία ισοδύναμη ακολουθία εντολών σε γλώσσα μηχανής

Έννοιες

- **Πηγαίο πρόγραμμα**

  Είναι το αρχικό πρόγραμμα σε γλώσσα υψηλού επιπέδου ενώ,

- **Αντικείμενο πρόγραμμα**

  Είναι το πρόγραμμα που παράγεται από το μεταγλωττιστή.

- **Εκτελέσιμο πρόγραμμα**

  Το τελικό πρόγραμμα που εκτελείται από τον υπολογιστή.

- **Συντάκτης**

  Το πρόγραμμα που δημιουργεί το περιβάλλον στο οποίο γίνεται η σύνταξη του αρχικού προγράμματος και η διόρθωσή του.

- **Συνδέτης – Φορτωτής**

  Το πρόγραμμα που πραγματοποιεί τη σύνδεση που απαιτείται να γίνει ανάμεσα στο αντικείμενο πρόγραμμα και σε άλλα τμήματα προγράμματος, τα οποία είτε συντάσσονται από τον προγραμματιστή είτε είναι αποθηκευμένα σε βιβλιοθήκες της γλώσσας με σκοπό την παραγωγή του τελικού προγράμματος που εκτελείται από τον υπολογιστή και ονομάζεται εκτελέσιμο πρόγραμμα.

- **Μεταγλώττιση και σύνδεση**

  Όλη η προαναφερόμενη διαδικασία μετάφρασης και εκτέλεσης του προγράμματος.

Λάθη[1]

Για να εκτελεστεί τελικά κάποιο πρόγραμμα απαραίτητη προϋπόθεση είναι η ανυπαρξία λαθών στο πηγαίο πρόγραμμα. Τα λάθη που μπορεί να παρουσιαστούν είναι δύο τύπων:

- **Συντακτικά (ή λάθη κατά την υλοποίηση)**

  Εμφανίζονται κατά τη μεταγλώττιση. Οφείλονται σε αναγραμματισμούς ονομάτων εντολών, ορθογραφικά λάθη, παράλειψη δήλωσης χρησιμοποιούμενων μεγεθών. Αν αυτά δεν διορθωθούν είναι αδύνατο να παραχθεί το εκτελέσιμο πρόγραμμα.

  Το μεταφραστικό πρόγραμμα(μεταγλωττιστής ή διερμηνευτής) πραγματοποιεί την ανίχνευση των λαθών και εμφανίζει κατάλληλα διαγνωστικά μηνύματα ως προς το είδος του λάθους, τη γραμμή εντολών στην οποία έχει γίνει το λάθος.

- **Λογικά (παράγουν λανθασμένα αποτελέσματα)**

  Εμφανίζονται κατά την εκτέλεση του προγράμματος. Παρόλο που το πρόγραμμα θα «τρέξει» κανονικά, θα έχει σφάλματα στα αποτελέσματα.

  Οφείλονται σε σφάλματα κατά την υλοποίηση του αλγορίθμου και είναι δύσκολα στη διόρθωσή τους.

  Η παραβίαση της περατότητας είναι λογικό λάθος.

- **Λάθη κατά την εκτέλεση (οδηγούν σε αντικανονικό τερματισμό του προγράμματος)**

  Συνήθως, τα λάθη αυτά παραβιάζουν το κριτήριο της καθοριστικότητας του αλγορίθμου.

  Παραδείγματα

  1. Η διαίρεση με το μηδέν
  2. Η τετραγωνική ρίζα ενός αρνητικού αριθμού
  3. Η χρήση μιας μεταβλητής που δεν έχει πάρει τιμή
  4. Η εισαγωγή ενός γράμματος κατά την ανάγνωση ενός ακεραίου.

Διόρθωση λαθών: Με βάση τα διαγνωστικά μηνύματα ο προγραμματιστής κάνει τις κατάλληλες διορθώσεις, εφαρμόζει ξανά τη μεταγλώττιση και αυτό κάνει τόσες φορές όσες είναι απαραίτητες μέχρι να διορθωθούν όλα τα λάθη.

Πλεονεκτήματα – Μειονεκτήματα Μεταγλωττιστών – Διερμηνευτών

- Με τον μεταγλωττιστή, η χρησιμοποίηση ενός προγράμματος προϋποθέτει τη μεταγλώττιση και σύνδεσή του.

1 Δες και το κεφάλαιο 13

- Με τον διερμηνευτή η εκτέλεση άρα και η διόρθωση είναι άμεσες, επειδή οι εντολές μεταφράζονται μία προς μία .

  Ωστόσο αυτό συνεπάγεται πιο αργή εκτέλεση του προγράμματος σε σχέση με εκείνη του εκτελέσιμου προγράμματος από το μεταγλωττιστή.

Σήμερα γίνεται συνδυασμός στα προγραμματιστικά περιβάλλοντα.

- Ο **διερμηνευτής** χρησιμοποιείται κατά τη **δημιουργία – σύνταξη**
- Ο **μεταγλωττιστής** χρησιμοποιείται κατά την **τελική έκδοση** του προγράμματος.

Πέρα από την ύπαρξη συντάκτη, μεταγλωττιστή, συνδέτη, κάθε προγραμματιστικό περιβάλλον έχει τα δικά του εργαλεία και ιδιότητες.

Κεφάλαιο 9

Πίνακες

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

Ο δείκτης είναι μία μεταβλητή που μπορεί να έχει οποιοδήποτε δεκτό όνομα. Είναι σύνηθες όμως στον προγραμματισμό ως δείκτες να χρησιμοποιούνται οι μεταβλητές i, j, k.

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

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

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

Εκτός από μονοδιάστατους και δισδιάστατους πίνακες υπάρχουν πίνακες με περισσότερες διαστάσεις, τρισδιάστατοι, τετραδιάστατοι και γενικά πολυδιάστατοι, ανάλογα με τον αριθμό των δεικτών που χρησιμοποιούνται για τον καθορισμό των στοιχείων.

Ωστόσο τα περισσότερα προβλήματα αντιμετωπίζονται με τη χρήση πινάκων μονοδιάστατων ή δισδιάστατων.

Πότε πρέπει να χρησιμοποιούνται πίνακες

Μειονεκτήματα πινάκων

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

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

Γενικά, αν τα δεδομένα που εισάγονται σε ένα πρόγραμμα πρέπει να διατηρούνται στη μνήμη μέχρι το τέλος της εκτέλεσης, τότε η χρήση πινάκων βοηθάει ή συχνά είναι απαραίτητη για την επίλυση του προβλήματος.

Σε άλλη περίπτωση μπορεί να αποφεύγεται η χρήση τους.

Τυπικές επεξεργασίες πινάκων

1. Υπολογισμός αθροισμάτων στοιχείων του πίνακα
2. Εύρεση του μέγιστου ή του ελάχιστου στοιχείου

   Σε ταξινομημένο πίνακα το μέγιστο και το ελάχιστο βρίσκονται στα δύο ακριανά στοιχεία του πίνακα.
3. Ταξινόμηση των στοιχείων του πίνακα

   Η μέθοδος της Ευθείας Ανταλλαγής (Φυσαλίδας) είναι από τις απλούστερες αλλά δεν είναι η πιο αποδοτική.

   Υπάρχουν πολλές άλλες μέθοδοι ταξινόμησης καθώς και παραλλαγές αυτών.

   Η επιλογή του καλύτερου αλγορίθμου εξαρτάται κυρίως:

   α. από το πλήθος των στοιχείων του πίνακα και

   β. την αρχική τους διάταξη, αν δηλαδή ο πίνακας είναι τελείως αταξινόμητος ή μερικώς ταξινομημένος.
4. Αναζήτηση ενός στοιχείου του πίνακα Δύο είναι οι πλέον διαδεδομένοι αλγόριθμοι αναζήτησης:

   α. Η σειριακή αναζήτηση

      Η σειριακή μέθοδος αναζήτησης είναι η πιο απλή, αλλά και η λιγότερο αποτελεσματική μέθοδος. Χρησιμοποιείται όμως υποχρεωτικά για πίνακες που δεν είναι ταξινομημένοι.

   β. Η δυαδική αναζήτηση

      Η δυαδική αναζήτηση χρησιμοποιείται μόνο σε ταξινομημένους πίνακες και είναι σαφώς αποδοτικότερη από τη σειριακή μέθοδο.
5. Συγχώνευση δύο πινάκων

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

Κεφάλαιο 10

Τμηματικός Προγραμματισμός

Τμηματικός προγραμματισμός[2] ονομάζεται η τεχνική σχεδίασης και ανάπτυξης των προγραμμάτων ως ένα σύνολο από απλούστερα τμήματα προγραμμάτων.

Όταν ένα τμήμα προγράμματος επιτελεί ένα αυτόνομο έργο και έχει γραφεί χωριστά από το υπόλοιπο πρόγραμμα τότε αναφερόμαστε σε **υποπρόγραμμα**.

Χαρακτηριστικά των Υποπρογραμμάτων

Ο χωρισμός ενός προγράμματος σε υποπρογράμματα προϋποθέτει την ανάλυση του αρχικού προβλήματος σε μικρότερα υποπροβλήματα. Υπάρχουν τρεις ιδιότητες που πρέπει να διακρίνουν τα υποπρογράμματα:

1. Κάθε υποπρόγραμμα έχει μόνο μία είσοδο και μία έξοδο.

   Προσοχή: Ως είσοδος και έξοδος **δεν** αναφερόμαστε στα δεδομένα και στα αποτελέσματα, αλλά ότι το υποπρόγραμμα **έχει μία αρχή** όπου αρχίζει η εκτέλεση των εντολών του **και ένα τέλος** όπου ολοκληρώνεται η εκτέλεση των εντολών του.

2. Κάθε υποπρόγραμμα πρέπει να είναι ανεξάρτητο από το άλλο.

3. Το υποπρόγραμμα πρέπει να είναι αυτόνομο και να μην επηρεάζει τα άλλα υποπρογράμματα. Αυτό δεν είναι πάντα εφικτό.

4. Κάθε υποπρόγραμμα πρέπει να μην είναι πολύ μεγάλο.

   Το μέγεθος είναι υποκειμενικό. Γενικά, κάθε υποπρόγραμμα πρέπει να είναι τόσο μεγάλο ώστε να μπορεί εύκολα να κατανοηθεί και να ελεγχθεί.

Πλεονεκτήματα του Τμηματικού Προγραμματισμού

1. Διευκολύνει την ανάπτυξη του αλγορίθμου και του αντίστοιχου προγράμματος.

   Επειδή επιτρέπει να εξετάσουμε και να λύσουμε μικρά απλά προβλήματα, τα οποία συνθέτουν το συνολικό πρόβλημα, αντί να αντιμετωπίσουμε ολόκληρο το συνολικό πρόβλημα.

2. Διευκολύνει την κατανόηση και διόρθωση του προγράμματος.

   Ο χωρισμός ενός προγράμματος σε μικρά αυτοτελή τμήματα επιτρέπει τη γρήγορη διόρθωσή τους χωρίς οι αλλαγές αυτές να επηρεάζουν όλο το υπόλοιπο πρόγραμμα.

   Επίσης, διευκολύνει όποιον προσπαθήσει να διαβάσει και να κατανοήσει το πρόγραμμα.

3. Απαιτεί λιγότερο χρόνο και προσπάθεια στη συγγραφή του προγράμματος.

   Πολύ συχνά χρειάζεται η ίδια λειτουργία σε διαφορετικά σημεία ενός προγράμματος. Στην περίπτωση αυτή, μπορούμε να γράψουμε μία φορά τη λειτουργία ως υποπρόγραμμα, το οποίο μετά μπορούμε να καλούμε όσες φορές και όπου αυτό χρειάζεται.

4. Επεκτείνει τις δυνατότητες των γλωσσών προγραμματισμού.

2 Δες και την ενότητα 6.4.2 του κεφαλαίου 6, σελίδα 132, όπου αναφέρει τον τμηματικό προγραμματισμό και την ιεραρχική σχεδίαση προγράμματος.

Η χρήση ενός υποπρογράμματος δε διαφέρει από τη χρήση των ενσωματωμένων συναρτήσεων μιας γλώσσας προγραμματισμού. Γράφοντας ένα υποπρόγραμμα μπορούμε να προσθέσουμε λειτουργίες που δεν έχει μια γλώσσα προγραμματισμού, όπως μια συνάρτηση που υπολογίζει το μέγιστο δύο αριθμών ή την εφαπτομένη μιας γωνίας.

Παράμετροι

Κάθε υποπρόγραμμα για να ενεργοποιηθεί, καλείται από ένα άλλο υποπρόγραμμα ή το αρχικό πρόγραμμα, το οποίο ονομάζεται κύριο πρόγραμμα. Το σημείο ενός υποπρογράμματος ή του κυρίου προγράμματος που καλεί ένα άλλο υποπρόγραμμα λέγεται για ευκολία τμήμα προγράμματος.

Το υποπρόγραμμα συχνά **πρέπει να επικοινωνεί** με το υπόλοιπο πρόγραμμα. Συνήθως **δέχεται τιμές** από το τμήμα προγράμματος που το καλεί και **επιστρέφει αποτελέσματα**. Παράμετροι λέγονται οι τιμές αυτές που περνούν από ένα υποπρόγραμμα στο άλλο.

Οι παράμετροι είναι σαν τις απλές μεταβλητές ενός προγράμματος με μόνη διαφορά ότι χρησιμοποιούνται, για να περνούν τιμές στα υποπρογράμματα.

- **Παράμετρος** είναι μια μεταβλητή που επιτρέπει το πέρασμα της τιμής της από ένα τμήμα προγράμματος σε ένα άλλο.

Διαδικασίες και Συναρτήσεις

Υπάρχουν δύο ειδών υποπρογράμματα, οι διαδικασίες και οι συναρτήσεις.

- Η **διαδικασία** είναι ένας τύπος υποπρογράμματος που **μπορεί να εκτελέσει όλες** τις λειτουργίες ενός προγράμματος.

  Οι διαδικασίες μπορούν να εκτελέσουν οποιαδήποτε λειτουργία ενός προγράμματος.

  1. Μπορούν να διαβάσουν δεδομένα,
  2. Μπορούν να τυπώσουν αποτελέσματα, να εκτελέσουν υπολογισμούς,
  3. Μπορούν να μεταβάλλουν τις τιμές των μεταβλητών τους.
  4. Με τη χρήση των παραμέτρων μπορούν να δεχθούν τιμές και να μεταφέρουν τιμές σε άλλα υποπρογράμματα ή στο κύριο πρόγραμμα.

- Η **συνάρτηση** είναι ένας τύπος υποπρογράμματος που **υπολογίζει και επιστρέφει μόνο μία τιμή με το όνομά της**, όπως οι μαθηματικές συναρτήσεις.

  Η λειτουργία των συναρτήσεων είναι πιο περιορισμένη.

  Οι συναρτήσεις **υπολογίζουν μία μόνο τιμή** και μόνο αυτή την τιμή επιστρέφουν στο τμήμα προγράμματος που τις κάλεσε.

Κάθε διαδικασία ή συνάρτηση μπορεί να καλείται από το κύριο πρόγραμμα ή από άλλη διαδικασία ή συνάρτηση.

Σε κάθε περίπτωση με το τέλος της εκτέλεσης της διαδικασίας ή της συνάρτησης γίνεται επιστροφή ακριβώς μετά το σημείο απ' όπου κλήθηκε.

Προσοχή:

1. **Ο τρόπος κλήσης και ο τρόπος σύνταξης** των διαδικασιών και των συναρτήσεων **είναι διαφορετικός**.

2. Τόσο οι συναρτήσεις όσο και οι διαδικασίες **γράφονται μετά το τέλος του προγράμματος.**
3. Οι **συναρτήσεις καλούνται** απλά με την **εμφάνιση του ονόματός τους** σε οποιαδήποτε έκφραση, ενώ για την **κλήση των διαδικασιών** χρησιμοποιείται η ειδική εντολή **Κάλεσε**.

Παράδειγμα 1

1. Έστω **συνάρτηση** που υπολογίζει το μέγιστο δύο αριθμών. Μπορούμε να την καλέσουμε:

   Max ← Μέγιστο(x, y)

2. Έστω **διαδικασία** που υπολογίζει το μέγιστο δύο αριθμών. Μπορούμε να την καλέσουμε:

   Κάλεσε Μέγιστο(x, y, max)

Παράδειγμα 2

1. Έστω **συνάρτηση** που υπολογίζει το μέγιστο δύο αριθμών. Θέλουμε να βρούμε το διπλάσιο του μεγίστου. Με τη συνάρτηση έχουμε δύο επιλογές.

   α. Μπορούμε να την καλέσουμε εισάγοντας το αποτέλεσμά της σε μια μεταβλητή max. Μετά μπορούμε να χρησιμοποιήσουμε τη μεταβλητή max, για να βρούμε το διπλάσιο της:

   Max ← Μέγιστο(x, y)

   Διπλάσιο ← 2 * max

   β. Μπορούμε να καλέσουμε τη συνάρτηση απ' ευθείας μέσα στην έκφραση που υπολογίζει το διπλάσιο, δίχως τη χρήση της μεταβλητής max.

   Διπλάσιο ← 2 * Μέγιστο(x, y)

2. Έστω **διαδικασία** που υπολογίζει το μέγιστο δύο αριθμών. Θέλουμε να βρούμε το διπλάσιο του μεγίστου. Με τη διαδικασία έχουμε μία μόνο επιλογή.

   ΚΑΛΕΣΕ Μέγιστο(x, y, max)

   Διπλάσιο ← 2 * max

Ορισμός και κλήση Συναρτήσεων

1. Το όνομα της συνάρτησης μπορεί να είναι οποιοδήποτε έγκυρο όνομα της «Γλώσσας».
2. Η λίστα παραμέτρων είναι μια λίστα μεταβλητών, των οποίων οι τιμές μεταβιβάζονται στη συνάρτηση από το τμήμα προγράμματος που την καλεί κατά την κλήση της.

   Δηλαδή, οι παράμετροι μιας συνάρτησης λειτουργούν μόνο ως είσοδος δεδομένων.
3. Οι συναρτήσεις μπορούν να επιστρέφουν τιμές όλων των τύπων δεδομένων. Μια συνάρτηση μπορεί να είναι: **Πραγματική**, **Ακέραια**, **Χαρακτήρας** ή **Λογική**.

   Στις εντολές της συνάρτησης πρέπει υποχρεωτικά να υπάρχει μια εντολή εκχώρησης τιμής στο όνομα της συνάρτησης, ώστε η συνάρτηση να επιστρέψει στο τμήμα προγράμματος που την κάλεσε, το αποτέλεσμα που υπολόγισε.

Ορισμός και κλήση Διαδικασιών

1. Το όνομα της διαδικασίας μπορεί να είναι οποιοδήποτε έγκυρο όνομα της «Γλώσσας».
2. Η λίστα παραμέτρων είναι μια λίστα μεταβλητών, των οποίων οι τιμές μεταβιβάζονται κατά την κλήση προς τη διαδικασία ή/και επιστρέφονται προς το τμήμα προγράμματος που την καλεί μετά το τέλος της διαδικασίας.

   Η λίστα των παραμέτρων της διαδικασίας δεν είναι υποχρεωτική.

   Αν υπάρχει ορίζει τις τιμές που περνούν στη διαδικασία ως είσοδος και τις τιμές που αυτή επιστρέφει ως έξοδο (αποτελέσματα).
3. Στις εντολές της διαδικασίας μπορεί να υπάρχουν οποιεσδήποτε εντολές της «Γλώσσας».

Πραγματικές και Τυπικές Παράμετροι

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

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

Όλες οι μεταβλητές είναι γνωστές και έχουν ισχύ μόνο για το τμήμα προγράμματος, στο οποίο έχουν δηλωθεί, ισχύουν δηλαδή τοπικά για το συγκεκριμένο κύριο πρόγραμμα ή υποπρόγραμμα.

Κανόνες λίστας παραμέτρων

Οι λίστες των παραμέτρων πρέπει να ακολουθούν τους εξής κανόνες:

1. Ο αριθμός των πραγματικών και των τυπικών παραμέτρων πρέπει να είναι ίδιος.
2. Κάθε πραγματική παράμετρος αντιστοιχεί στην τυπική παράμετρο που βρίσκεται στην αντίστοιχη θέση.

Για παράδειγμα, η πρώτη της λίστας των τυπικών παραμέτρων με την πρώτη της λίστας των πραγματικών παραμέτρων.

3. Η τυπική και η αντίστοιχη της πραγματική παράμετρος πρέπει να είναι του ίδιου τύπου.

Εμβέλεια

Το τμήμα του προγράμματος που ισχύουν οι μεταβλητές λέγεται **εμβέλεια** μεταβλητών

- Οι μεταβλητές στη **«Γλώσσα»** είναι γνωστές στο αντίστοιχο υποπρόγραμμα που δηλώνονται και μόνο σε αυτό.
- Όλες οι μεταβλητές (και οι σταθερές) είναι **τοπικές** στο συγκεκριμένο τμήμα προγράμματος.
- Ο μόνος τρόπος για να περάσει μία τιμή από ένα υποπρόγραμμα σε ένα άλλο ή από το κυρίως πρόγραμμα σε ένα υποπρόγραμμα είναι δια μέσου των παραμέτρων κατά το στάδιο της κλήσης του υποπρογράμματος και μετά το τέλος της εκτέλεσης του υποπρογράμματος.

Απεριόριστη εμβέλεια.

- Όλες οι σταθερές είναι γνωστές και μπορούν να χρησιμοποιούνται σε οποιοδήποτε τμήμα του προγράμματος, άσχετα που δηλώθηκαν.
- Όλες οι μεταβλητές είναι **καθολικές**.

Η απεριόριστη εμβέλεια καταστρατηγεί την αρχή της αυτονομίας των υποπρογραμμάτων, και δημιουργεί πολλά προβλήματα.

Περιορισμένη εμβέλεια

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

  Στη «Γλώσσα» έχουμε περιορισμένη εμβέλεια.

Τα πλεονεκτήματα της περιορισμένης εμβέλειας είναι:

1. Η απόλυτη αυτονομία όλων των υποπρογραμμάτων και
2. Η δυνατότητα να χρησιμοποιείται οποιοδήποτε όνομα, χωρίς να ενδιαφέρει αν το ίδιο χρησιμοποιείται σε άλλο υποπρόγραμμα.

Μερικώς περιορισμένη εμβέλεια.

- 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.

Η Στοίβα Χρόνου Εκτέλεσης

Όταν ένα υποπρόγραμμα καλείται από ένα τμήμα προγράμματος τότε η αμέσως επόμενη διεύθυνση του τμήματος προγράμματος που ονομάζεται **διεύθυνση επιστροφής** αποθηκεύεται από το μεταφραστή σε μια στοίβα που ονομάζεται **στοίβα χρόνου εκτέλεσης**.

Όταν το υποπρόγραμμα ολοκληρώσει την εκτέλεσή του τότε η διεύθυνση επιστροφής απωθείται από τη στοίβα χρόνου εκτέλεσης, ώστε το πρόγραμμα να συνεχίσει την εκτέλεσή του με την αμέσως επόμενη εντολή που ακολουθεί την κλήση του υποπρογράμματος.

Παράδειγμα

ΚΑΛΕΣΕ Μέγιστο(x, y, max)	ΔΙΑΔΙΚΑΣΙΑ **Μέγιστο**(x, y, max)
(@) Διπλάσιο ← 2 * max	ΜΕΤΑΒΛΗΤΕΣ
	ΠΡΑΓΜΑΤΙΚΕΣ: x, y, max
	ΑΡΧΗ
	ΑΝ x>y ΤΟΤΕ
	max ← x
	ΑΛΛΙΩΣ
	max ← y
	ΤΕΛΟΣ_ΑΝ
	ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ

Αυτή είναι η **διεύθυνση επιστροφής**. Είναι η διεύθυνση της εντολής που **ακολουθεί την κλήση** του υποπρογράμματος.

Κεφάλαιο 13

Κατηγορίες λαθών

Σε ένα πρόγραμμα είναι δυνατό να παρουσιαστούν διαφορετικής μορφής λάθη, τα οποία μπορούν να χωριστούν σε τρεις βασικές κατηγορίες:

Λάθη κατά την υλοποίηση (συντακτικά)

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

  Για Ι από 1 έως 10

- Η χρήση μιας δομής ελέγχου χωρίς την εντολή τερματισμού της

  Αν Χ = Τότε

  Εμφάνισε ‘Λάθος’

Ένα λάθος που προκαλείται κατά τη συγγραφή του προγράμματος, ανιχνεύεται από το μεταγλωττιστή, ο οποίος εμφανίζει κάποιο προειδοποιητικό μήνυμα.

Η εκτέλεση του προγράμματος δεν επιτρέπεται μέχρι να το διορθώσει ο προγραμματιστής.

Τα σύγχρονα προγραμματιστικά περιβάλλοντα παρέχουν εργαλεία αυτόματου ελέγχου σύνταξης των εντολών και παρακολουθούν τον προγραμματιστή κατά τη συγγραφή του προγράμματος.

Μόλις διαπιστώσουν κάποιο συντακτικό λάθος, σταματούν και απαιτούν τη διόρθωσή του.

- Συνήθως αντιλαμβάνονται ακριβώς το λάθος που δημιουργήθηκε
- Προτείνουν αναλυτικά τον τρόπο διόρθωσής του, εμφανίζοντας σε ενημερωτικό πλαίσιο την ορθή σύνταξη της εντολής που προκλήθηκε το λάθος.

Λάθη κατά την εκτέλεση

Τα λάθη που προκαλούνται κατά το χρόνο εκτέλεσης του προγράμματος, είναι πιο επώδυνα γιατί συνήθως εμφανίζονται κατά την εκτέλεση του προγράμματος και τις περισσότερες φορές προκαλούν τον αντικανονικό τερματισμό της εφαρμογής και το κρέμασμα του συστήματος.

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

Η πρόληψη τέτοιων λαθών είναι αρκετά δύσκολη

- Συνήθως οφείλονται σε καταστάσεις που δεν είναι εύκολο να ελεγχθούν
- Πολλές φορές εμφανίζονται μετά από μεγάλο χρονικό διάστημα.

Τέτοια λάθη είναι δυνατό να προκληθούν:

- Από την κλήση μιας διαδικασίας με δεδομένα που δεν μπορεί να χειριστεί, όπως:
  - Η αναζήτηση διαγραμμένων αρχείων,
  - Η προσπάθεια διαίρεσης ενός αριθμού με το μηδέν
  - Η υπερχείλιση μιας αριθμητικής μεταβλητής
- Από δυσλειτουργία του υλικού μέρους του υπολογιστή, όπως
  - Η καταστροφή του σκληρού δίσκου του συστήματος,
  - Ο τερματισμός μιας σύνδεσης δικτύου
  - Η αποσύνδεση του εκτυπωτή.

Λογικά λάθη

Τα λογικά λάθη είναι συνήθως λάθη σχεδιασμού και δεν προκαλούν τη διακοπή της εκτέλεσης του προγράμματος.

- Ο **μεταγλωττιστής** της γλώσσας προγραμματισμού δεν ανιχνεύει κανένα συντακτικό λάθος
- Κατά την **εκτέλεση** του προγράμματος δεν παρουσιάζονται ανεπιθύμητες καταστάσεις σφαλμάτων
- Όμως, **δεν παράγονται τα επιθυμητά αποτελέσματα**.

Η ανίχνευση τέτοιων λαθών

- Δεν είναι δυνατό να πραγματοποιηθεί από κάποιο εργαλείο του υπολογιστή
- Διαπιστώνονται μόνο με τη **διαδικασία ελέγχου** και την **ανάλυση των αποτελεσμάτων** των προγραμμάτων.

Εκσφαλμάτωση

Εκσφαλμάτωση καλείται η διαδικασία ελέγχου, εντοπισμού και διόρθωσης των σφαλμάτων ενός προγράμματος.

Στόχος της διαδικασίας εκσφαλμάτωσης είναι ο εντοπισμός των σημείων του προγράμματος που προκαλούν προβλήματα στη λειτουργία του.

Η εργασία της εκσφαλμάτωσης δεν είναι εύκολη, απαιτεί βαθιά γνώση της γλώσσας προγραμματισμού και ικανότητες από τον προγραμματιστή.

Για τον εντοπισμό ενός λάθους δεν υπάρχουν ιδιαίτερα μυστικά.

Η εκσφαλμάτωση είναι ένα πρόβλημα λογικής.

Όσο πιο καλά αντιλαμβάνεται ο προγραμματιστής τον τρόπο που εργάζεται το πρόγραμμα, τόσο πιο εύκολα και σύντομα θα εντοπίσει λάθη που προκαλούν δυσλειτουργίες.

1. Η εισαγωγή γραμμών με σχόλια σε ένα πρόγραμμα βοηθά σημαντικά την εκσφαλμάτωση.
2. Τα ονόματα των μεταβλητών πρέπει να ανάγουν στο περιεχόμενο τους. Έτσι διευκολύνεται η εκσφαλμάτωση.

Η εκσφαλμάτωση τέτοιων λαθών μπορεί να γίνει:

- Μέσα από εργαλεία εκσφαλμάτωσης ή
- Από ειδικές εντολές ή συναρτήσεις που προσφέρει το περιβάλλον προγραμματισμού.