ΑΕΠΠ - Θεωρία

Η έννοια Πρόβλημα 1.1

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

Παραδείγματα προβλημάτων που αφορούν στην καθημερινή ζωή:
Ανεργία – ναρκωτικά – υποσιτισμός - ξενοφοβία – μόλυνση – πόλεμοι

Παραδείγματα προβλημάτων που αφορούν στην τεχνολογία:
Ενεργειακό – τρύπα όζοντος – αργή ταχύτητα μετάδοσης δεδομένων

Κατανόηση προβλήματος 1.2

Η αντιμετώπιση ενός προβλήματος είναι καταδικασμένη σε αποτυχία αν πρώτα δεν έχει γίνει απόλυτα κατανοητό το πρόβλημα που τίθεται.

Η κατανόηση ενός προβλήματος είναι συνάρτηση
της σωστής διατύπωσης από το δημιουργό
της σωστής ερμηνείας από αυτόν που καλείται να το αντιμετωπίσει

Συνηθέστερα μέσα για τη διατύπωση ενός προβλήματος είναι
ο γραπτός και
ο προφορικός λόγος

Τα προβλήματα που μπορεί να κληθούμε να αντιμετωπίσουμε αφορούν
στα μαθηματικά
στη φυσική
στη λογική
στην καθημερινή ζωή κ.λ.π

Σαφήνεια διατύπωσης

Ο προφορικός ή γραπτός λόγος πρέπει να χαρακτηρίζεται από σαφήνεια.

Πρόκληση παρερμηνειών και παραπλανήσεων μπορεί να γίνει από
άστοχη χρήση ορολογίας
λανθασμένη σύνταξη

παράδειγμα ασαφούς διατύπωσης
«ο Γιάννης και η Μαρία είναι παντρεμένοι»
πρώτη ερμηνεία : ο Γιάννης και η Μαρία είναι παντρεμένοι μεταξύ τους
δεύτερη ερμηνεία : ο Γιάννης είναι παντρεμένος και η Μαρία είναι παντρεμένη

Με τον όρο δεδομένο δηλώνεται οποιοδήποτε στοιχείο μπορεί να γίνει από εμάς αντιληπτό με μία από τις πέντε αισθήσεις μας.

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

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

Δομή προβλήματος 1.3

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

Ανάλυση είναι ο χωρισμός ενός προβλήματος στα στοιχεία από τα οποία αποτελείται δηλ σε άλλα απλούστερα προβλήματα.

Με τη σειρά τους τα νέα προβλήματα μπορούν να αναλυθούν σε άλλα ακόμη πιο απλά.
Με την ανάλυση ελαττώνεται η δυσκολία αντιμετώπισης των προβλημάτων.

Η παρουσίαση της ανάλυσης ενός προβλήματος μπορεί να γίνει:

- **Φραστικά** (περιγράφουμε με λόγια πως και σε ποια υποπροβλήματα αναλύεται το πρόβλημα και τα υποπροβλήματα που προκύπτουν)
- **με διαγράμματα** (η περιγραφή γίνεται με ένα διάγραμμα σε σχήμα γενεαλογικού δένδρου όπου κάθε πρόβλημα έχει παιδιά τα υποπροβλήματα στα οποία αναλύεται)

Καθορισμός απαιτήσεων 1.4

Η σωστή επίλυση ενός προβλήματος προϋποθέτει :
- τον επακριβή προσδιορισμό των δεδομένων
- λεπτομερειακή καταγραφή των ζητουμένων

Θα πρέπει να δοθεί μεγάλη προσοχή στην ανίχνευση των δεδομένων και στην αποσαφήνιση των ζητουμένων του προβλήματος.

Αν δεν γίνει κατανοητό τι ακριβώς ζητάει το πρόβλημα θα πρέπει να τεθούν μια σειρά από ερωτήσεις για τη διευκρίνηση πιθανών απορριών σχετικά με τα ζητούμενα, τον τρόπο παρουσίασής τους κλπ.

Τα στάδια αντιμετώπισης ενός προβλήματος είναι τρία
- **Κατανόηση**, όπου απαιτείται η σωστή και πλήρης αποσαφήνιση των δεδομένων και των ζητούμενων του προβλήματος
- **Ανάλυση**, όπου το αρχικό πρόβλημα διασπάται σε άλλα επί μέρους απλούστερα προβλήματα
- **Επίλυση**, όπου υλοποιείται ή λύση του προβλήματος, μέσω της λύσης των επιμέρους προβλημάτων.

Τι είναι Αλγόριθμος 2.1

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

Κάθε αλγόριθμος πρέπει να ικανοποιεί τα επόμενα κριτήρια

Είσοδος: καμία, μία ή περισσότερες τιμές δεδομένων πρέπει να δίνονται ως είσοδοι στον αλγόριθμο. Στην περίπτωση που δε δέχεται εξωτερικά καμία είσοδο ο αλγόριθμος θα δημιουργεί κάποιες τιμές δεδομένων εσωτερικά με κάποια συγκεκριμένη διαδικασία	οι τιμές των μεταβλητών εισόδου όπως οι μισθοί των υπαλλήλων ή οι βαθμοί των μαθητών μιας τάξης
Έξοδος: ο αλγόριθμος πρέπει να δημιουργεί τουλάχιστον μια τιμή δεδομένων ως αποτέλεσμα	οι τιμές των μεταβλητών εξόδου - τα αποτελέσματα όπως ο μέσος όρος των βαθμών ενός μαθητή
Περατότητα: ο αλγόριθμος να τελειώνει μετά από πεπερασμένα βήματα εκτέλεσης των εντολών του	Αν κατά την εκτέλεση ενός αλγορίθμου έχουμε άπειρες επαναλήψεις κάποιας ενέργειας τότε ο αλγόριθμος δεν έχει περατότητα
Καθοριστικότητα: κάθε εντολή πρέπει να καθορίζεται χωρίς καμιά αμφιβολία για τον τρόπο εκτέλεσής της	Μια εντολή διαίρεσης πρέπει να θεωρεί και την περίπτωση όπου ο διαιρέτης παίρνει μηδενική τιμή
Αποτελεσματικότητα: κάθε μεμονωμένη εντολή του αλγορίθμου να είναι απλή και κατανοητή ώστε να μπορεί να εκτελεστεί επακριβώς και να δίνει αποτελέσματα	Η εντολή δεν αρκεί να έχει ορισθεί αλλά πρέπει να είναι και εκτελέσιμη

Περιγραφή και αναπαράσταση αλγορίθμων 2.3

Ελεύθερο κείμενο: αποτελεί τον πιο ανεπεξέργαστο και αδόμητο τρόπο παρουσίασης αλγορίθμου με χρήση απλής ελληνικής γλώσσας, όπως μιλάμε (κίνδυνος έλλειψης αποτελεσματικότητας)

Διαγράμματα ροής: χρήση ειδικών συμβόλων για την αναπαράσταση των διαφόρων βημάτων του αλγορίθμου (δεν αποτελεί την καλύτερη λύση και χρησιμοποιούνται σπάνια είναι όμως ο πιο εποπτικός τρόπος)

Φυσική γλώσσα: ελεύθερο κείμενο όπου οι προτάσεις έχουν διαχωριστεί σε παραγράφους και έχουν αριθμηθεί (μπορεί να παραβιαστεί το κριτήριο του καθορισμού)

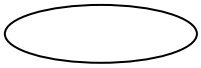
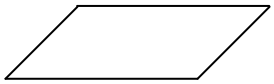
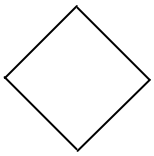
Κωδικοποίηση (Ψευδογλώσσα): χρήση συγκεκριμένων λέξεων με αυστηρά καθορισμένη έννοια και τρόπο χρήσης με τις οποίες φτιάχνουμε ένα πρόγραμμα που όταν

εκτελεστεί θα δώσει τα ίδια αποτελέσματα με τον αλγόριθμο (είναι ο πιο δομημένος τρόπος)

Βασικές συνιστώσες/εντολές ενός Αλγορίθμου 2.4

- Δομές ακολουθίας (σειριακές εντολές, αναθέσεις τιμών)
- Επιλογής νε βάση κριτήρια
- Διαδικασίες επανάληψης
- Ενέργειες πολλαπλών επιλογών
- Συνδυασμοί εμφωλευμένων περιπτώσεων

Σύμβολα διαγράμματος ροής

	Δηλώνει την αρχή και το τέλος κάθε αλγορίθμου
	Δηλώνει την εκτέλεση μιας ή περισσοτέρων πράξεων
	Δηλώνει είσοδο ή έξοδο στοιχείων
	Δηλώνει μια ερώτηση με δύο εξόδους για απάντηση (συνθήκη)

Καθένα από τα βήματα του αλγορίθμου ονομάζεται **εντολή**

Οι εντολές ενός αλγορίθμου μπορούν να εκτελούνται **σειριακά** (η μια μετά την άλλη) ή σε κάποιο σημείο **η ροή του αλγορίθμου να αλλάζει** σύμφωνα με μια συνθήκη

Πρόγραμμα Η/Υ είναι ένας αλγόριθμος εκφρασμένος με τέτοιο τρόπο ώστε να μπορεί να εκτελεστεί από ένα Η/Υ

Σταθερά ονομάζεται το μέγεθος που η τιμή του παραμένει αμετάβλητη σε όλη τη διάρκεια εκτέλεσης του αλγορίθμου
Οι σταθερές διακρίνονται σε :

- αριθμητικές π.χ. 123 -1,25
- αλφαριθμητικές π.χ. “Μέσος όρος”, “Κατάσταση αποτελεσμάτων”
- λογικές που είναι ακριβώς δύο ΑΛΗΘΗΣ και ΨΕΥΔΗΣ

Μεταβλητή είναι το μέγεθος στο οποίο εκχωρείται μια τιμή, η οποία μπορεί να αλλάζει κατά τη διάρκεια εκτέλεσης του αλγορίθμου.

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

- αριθμητικές (οι τιμές είναι μόνο αριθμοί)
- αλφαριθμητικές (οι τιμές μπορεί να είναι οποιοσδήποτε χαρακτήρας)
- λογικές (οι τιμές μπορεί να είναι μόνο δύο ΑΛΗΘΗΣ, ΨΕΥΔΗΣ χωρίς εισαγωγικά)

ΠΡΟΣΟΧΗ

- ☞ Μια μεταβλητή έχει όνομα, τύπο και τιμή.
- ☞ Ποτέ το όνομα μιας μεταβλητής δεν μπορεί να ξεκινάει με αριθμό.
- ☞ Τα ονόματα των μεταβλητών και των σταθερών επιλέγονται με τέτοιο τρόπο ώστε να εξηγούν τη σημασία και το ρόλο των μεγεθών που εκφράζουν.

Δεσμευμένες λέξεις είναι λέξεις που χρησιμοποιούνται σε έναν αλγόριθμο για να περιγράψουν συγκεκριμένες ενέργειες.
Οι λέξεις αυτές δεν μπορούν να χρησιμοποιηθούν για ονόματα σταθερών ή μεταβλητών.

Αποδεκτά ονόματα μεταβλητών

1. να ξεκινούν πάντα από γράμμα
2. να περιέχουν μόνο γράμματα, αριθμούς και την κάτω παύλα _
3. να μην είναι δεσμευμένη λέξη

Παραδείγματα αποδεκτών ονομάτων μεταβλητών

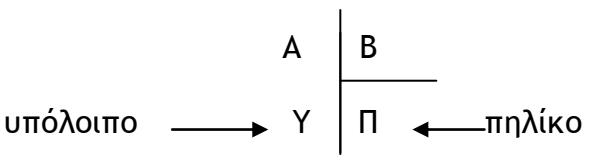
Α, α	Ένα γράμμα
Β1, C_1	Γράμμα και αριθμός
βαθμός, όνομα, μέσος_όρος, αριθ	Μια ή δύο λέξεις ή μέρος λέξης

Τελεστές λέμε τα γνωστά σύμβολα που χρησιμοποιούνται στις διάφορες πράξεις.
Διακρίνονται σε :

Αριθμητικούς : **+, -, *, /, ^, Mod, div**
Λογικούς : **και** (σύζευξη), **ή** (διάζευξη), **όχι** (άρνηση)
Συγκριτικούς : **<=, <, >=, >, =, <>**

Προτεραιότητα τελεστών
Παρενθέσεις -> Δυνάμεις -> Πολ/σμοί / Διαιρέσεις -> Προσθέσεις / Αφαιρέσεις -> Συγκρίσεις -> Άρνηση -> Σύζευξη -> Διάζευξη

Τι είναι το mod και τι το div
Έστω ότι έχουμε δύο ακέραιους αριθμούς τον Α και τον Β και ότι θέλουμε να τους διαιρέσουμε



- το πηλίκο της ακέραιας διαίρεσης συμβολίζεται με **A div B**
- υπόλοιπο της ακέραιας διαίρεσης συμβολίζεται με **A mod B**

Π.χ
10 mod 3 = 1 2 mod = 2
10 div 3 = 3 2 div = 0

Πότε χρησιμοποιούνται οι τελεστές mod και div

1. Για να δείξουμε ότι ένας ακέραιος αριθμός Α είναι πολλαπλάσιο ενός ακεραίου Β. Σε αυτή την περίπτωση **A mod B = 0**
2. Για να διαχωρίσουμε τους άρτιους από τους περιττούς. Ένας ακέραιος Α είναι άρτιος αν **A mod 2 = 0** αλλιώς είναι περιττός
3. Για να πάρουμε σαν αποτέλεσμα μιας διαίρεσης μεταξύ ακεραίων πάντα έναν ακέραιο αριθμό.

Οι λογικοί τελεστές είναι αυτοί που εκτελούν τις λογικές πράξεις σε έναν αλγόριθμο. Χρησιμοποιούνται για να κατασκευάσουμε σύνθετες λογικές εκφράσεις οι οποίες θα δίνουν ως αποτέλεσμα την τιμή Αληθής ή Ψευδής

απάντηση = 'ναι' Ή απάντηση = 'ΝΑΙ'

Εκφράσεις είναι προτάσεις που διαμορφώνονται από τους τελεστέους που είναι οι σταθερές και οι μεταβλητές και από τους τελεστές. Μπορεί να αποτελείται από μια μόνο μεταβλητή ή σταθερά μέχρι μια πολύπλοκη μαθηματική παράσταση.
Για παράδειγμα

B * 2 + Y^3
5 * X – (A + 3) > 8 * Y
X > 4 ΚΑΙ X <= 24

Η εντολή που αποδίδει τιμή σε μια μεταβλητή ονομάζεται **εντολή εκχώρησης** και συμβολίζετε με ←

Μεταβλητή ← Έκφραση

Παραδείγματα εκχώρησης τιμής

α ← 2	η τιμή της μεταβλητής α είναι το 2
α ← β + γ	η τιμή της μεταβλητής α είναι το άθροισμα των τιμών των μεταβλητών β και γ
τ ← αληθής	η τιμή της μεταβλητής τ είναι η αληθής
α ← α + 1	η νέα τιμή της μεταβλητής α είναι η παλιά συν 1 (αύξηση του α κατά 1)
β ← "Νίκος"	η τιμή της μεταβλητής β είναι το όνομα Νίκος

Στο αριστερό μέρος της εντολής δεν είναι δυνατό να υπάρχει παράσταση
Για παράδειγμα η έκφραση 2*Α ← 3*Α+5 δεν είναι αποδεκτή

Το όνομα μιας μεταβλητής μπορεί να είναι το ίδιο με την τιμή της πχ. Α ← "Α" ή άθροισμα ← "άθροισμα"

Για να είναι δυνατή η εκτέλεση μιας εντολής εκχώρησης πρέπει όλες οι μεταβλητές που βρίσκονται στο δεξιό μέρος της εντολής να έχουν τιμή.

Εισαγωγή - Διάβασμα δεδομένων

Επιτυγχάνεται με την εντολή **Διάβασε** η οποία συνοδεύεται με το όνομα μιας ή περισσοτέρων μεταβλητών.
Μετά την ολοκλήρωση της ενέργειας αυτής, η μεταβλητή θα έχει λάβει ως περιεχόμενο την τιμή που πληκτρολογήσατε.

Παράδειγμα
- **Διάβασε** βαθμός

Αν πληκτρολογήσετε τον αριθμό 17 η τιμή της μεταβλητής *βαθμός* θα είναι το 17

Εμφάνιση αποτελεσμάτων

Επιτυγχάνεται με την εντολή **Εμφάνισε (Εκτύπωσε** αν πρόκειται για τύπωση στον εκτυπωτή) η οποία συνοδεύεται με το όνομα μιας ή περισσοτέρων μεταβλητών, μηνύματα ή με το συνδυασμό των παραπάνω.

Παραδείγματα
- **Εμφάνισε** βαθμός

Εμφανίζει την τιμή της μεταβλητής βαθμός

- **Εμφάνισε** αριθ1 + αριθ2

Εμφανίζει το άθροισμα των τιμών των μεταβλητών αριθ1 και αριθ2

- **Εμφάνισε** "το άθροισμα των αριθμών είναι :", αριθ1 + αριθ2

Εμφανίζει το μήνυμα που είναι μέσα στα εισαγωγικά και αμέσως μετά το αποτέλεσμα της πρόσθεσης

Βασικές αλγοριθμικές δομές

Α. Δομή ακολουθίας

Σε αυτήν την δομή οι εντολές εκτελούνται η μια μετά την άλλη μία μόνο φορά και χωρίς να παραλειφθεί καμία.
Η ακολουθιακή δομή εντολών χρησιμοποιείται για την αντιμετώπιση απλών προβλημάτων όπως το παρακάτω παράδειγμα

Αλγόριθμος άθροισμα
Διάβασε α
Διάβασε β
γ ← α + β
Εμφάνισε γ
Τέλος άθροισμα

Β. Δομή επιλογής

Συνήθως η λύση ενός προβλήματος εκτός από τα απλά βήματα που εκτελούνται το ένα μετά το άλλο, περιέχει και κάποιες συνθήκες όπου λαμβάνονται αποφάσεις με βάση κάποια δεδομένα κριτήρια.

Γενικά η δομή επιλογής περιλαμβάνει :

- τον έλεγχο κάποιας συνθήκης που μπορεί να έχει δύο τιμές (αληθής ή ψευδής)
- την ομάδα εντολών που θα εκτελεστεί σε κάθε περίπτωση (υπάρχει περίπτωση μια ομάδα εντολών να μην εκτελεστεί ποτέ)

παρακάτω δίνεται ένας πίνακας με τις δομές επιλογής

απλή επιλογή Αν <συνθήκη> τότε Εντολές Τέλος_αν	 Διάβασε α Αν α < 0 τότε α ← α * (-1) Τέλος_αν Εμφάνισε α
Σύνθετη επιλογή Αν <συνθήκη> τότε Εντολές 1 Αλλιώς Εντολές 2 Τέλος_αν	 Διάβασε α, β Αν α < β τότε γ ← α +β αλλιώς γ ← α * β Τέλος_αν Εμφάνισε γ
Πολλαπλή επιλογή Αν <συνθήκη1> τότε Εντολές 1 Αλλιώς_αν <συνθήκη2> τότε Εντολές 2 Αλλιώς_αν …… ……. Αλλιώς Εντολές ν Τέλος_αν	 Διάβασε α Αν α = 1 τότε Εμφάνισε “Α” Αλλιώς_αν α = 2 τότε Εμφάνισε “Β” Αλλιώς_αν α = 3 τότε Εμφάνισε “Γ” Αλλιώς Εμφάνισε “άγνωστος” Τέλος_αν

Εμφωλευμένη επιλογή	Διάβασε βάρος, ύψος
Αν <συνθήκη1> τότε	Αν βάρος < 80 τότε
Αν <συνθήκη2> τότε	Αν ύψος < 1.70 τότε
Εντολές 1	Εμφάνισε “ελαφρύς, κοντός”
Αλλιώς	Αλλιώς
Εντολές 2	Εμφάνισε “ελαφρύς, ψηλός”
Τέλος_αν	Τέλος_αν
Αλλιώς	Αλλιώς
Αν <συνθήκη3> τότε	Αν ύψος < 1.70 τότε
Εντολές 3	Εμφάνισε “βαρύς, κοντός”
Αλλιώς	Αλλιώς
Εντολές 4	Εμφάνισε “βαρύς, ψηλός”
Τέλος_αν	Τέλος_αν
Τέλος_αν	Τέλος_αν

Συνθήκες

Μια συνθήκη έχει πάντοτε μια τιμή από δύο, **αληθής** ή **ψευδής** ανάλογα με το αν ισχύει ή όχι η συνθήκη

Σύμβολα που χρησιμοποιούνται στις συνθήκες είναι οι τελεστές σύγκρισης :
<, >, <=, >=, =, <>

Οι λογικές πράξεις είναι:

Σύζευξη – ΚΑΙ
Η σύζευξη δύο λογικών συνθηκών είναι αληθής όταν και οι δύο είναι αληθείς

Διάζευξη – Ή
Η διάζευξη δύο λογικών συνθηκών είναι αληθής όταν τουλάχιστον μια είναι αληθής

Άρνηση – ΟΧΙ
Η άρνηση μιας συνθήκης είναι αληθής όταν ή συνθήκη είναι ψευδής

Ο επόμενος πίνακας δίνει τις τιμές των τριών αυτών λογικών πράξεων για όλους τους συνδυασμούς τιμών

Α	**Β**	**Α ΚΑΙ Β**	**Α Ή Β**	**ΟΧΙ Α**
Ψευδής	Ψευδής	Ψευδής	Ψευδής	Αληθής
Ψευδής	Αληθής	Ψευδής	Αληθής	Αληθής
Αληθής	Ψευδής	Ψευδής	Αληθής	Ψευδής
Αληθής	Αληθής	Αληθής	Αληθής	Ψευδής

Γ. Δομή επανάληψης

Η δομή επανάληψης χρησιμοποιείται όταν μια σειρά από εντολές πρέπει να επαναληφθεί πολλές φορές μέσα σε ένα αλγόριθμο.
Η επανάληψη αυτή συνεχίζεται ανάλογα με την τιμή που θα έχει κάποια συνθήκη η οποία όμως πρέπει να διασφαλίζει ότι η επανάληψη θα σταματήσει σε κάποιο σημείο και δε θα εκτελείται απεριόριστα.
Η διαδικασία επανάληψης είναι ιδιαίτερα συχνή, και υπάρχουν τρεις εντολές γι' αυτό το σκοπό

Έλεγχος επανάληψης στην αρχή
 	
Όσο <συνθήκη> επανάλαβε
  Εντολές
Τέλος_επανάληψης
	Ι ← 1
	Όσο Ι <= 100 επανάλαβε
	  Εμφάνισε Ι
	  Ι ← Ι + 1
	Τέλος_επανάληψης

Έλεγχος επανάληψης στο τέλος
 	
Αρχή επανάληψης
  Εντολές
Μέχρις_ότου <συνθήκη>
	Ι ← 1
	Αρχή επανάληψης
	  Εμφάνισε Ι
	  Ι ← Ι + 1
	Μέχρις_ότου Ι > 100

Συγκεκριμένος αριθμός επαναλήψεων
 	
Για Ι από α_τ μέχρι τ_τ με_βήμα β
  Εντολές
Τέλος_επανάληψης
	
	Για Ι από 1 μέχρι 100
	  Εμφάνισε Ι
	Τέλος_επανάληψης

Παρατηρήσεις στις επαναληπτικές διαδικασίες

Βρόγχος είναι το σύνολο των εντολών που επαναλαμβάνεται σε μια δομή επανάληψης.

Μετρητές είναι μεταβλητές που μετρούν το πλήθος των επαναλήψεων.

Η εντολή **Για ……** δε χρειάζεται κάποιο μετρητή να μετράει τον αριθμό των επαναλήψεων γιατί η **μεταβλητή Ι** έχει αυτό το ρόλο.
Οι άλλες δύο εντολές πρέπει να έχουν μια μεταβλητή (π.χ Ι) για να μετρούν το πλήθος των επαναλήψεων όταν χρειάζεται.

Στη **Για** η ομάδα εντολών δεν θα εκτελεστεί στην περίπτωση που η **α_τ > τ_τ** και το βήμα είναι θετικό.

Οι επαναληπτικές δομές **Όσο …… επανάλαβε** και **Αρχή επανάληψης …… μέχρις_ότου** χρησιμοποιούνται συνήθως όταν δεν ξέρουμε από πριν τον αριθμό των επαναλήψεων του βρόγχου.
Αν υλοποιήσετε μια επανάληψη με **Όσο** και με **μέχρις_ότου** τότε η συνθήκη της **Όσο** θα είναι αντίθετη από αυτή της **μέχρις_ότου.**

Παράδειγμα
Όσο όνομα <> 'Τέλος' επανάλαβε
Μέχρις _ότου όνομα = 'Τέλος'

Στη δομή επανάληψης **μέχρις_ότου** η ομάδα εντολών του βρόγχου εκτελείται τουλάχιστον μια φορά, επειδή η συνθήκη βρίσκεται και ελέγχεται στο τέλος.

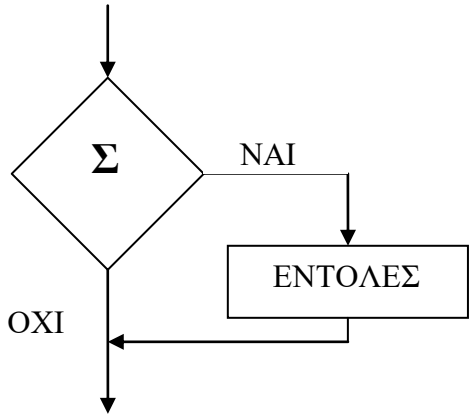
Στην **Όσο** επειδή η συνθήκη ελέγχεται στην αρχή, αν δεν ισχύει, η ομάδα εντολών της **Όσο** δεν εκτελείται καμία φορά.
Η **Όσο** τερματίζεται όταν η συνθήκη γίνει ψευδής.
Η **μέχρις_ότου** τερματίζεται όταν η συνθήκη γίνει αληθής.

Όταν μια εντολή επανάληψης εμπεριέχεται σε μια άλλη εντολή επανάληψης τότε καλείται εμφωλευμένη δομή επανάληψης

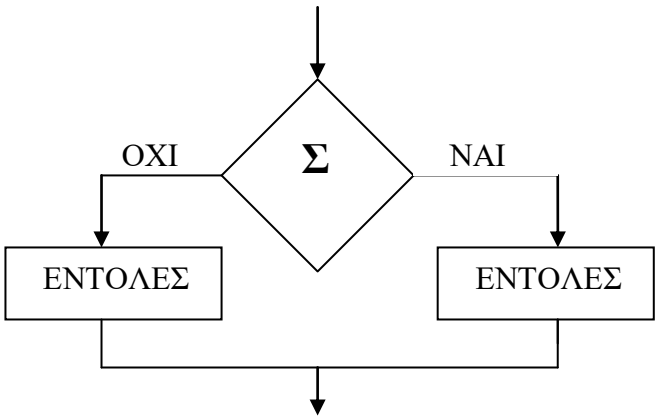
Η χρήση τιμών για τον τερματισμό μίας επαναληπτικής διαδικασίας είναι συνήθης στον προγραμματισμό. Η τιμή αυτή ορίζεται από τον προγραμματιστή και αποτελεί μια σύμβαση για το τέλος του προγράμματος. Η τιμή αυτή είναι τέτοια, ώστε να μην είναι λογικά σωστή για το πρόβλημα, για παράδειγμα η τιμή 0 αποκλείεται να είναι σωστή τιμή για μια ηλικία ή για έναν μισθό.
Η τιμή αυτή συχνά αποκαλείται **τιμή φρουρός.**

Διαγράμματα ροής

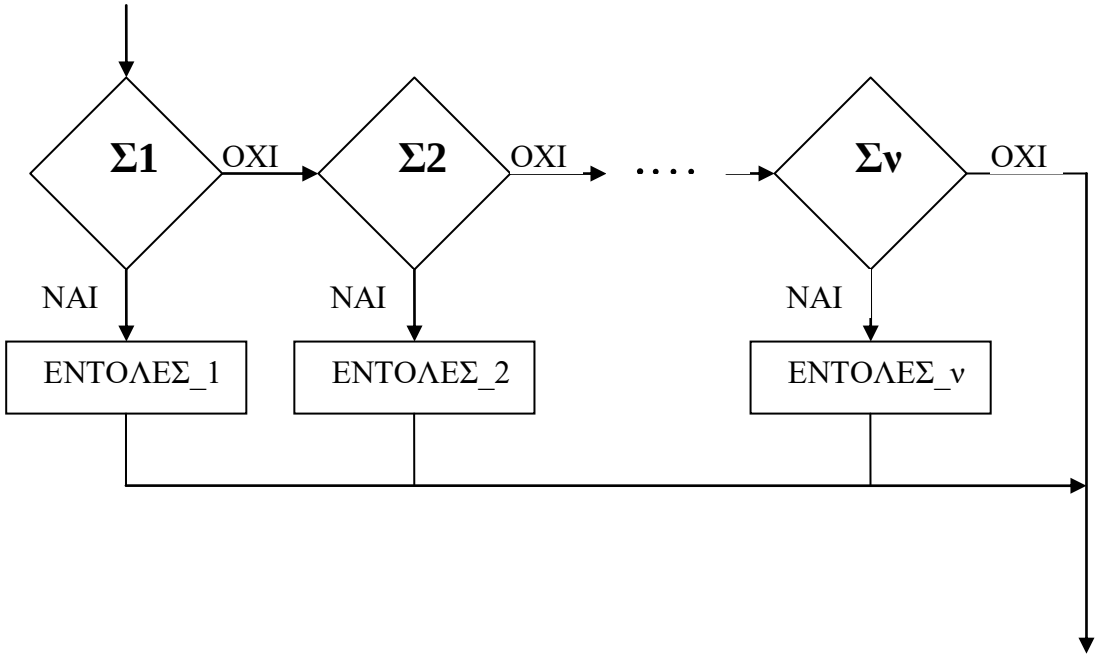
ΑΠΛΗ ΕΠΙΛΟΓΗ



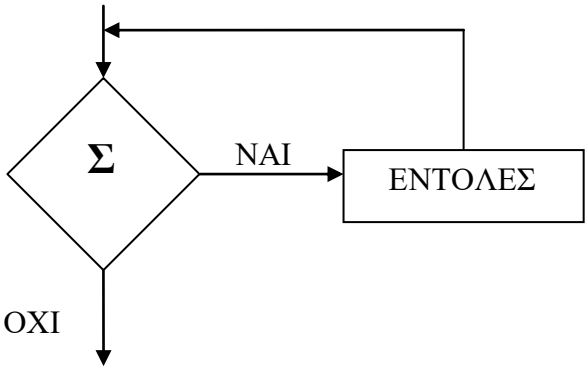
ΣΥΝΘΕΤΗ ΕΠΙΛΟΓΗ



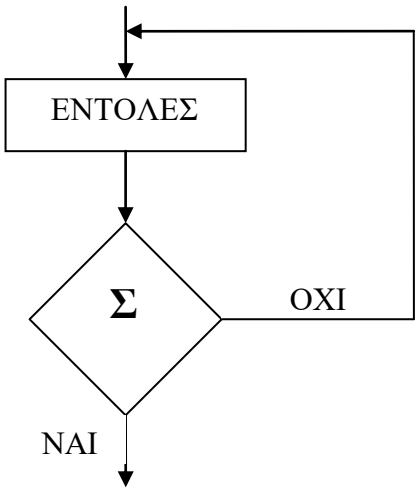
ΠΟΛΛΑΠΛΗ ΕΠΙΛΟΓΗ



ΕΠΑΝΑΛΗΠΤΙΚΗ ΔΟΜΗ

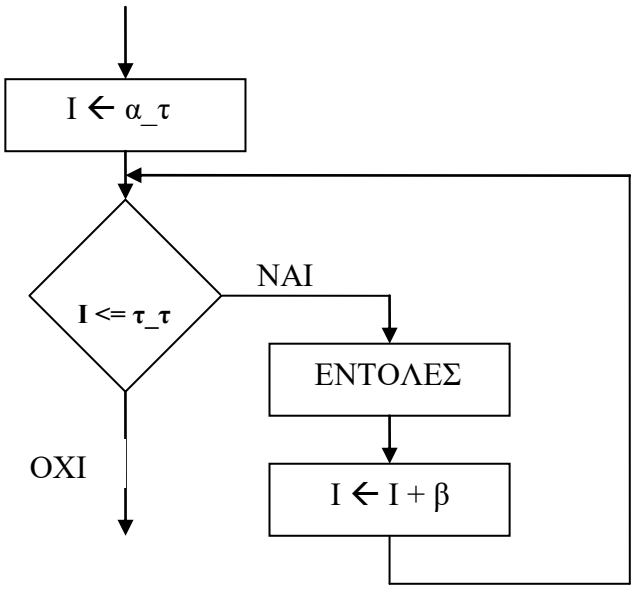


ΜΕ ΟΣΟ



ΜΕ ΜΕΧΡΙΣ_ΟΤΟΥ

ΕΠΑΝΑΛΗΠΤΙΚΗ ΔΟΜΗ ΜΕ ΓΙΑ



Ολίσθηση

Στα κυκλώματα του υπολογιστή τα δεδομένα αποθηκεύονται με δυαδική μορφή δηλ. με 0 και 1 ανεξάρτητα αν είναι αριθμοί (ακέραιοι, πραγματικοί) ή χαρακτήρες.
Έτσι ο αριθμός 17 του δεκαδικού συστήματος ισοδυναμεί με τον αριθμό 00010001 του δυαδικού συστήματος.
Αν μετακινήσουμε τα ψηφία αυτά κατά μια θέση αριστερά θα πάρουμε τον αριθμό 00100010 δηλ. τον αριθμό 34 στο δεκαδικό σύστημα.
Επίσης αν μετακινήσουμε τα ψηφία κατά μια θέση δεξιά τότε προκύπτει ο αριθμός 00001000 που ισοδυναμεί με τον αριθμό 8 του δεκαδικού συστήματος.

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

Πολλαπλασιασμός αλά Ρωσικά. Ποια η πρακτική σημασία του;

Η μέθοδος αυτή χρησιμοποιείται πρακτικά στους υπολογιστές, γιατί υλοποιείται πολύ πιο απλά απ' ότι ο γνωστός μας χειρωνακτικός τρόπος πολλαπλασιασμού που απαιτεί πολλαπλασιασμό με οποιονδήποτε ακέραιο και πρόσθεση.
Πιο συγκεκριμένα, απαιτεί πολλαπλασιασμό επί δύο, διαίρεση διά δύο και πρόσθεση. Σε αντίθεση η γνωστή μας διαδικασία πολλαπλασιασμού. Σε επίπεδο, λοιπόν, κυκλωμάτων υπολογιστή ο πολλαπλασιασμός επί δύο και η διαίρεση διά δύο μπορούν να υλοποιηθούν ταχύτατα με μία απλή εντολή ολίσθησης (shift).
Το τελευταίο γεγονός είναι ο λόγος που ο πολλαπλασιασμός αλά ρωσικά είναι προτιμότερος απ' ότι ο χειρωνακτικός τρόπος πολλαπλασιασμού δύο ακεραίων.

Αλγόριθμος Πολλαπλασιασμός_αλά_ρωσικά
Δεδομένα // Μ1,Μ2 ακέραιοι //
Ρ ←0
Όσο Μ2 > 0 **επανάλαβε**
Αν Μ2 mod 2 = 1 τότε Ρ ←Ρ+Μ1
Μ1 ←Μ1*2
Μ2 ←Μ2 div 2
Τέλος_επανάληψης
Αποτελέσματα // Ρ, το γινόμενο των ακεραίων Μ1,Μ2 //
Τέλος Πολλαπλασιασμός_αλά_ρωσικά

Δομές Δεδομένων και αλγόριθμοι 3.2

Δομές Δεδομένων

Τα δεδομένα αποθηκεύονται στον υπολογιστή είτε στην κύρια είτε στη δευτερεύουσα μνήμη του. Η αποθήκευση αυτή δε γίνεται κατά έναν τυχαίο τρόπο αλλά συστηματικά 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.
Ο κάθε κόμβος περιέχει ένα στοιχείο κάποιου συγκεκριμένου τύπου

Βασικές λειτουργίες επί των δομών δεδομένων:

- **Προσπέλαση**, πρόσβαση σε ένα κόμβο με σκοπό να εξετασθεί ή να τροποποιηθεί το περιεχόμενό του.
- **Εισαγωγή**, δηλ. η προσθήκη νέων κόμβων σε μια υπάρχουσα δομή.
- **Διαγραφή**, δηλ η αφαίρεση ενός κόμβου από τη δομή.
- **Αναζήτηση**, η προσπέλαση των κόμβων μιας δομής προκειμένου να εντοπιστούν ένας ή περισσότεροι κόμβοι που έχουν μια δεδομένη ιδιότητα.
- **Ταξινόμηση**, όπου οι κόμβοι μιας δομής διατάσσονται κατά αύξουσα ή φθίνουσα σειρά.
- **Αντιγραφή**, κατά την οποία όλοι 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**, που αποτελεί την αντίστροφη πράξη της συγχώνευσης

Ποιες λειτουργίες επί των δομών δεδομένων δεν μπορούν να χρησιμοποιηθούν σε πίνακα;
Οι λειτουργίες εισαγωγής και διαγραφής. Ο πίνακας είναι στατική δομή δεδομένων και δεν μπορεί να τροποποιηθεί το πλήθος των κόμβων (στοιχείων) της.

Σχέση Δομής Δεδομένων και Αλγορίθμου

Αλγόριθμοι + Δομές Δεδομένων = Προγράμματα

Οι δομές Δεδομένων διακρίνονται σε δύο μεγάλες κατηγορίες:

Στατικές δομές	Δυναμικές δομές
Το ακριβές μέγεθος της απαιτούμενης μνήμης καθορίζεται κατά τη στιγμή του προγραμματισμού τους και όχι κατά τη διάρκεια της εκτέλεσης του προγράμματος.	Δεν έχουν σταθερό μέγεθος, αλλά ο αριθμός των κόμβων τους μεγαλώνει ή μικραίνει καθώς στη δομή εισάγονται νέα δεδομένα ή διαγράφονται κάποια δεδομένα αντίστοιχα.
Τα στοιχεία τους αποθηκεύονται σε συνεχόμενες θέσεις μνήμης.	Δεν αποθηκεύονται σε συνεχόμενες θέσεις μνήμης αλλά στηρίζονται στην τεχνική της δυναμικής παραχώρησης μνήμης.

Υλοποιούνται με πίνακες	Υλοποιούνται με αρχεία
Αποθηκεύονται στην κύρια μνήμη	Αποθηκεύονται στη δευτερεύουσα μνήμη

Πίνακες 3.3

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

Η αναφορά στα στοιχεία ενός πίνακα γίνεται με τη χρήση του ονόματος του πίνακα ακολουθούμενου από την τιμή ενός ή περισσοτέρων δεικτών σε παρένθεση ή αγκύλη.

Ο δείκτης μπορεί να είναι μια σταθερά, μια μεταβλητή ή μια αριθμητική έκφραση με ακέραιο πάντα αποτέλεσμα.

Ένας πίνακας μπορεί να είναι μονοδιάστατος, δισδιάστατος, τρισδιάστατος και γενικά ν-διάστατος.

Ο Α[10] είναι ένας γραμμικός ή μονοδιάστατος πίνακας με 10 στοιχεία
Ο Β[5, 7] είναι ένας ορθογώνιος ή δισδιάστατος πίνακας που αποτελείται από 5 γραμμές και 7 στήλες δηλ συνολικά 35 στοιχεία.

Μπορούμε να θεωρήσουμε τον δισδιάστατο πίνακα ως έναν μονοδιάστατο που κάθε θέση του περιέχει έναν μονοδιάστατο πίνακα.

Με τη χρήση των πινάκων μειώνεται το πλήθος των μεταβλητών που χρειάζεται να ορίσουμε σε ένα αλγόριθμο. Όμως η χρήση πίνακα δικαιολογείται μόνο στην περίπτωση όπου τα δεδομένα που έχουμε εισάγει θέλουμε να τα έχουμε διαθέσιμα σε διαφορετικά σημεία του προγράμματος οπότε το μόνο που μας μένει είναι να τα αποθηκεύσουμε σε έναν πίνακα.

Εισαγωγή στοιχείων σε μονοδιάστατο πίνακα Α[Ν]	Εμφάνιση στοιχείων μονοδιάστατου πίνακα Α[Ν]
Για Ι από 1 μέχρι Ν Διάβασε Α[Ι] Τέλος_επανάληψης	Για Ι από 1 μέχρι Ν Εμφάνισε Α[Ι] Τέλος_επανάληψης

Αναζήτηση στοιχείων πίνακα 3.6

Η πιο απλή μορφή αναζήτησης στοιχείου σε πίνακα είναι η σειριακή ή γραμμική μέθοδος.
Είναι η λιγότερη αποτελεσματική και εφαρμόζεται σε περιπτώσεις όπου:

- ✓ ο πίνακας είναι μη ταξινομημένος
- ✓ ο πίνακας είναι μικρού μεγέθους
- ✓ η αναζήτηση σε ένα συγκεκριμένο πίνακα γίνεται σπάνια

Στον επόμενο αλγόριθμο αναζητείται η τιμή key στον πίνακα table ο οποίος έχει Ν στοιχεία.
Μετά την εκτέλεση του αλγορίθμου η μεταβλητή **βρέθηκε** επιστρέφει:

- την τιμή 0 αν η αναζήτηση είναι ανεπιτυχής
- τη θέση του στοιχείου στον πίνακα αν η αναζήτηση είναι επιτυχής

```
Αλγόριθμος σειριακή_αναζήτηση
Δεδομένα // N, table, Key//
I ← 1
βρέθηκε ← ψευδής
θέση ← 0
Όσο (βρέθηκε = ψευδής) και (I <= N) επανάλαβε
        Αν table[I] = Key τότε
                θέση ←I
                βρέθηκε ← αληθής
        Αλλιώς
                I ← I + 1
        Τέλος_αν
Τέλος_επανάληψης
Αποτελέσματα // βρέθηκε, θέση //

Τέλος σειριακή_αναζήτηση
```

Ταξινόμηση 3.7

Η τακτοποίηση των κόμβων μίας δομής με μια ιδιαίτερη σειρά είναι μια πολύ σημαντική λειτουργία που ονομάζεται ταξινόμηση.
Σκοπός της είναι η ευκολότερη αναζήτηση των στοιχείων του ταξινομημένου πίνακα.
Η χρησιμότητα της ταξινόμησης αποδεικνύεται στην πράξη σε αναρίθμητες περιπτώσεις αναζήτησης αριθμητικών ή αλφαριθμητικών δεδομένων όπως σε

- ✓ Λεξικά
- ✓ Τηλεφωνικούς καταλόγους
- ✓ Καταλόγους φόρου εισοδήματος
- ✓ Βιβλιοθήκες

Δοθέντων των στοιχείων $\alpha_1,\alpha_2,\ldots,\alpha_n$ η ταξινόμηση συνίσταται στη μετάθεση της θέσης των στοιχείων ώστε να τοποθετηθούν σε μια σειρά $\alpha_{\kappa 1},\alpha_{\kappa 2},\ldots,\alpha_{\kappa n}$ έτσι ώστε δοθείσης μιας συνάρτησης διάταξης f να ισχύει $f(\alpha_{\kappa 1})<=f(\alpha_{\kappa 2})<=\ldots,f(\alpha_{\kappa n})$

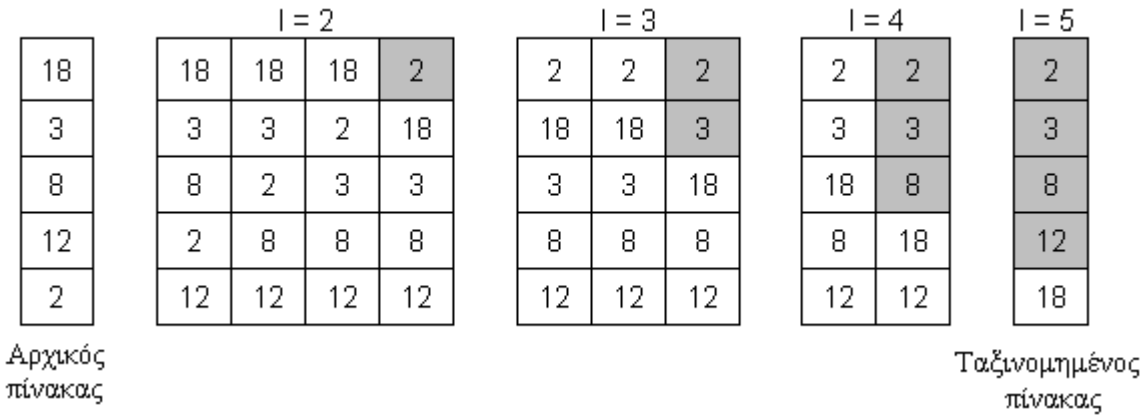
Ταξινόμηση ευθείας ανταλλαγής (φυσαλίδας)

Η μέθοδος αυτή βασίζεται στην αρχή της σύγκρισης και ανταλλαγής ζευγών γειτονικών στοιχείων μέχρις ότου διαταχθούν όλα τα στοιχεία.
Η ταξινόμηση φυσαλίδας είναι ο πιο απλός και ταυτόχρονα ο πιο αργός αλγόριθμος.
Πιο γρήγοροι αλγόριθμοι είναι της ταξινόμησης με επιλογή, με παρεμβολή και η γρήγορη ταξινόμηση.
Παρακάτω δίνεται η ταξινόμηση μονοδιάστατου πίνακα σε αύξουσα σειρά.

```
Αλγόριθμος ταξινόμηση_φυσαλίδας
Δεδομένα // table, N //
Για I από 2 μέχρι N
        Για K από N μέχρι I με_βήμα –1
                Αν table[K-1] > table[K] τότε
                        T ← table[K-1]
                        table[K-1] = table[K]
                        table[K] = T
                Τέλος_αν
        Τέλος_επανάληψης
Τέλος_επανάληψης
Αποτελέσματα // table //
Τέλος ταξινόμηση_φυσαλίδας
```

Ο αλγόριθμος της ταξινόμησης σε φθίνουσα σειρά προκύπτει από τον παραπάνω αν στη συνθήκη αντί του **table[K-1] > table[K]** βάλουμε **table[K-1] < table[K]**

Στο παρακάτω σχήμα φαίνεται πως από έναν μονοδιάστατο πίνακα μη ταξινομημένο προκύπτει με διαδοχικές αντιμεταθέσεις ένας ταξινομημένος πίνακας



18	18	18	18	2
3	3	3	2	18
8	8	2	3	3
12	2	8	8	8
2	12	12	12	12

2	2	2
18	18	3
3	3	18
8	8	8
12	12	12

2	2
3	3
18	8
8	18
12	12

2
3
8
12
18

Εισαγωγή στον προγραμματισμό 6.3

Φυσικές και τεχνητές γλώσσες

Μια γλώσσα προσδιορίζεται από:
- το αλφάβητο
- το λεξιλόγιο
- τη γραμματική
- τη σημασιολογία

Το αλφάβητο
Αλφάβητο μιας γλώσσας λέγεται το σύνολο των στοιχείων που χρησιμοποιείται από μια γλώσσα

Το αλφάβητο μιας γλώσσας περιλαμβάνει:
a. Τα γράμματα του αλφαβήτου (Αγγλικού / Ελληνικού)
b. Τα αριθμητικά ψηφία (0-9)
c. Σημεία στίξης
d. Ειδικοί χαρακτήρες

Το λεξιλόγιο
Το λεξιλόγιο αποτελείται από ένα υποσύνολο όλων των ακολουθιών που δημιουργούνται από τα στοιχεία του αλφαβήτου δηλ από τις λέξεις που είναι δεκτές από τη γλώσσα.
Για παράδειγμα η λέξη **Διάβασε** είναι αποδεκτή αλλά η λέξη **Εισήγαγε** όχι.

Η γραμματική
Η γραμματική αποτελείται από το **τυπικό** και το **συντακτικό.**
Το **τυπικό** ορίζει τις μορφές με τις οποίες μια λέξη είναι αποδεκτή
Το **συντακτικό** καθορίζει τη νομιμότητα της διάταξης και της σύνδεσης των λέξεων της γλώσσας για τη δημιουργία σωστών εντολών
Για παράδειγμα η εντολή «Για 1 μέχρι 10» περιέχει λάθος

Η σημασιολογία
Η σημασιολογία καθορίζει το νόημα των λέξεων και των εντολών που χρησιμοποιούνται από μια γλώσσα.
Για παράδειγμα η εντολή **Διάβασε** σημαίνει ότι ο υπολογιστής θα περιμένει να δώσουμε μια τιμή από το πληκτρολόγιο

Οι φυσικές γλώσσες εξελίσσονται συνέχεια (νέες λέξεις δημιουργούνται και κανόνες γραμματικής αλλάζουν) σε αντίθεση με τις τεχνητές γλώσσες που είναι στάσιμες.

Τεχνικές σχεδίασης προγραμμάτων 6.4

Ιεραρχική σχεδίαση προγράμματος

Η τεχνική της Ιεραρχικής σχεδίασης και επίλυσης (top-down) περιλαμβάνει τον καθορισμό των βασικών λειτουργιών ενός προγράμματος και στη συνέχεια τη διάσπαση των λειτουργιών αυτών σε όλο και μικρότερες λειτουργίες μέχρι το τελευταίο επίπεδο που οι λειτουργίες είναι πολύ απλές ώστε να επιλυθούν εύκολα.

Σκοπός της **Ιεραρχικής σχεδίασης** είναι η διάσπαση του προβλήματος σε μια σειρά από απλούστερα υποπροβλήματα τα οποία είναι εύκολο να επιλυθούν.

Τμηματικός προγραμματισμός

Η Ιεραρχική σχεδίαση προγράμματος υλοποιείται με τον Τμηματικό προγραμματισμό. Μετά την ανάλυση του προβλήματος σε υποπροβλήματα, κάθε υποπρόβλημα αποτελεί ανεξάρτητη ενότητα που γράφεται ξεχωριστά από τα υπόλοιπα τμήματα του προγράμματος.

Ο Τμηματικός προγραμματισμός

- διευκολύνει τη δημιουργία του προγράμματος
- μειώνει τα λάθη
- επιτρέπει την ευκολότερη παρακολούθηση και συντήρηση του προγράμματος

Δομημένος προγραμματισμός

Ο Δομημένος προγραμματισμός στηρίζεται στη χρήση τριών και μόνο στοιχειωδών λογικών δομών:

1. τη δομή ακολουθίας
2. τη δομή της επιλογής
3. τη δομή επανάληψης

Όλα τα προγράμματα μπορούν να γραφούν χρησιμοποιώντας μόνο αυτές τις τρεις δομές καθώς και συνδυασμό τους. Κάθε πρόγραμμα και κάθε ενότητα προγράμματος έχει μόνο μια είσοδο και μόνο μια έξοδο.

Πλεονεκτήματα του Δομημένου προγραμματισμού

- Δημιουργία απλούστερων προγραμμάτων
- Διευκόλυνση ανάλυσης του προγράμματος σε τμήματα
- Περιορισμός των λαθών κατά την ανάπτυξη του προγράμματος
- Διευκόλυνση στην ανάγνωση και κατανόηση του προγράμματος
- Ευκολότερη διόρθωση και συντήρηση

Προγραμματιστικά περιβάλλοντα 6.7

Κάθε πρόγραμμα που γράφεται σε μια γλώσσα προγραμματισμού πρέπει να μετατραπεί σε γλώσσα μηχανής για να αναγνωριστεί από τον υπολογιστή.
Τη μετατροπή αυτή την κάνουν ειδικά μεταφραστικά προγράμματα.

μεταγλωττιστές (compiler)	**διερμηνευτές** (interpreters)
Δέχεται στην είσοδο ένα αρχικό πρόγραμμα, το **πηγαίο** και το μεταγλωττίζει. Αν υπάρχουν λάθη τα εμφανίζει συνολικά στο τέλος. Μετά τη διόρθωση των λαθών ο μεταγλωττιστής παράγει ένα πρόγραμμα που λέγεται **αντικείμενο**. Το αντικείμενο πρόγραμμα πρέπει να συνδεθεί με τις **βιβλιοθήκες** για να παραχθεί το τελικό πρόγραμμα που λέγεται **εκτελέσιμο**.	Διαβάζει μια προς μια τις εντολές του **πηγαίου** προγράμματος και τις εκτελεί. Αν προκύψει κάποιο λάθος το εμφανίζει την ίδια στιγμή. Πρέπει να διορθώσουμε το λάθος για να συνεχίσουμε την εκτέλεση του προγράμματος. Όταν φτάσουμε στην τελευταία εντολή το πρόγραμμα θα είναι απαλλαγμένο από λάθη.
Πλεονέκτημα είναι η γρήγορη εκτέλεση του εκτελέσιμου προγράμματος	Έχει το πλεονέκτημα της άμεσης εκτέλεσης και διόρθωσης του προγράμματος
Το μειονέκτημα είναι ότι προτού χρησιμοποιηθεί το πρόγραμμα πρέπει πρώτα να μεταγλωττιστεί και να συνδεθεί	Έχει το μειονέκτημα ότι η εκτέλεση του προγράμματος καθίσταται πιο αργή

Πηγαίο είναι το αρχικό πρόγραμμα που φτιάχνουμε σε μια γλώσσα υψηλού επιπέδου

Συντάκτης είναι το πρόγραμμα μέσω του οποίου φτιάχνουμε και διορθώνουμε το πηγαίο πρόγραμμα

Μεταγλωττιστής είναι ένα ειδικό πρόγραμμα που μετατρέπει το πηγαίο σε γλώσσα μηχανής

Διερμηνευτής είναι ένα ειδικό πρόγραμμα που μετατρέπει το πηγαίο σε γλώσσα μηχανής και ταυτόχρονα το εκτελεί

Βιβλιοθήκες είναι ένα σύνολο έτοιμων προγραμμάτων που χρησιμοποιούνται σε συνδυασμό με το πηγαίο πρόγραμμα

Συνδέτης είναι το πρόγραμμα που συνδέει το αντικείμενο πρόγραμμα με κάποια προγράμματα από τις βιβλιοθήκες. Το αποτέλεσμα του συνδέτη είναι το εκτελέσιμο πρόγραμμα.

Εκτελέσιμο είναι το πρόγραμμα που προκύπτει μετά τη μεταγλώττιση και τη σύνδεση και είναι το πρόγραμμα που καταλαβαίνει και εκτελεί ο υπολογιστής.

Πηγαίο → Μεταγλωττιστής → Αντικείμενο → Συνδέτης → Εκτελέσιμο

Τα σύγχρονα προγραμματιστικά περιβάλλοντα περιέχουν με ενιαίο τρόπο τα εξής:

Συντάκτη - μεταγλωττιστή - συνδέτη

Είδη λαθών

Τα **συντακτικά** λάθη δημιουργούνται 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, η παράληψη δήλωσης δεδομένων, η χρήση μιας δομής ελέγχου χωρίς την εντολή τερματισμού της.
Ένα λάθος που προκαλείται κατά τη συγγραφή του προγράμματος, ανιχνεύεται από το μεταγλωττιστή, ο οποίος εμφανίζει προς το προγραμματιστή κάποιο προειδοποιητικό μήνυμα. Αν το πρόγραμμα περιέχει ένα λάθος αυτής της μορφής, δεν επιτρέπεται η εκτέλεσή του, μέχρι να το διορθώσει ο προγραμματιστής.

Τα **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.

Βασικά στοιχεία προγραμματισμού 7

Αλφάβητο της Γλώσσας
Τα γράμματα του Ελληνικού και Αγγλικού αλφαβήτου
Τα ψηφία 0-9
Οι ειδικοί χαρακτήρες + - * / = ( ) . , ’ ! & κενό

Τύποι δεδομένων

Ακέραιος τύπος	Περιλαμβάνει τους ακέραιους
Πραγματικός τύπος	Περιλαμβάνει τους πραγματικούς
Χαρακτήρας ή αλφαριθμητικός	Περιέχει οποιοδήποτε χαρακτήρα παράγεται από το πληκτρολόγιο.
Λογικός	Δέχεται μόνο δύο τιμές **Αληθής** και **Ψευδής**

Σταθερές
Είναι προκαθορισμένες τιμές που δεν μεταβάλλονται κατά τη διάρκεια εκτέλεσης του προγράμματος. Η χρήση τους κάνει πιο κατανοητό το πρόγραμμα και κατά συνέπεια ευκολότερο να διορθωθεί και να συντηρηθεί.

Δήλωση σταθερών

```
Σταθερές
  Π = 3.14159
  ΦΠΑ = 0.18
  Επώνυμο = 'Παπαδούλης'
```

Μεταβλητές

Παριστάνουν ποσότητες που οι τιμές τους μπορούν να μεταβάλλονται. Αντιστοιχούν σε συγκεκριμένες θέσεις μνήμης και οι τιμές των μεταβλητών είναι το περιεχόμενο αυτών των θέσεων.
Ο τύπος της μεταβλητής σε αντίθεση με την τιμή της δεν μπορεί να αλλάζει.

Δήλωση μεταβλητών

```
Μεταβλητές
  Πραγματικές: ΜΟ, Α
  Ακέραιες: μισθός, θερμ[30]
  Λογικές: βρέθηκε
  Χαρακτήρες: επώνυμο
```

Τα ονόματα των σταθερών και των μεταβλητών μπορούν να αποτελούνται από γράμματα, ψηφία (0-9) και την κάτω παύλα.
Επίσης πρέπει να **αρχίζουν με γράμμα**.
Ο τύπος της μεταβλητής παραμένει αναλλοίωτος.
Συνιστάται τα ονόματα των μεταβλητών και των σταθερών να ανάγουν στο περιεχόμενό τους.

Αριθμητικοί τελεστές

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

Αριθμητικές εκφράσεις

Για τη σύνταξη μιας αριθμητικής έκφρασης χρησιμοποιούνται σταθερές, μεταβλητές, συναρτήσεις, αριθμητικοί τελεστές και παρενθέσεις. Οι αριθμητικές εκφράσεις υλοποιούν απλές ή σύνθετες μαθηματικές πράξεις.
Πρέπει όλες οι μεταβλητές που εμφανίζονται σε μια έκφραση να έχουν κάποια τιμή.

Συναρτήσεις

ΗΜ(Χ)	Ημίτονο

ΣΥΝ(Χ)	Συνημίτονο
ΕΦ(Χ)	Εφαπτομένη
Τ_Ρ(Χ)	Τετραγωνική ρίζα
ΛΟΓ(Χ)	Φυσικός λογάριθμος
Ε(Χ)	e^x
Α_Μ(Χ)	Ακέραιο μέρος Χ
Α_Τ(Χ)	Απόλυτη τιμή Χ

Εντολή εκχώρησης
Χρησιμοποιείται για την απόδοση τιμών στις μεταβλητές κατά τη διάρκεια εκτέλεσης του προγράμματος.
Παραδείγματα
Α ← 12
Όνομα ← 'Γιώργος'
Εμβαδόν ← α * β

Σε μια εντολή εκχώρησης η μεταβλητή και η έκφραση πρέπει να είναι του ίδιου τύπου

Δομή προγράμματος 7.10

- Η πρώτη εντολή κάθε προγράμματος είναι υποχρεωτικά η επικεφαλίδα του προγράμματος η οποία είναι η λέξη ΠΡΟΓΡΑΜΜΑ ακολουθούμενο από το όνομα του προγράμματος το οποίο πρέπει να υπακούει στους κανόνες δημιουργίας ονομάτων.

- Ακολουθεί το τμήμα δήλωσης των σταθερών και των μεταβλητών.

- Ακολουθεί το κύριο μέρος του προγράμματος που περιλαμβάνει όλες τις εκτελέσιμες εντολές οι οποίες περιλαμβάνονται ανάμεσα στις λέξεις ΑΡΧΗ και ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ.

- Αν το πρόγραμμα χρησιμοποιεί διαδικασίες γράφονται μετά το ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
- Κάθε εντολή γράφεται σε ξεχωριστή γραμμή. Αν πρέπει να συνεχιστεί και στην επόμενη γραμμή τότε ο πρώτος χαρακτήρας αυτής της γραμμής πρέπει να είναι ο &.

- Αν ο πρώτος χαρακτήρας είναι το ! σημαίνει ότι αυτή η γραμμή περιέχει σχόλια και όχι εκτελέσιμες εντολές.

Εντολές επιλογής 8.1

Λογική Έκφραση
Για τη σύνταξη μιας λογικής έκφρασης ή συνθήκης χρησιμοποιούνται σταθερές, μεταβλητές, αριθμητικές παραστάσεις, συγκριτικοί και λογικοί τελεστές, καθώς και παρενθέσεις.
Στις λογικές εκφράσεις γίνεται σύγκριση της τιμής μίας έκφρασης, που βρίσκεται

αριστερά από το συγκριτικό τελεστή με την τιμή μιας άλλης έκφρασης που βρίσκεται δεξιά. Το αποτέλεσμα είναι μία λογική τιμή **ΑΛΗΘΗΣ ή ΨΕΥΔΗΣ.**

Οι χρησιμοποιούμενοι συγκριτικοί τελεστές παρουσιάζονται στον επόμενο πίνακα.

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

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

Οι συγκρίσεις γίνονται σε δεδομένα αριθμητικά, αλφαριθμητικά και λογικά.

- Η σύγκριση μεταξύ δύο αριθμών γίνεται με προφανή τρόπο. Στην περίπτωση των πραγματικών αριθμών θεωρούμε ότι οι αριθμοί μπορούν να έχουν άπειρο αριθμό ψηφίων.

- Η σύγκριση ατομικών χαρακτήρων στηρίζεται στην αλφαβητική σειρά, για παράδειγμα το 'α' θεωρείται μικρότερο από το 'β'.

- Η σύγκριση αλφαριθμητικών δεδομένων βασίζεται στη σύγκριση χαρακτήρα προς χαρακτήρα σε κάθε θέση μέχρις ότου βρεθεί κάποια διαφορά, για παράδειγμα η λέξη 'κακός' θεωρείται μικρότερη από τη λέξη 'καλός' αφού το γράμμα κ προηγείται του γράμματος λ.

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

Σύνθετες Εκφράσεις

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

Παραδείγματα

0<Χ<5 γράφεται Χ>0 ΚΑΙ Χ<5

Χ=1 ή 2 ή 3 γράφεται Χ=1 Ή Χ=2 Ή Χ=3

Η ιεραρχία των λογικών τελεστών είναι μικρότερη των αριθμητικών.

Εντολές επανάληψης 8.2

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

Συγκεκριμένα:

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

Πίνακες 9

Το όνομα του πίνακα καθορίζει μία ομάδα διαδοχικών θέσεων στη μνήμη. Κάθε συγκεκριμένη θέση μνήμης καλείται στοιχείο του πίνακα και προσδιορίζεται από την τιμή ενός δείκτη.

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

Το όνομα του πίνακα μπορεί να είναι οποιοδήποτε δεκτό όνομα της ΓΛΩΣΣΑΣ και ο δείκτης είναι μία ακέραια έκφραση, σταθερή ή μεταβλητή που περικλείεται μέσα στα σύμβολα [και].
Το στοιχείο Α[Ι] αναφέρεται στο Ι-στό στοιχείο του πίνακα.

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. Τη δέσμευση των θέσεων πραγματοποιεί ο μεταγλωττιστής κατά την έναρξη εκτέλεσης του προγράμματος.

Πότε πρέπει να χρησιμοποιούνται πίνακες

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

Πέρα από τα πλεονεκτήματα που αναφέρθηκαν, υπάρχουν και δύο μειονεκτήματα από τη χρήση πινάκων.

Οι πίνακες απαιτούν μνήμη.
Κάθε πίνακας δεσμεύει από την αρχή του προγράμματος πολλές θέσεις μνήμης. Σε

ένα μεγάλο και σύνθετο πρόγραμμα η άσκοπη χρήση μεγάλων πινάκων μπορεί να οδηγήσει ακόμη και σε αδυναμία εκτέλεσης του προγράμματος.

Οι πίνακες περιορίζουν τις δυνατότητες του προγράμματος.
Οι πίνακες είναι στατικές δομές και το μέγεθος τους πρέπει να δηλώνεται στην αρχή του προγράμματος, και παραμένει υποχρεωτικά σταθερό κατά την εκτέλεση του προγράμματος.

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

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

Τυπικές επεξεργασίες πινάκων

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

Υπολογισμός αθροισμάτων στοιχείων του πίνακα
Πολύ συχνά απαιτείται ο υπολογισμός του αθροίσματος στοιχείων του πίνακα που έχουν κοινά χαρακτηριστικά για παράδειγμα βρίσκονται στην ίδια στήλη ή στην ίδια γραμμή.

Εύρεση του μέγιστου ή του ελάχιστου στοιχείου
Αν ο πίνακας δεν είναι ταξινομημένος, τότε πρέπει να συγκριθούν τα στοιχεία ένα προς ένα, για να βρεθεί το μέγιστο ή το ελάχιστο. Αν ο πίνακας είναι ταξινομημένος, τότε προφανώς το μέγιστο και το ελάχιστο βρίσκονται στα δύο ακριανά στοιχείαΤΟΥ πίνακα.

Ταξινόμηση των στοιχείων του πίνακα
Η μέθοδος ταξινόμησης 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 αλγόριθμου εξαρτάται κυρίως από το πλήθος των στοιχείων του πίνακα και την αρχική τους διάταξη, αν δηλαδή ο πίνακας είναι τελείως αταξινόμητος ή μερικώς ταξινομημένος.

Αναζήτηση ενός στοιχείου του πίνακα

Δύο είναι οι πλέον διαδεδομένοι αλγόριθμοι αναζήτησης:

- Η σειριακή αναζήτηση
- Η δυαδική αναζήτηση

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

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

Υποπρογράμματα - Τμηματικός προγραμματισμός 10.1

Όταν τα προβλήματα είναι αρκετά απλά μπορούν να αναπτυχθούν σωστά σε ένα και μόνο πρόγραμμα. Ο καλύτερος τρόπος για να αντιμετωπισθούν σύνθετα προβλήματα και να γραφούν τα αντίστοιχα προγράμματα είναι η ιεραρχική προσέγγιση, η ανάπτυξη του προγράμματος από επάνω προς τα κάτω (top-down).

Κάθε πρόβλημα διαιρείται σε μικρότερα επιμέρους προβλήματα και κάθε ένα από αυτά τα προγράμματα διαιρείται σε ακόμα απλούστερα και μικρότερα. Στο τέλος τα επιμέρους υποπροβλήματα είναι αρκετά απλά ώστε οι αντίστοιχοι αλγόριθμοι και τα αντίστοιχα τμήματα προγράμματος να μπορούν να σχεδιασθούν και να γραφούν εύκολα.
Ο τελικός αλγόριθμος του προβλήματος ανάγεται σε πολλούς απλούστερους επί μέρους αλγόριθμους και το τελικό πρόγραμμα σε πολλά απλούστερα τμήματα προγράμματος.

Η τεχνική του τμηματικού προγραμματισμού είναι ένα από τα βασικότερα συστατικά του δομημένου προγραμματισμού ο οποίος εξασφαλίζει σε μεγάλο βαθμό την επιτυχή και εύκολη δημιουργία σωστών προγραμμάτων.

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

Ιδιότητες που πρέπει να διακρίνουν τα υποπρογράμματα 10.2

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.

Κάθε υποπρόγραμμα πρέπει να είναι ανεξάρτητο από τα άλλα.
Αυτό σημαίνει ότι κάθε υποπρόγραμμα μπορεί να σχεδιαστεί, να αναπτυχθεί και να συντηρηθεί αυτόνομα χωρίς να επηρεαστούν άλλα υποπρογράμματα. Στην πράξη βέβαια η απόλυτη ανεξαρτησία είναι δύσκολο να επιτευχθεί.

Κάθε υποπρόγραμμα πρέπει να μην είναι πολύ μεγάλο.
Η έννοια του μεγάλου προγράμματος είναι υποκειμενική, αλλά πρέπει κάθε υποπρόγραμμα να είναι τόσο, ώστε να είναι εύκολα κατανοητό για να μπορεί να ελέγχεται. Γενικά κάθε υποπρόγραμμα πρέπει να εκτελεί μόνο μία λειτουργία. Αν εκτελεί περισσότερες λειτουργίες, τότε συνήθως μπορεί και πρέπει να διασπαστεί σε ακόμη μικρότερα υποπρογράμματα.

Πλεονεκτήματα του τμηματικού προγραμματισμού 10.3

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

Διευκολύνει την κατανόηση και διόρθωση του προγράμματος.
Ο χωρισμός του προγράμματος σε μικρότερα αυτοτελή τμήματα επιτρέπει τη γρήγορη διόρθωση ενός συγκεκριμένου τμήματος του χωρίς οι αλλαγές αυτές να επηρεάσουν όλο το υπόλοιπο πρόγραμμα.

Επίσης διευκολύνει οποιονδήποτε χρειαστεί να διαβάσει και να κατανοήσει τον τρόπο που λειτουργεί το πρόγραμμα. Όπως έχει πολλές φορές τονιστεί αυτό είναι πολύ σημαντικό χαρακτηριστικό του σωστού προγραμματισμού, αφού ένα μεγάλο πρόγραμμα στον κύκλο της ζωής του χρειάζεται να συντηρηθεί από διαφορετικούς προγραμματιστές.

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

Επεκτείνει τις δυνατότητες των γλωσσών προγραμματισμού.
Ένα υποπρόγραμμα που έχει γραφεί μπορεί να χρησιμοποιηθεί πολύ εύκολα και σε άλλα προγράμματα. Από τη στιγμή που έχει δημιουργηθεί, η χρήση του δεν διαφέρει από τη χρήση των ενσωματωμένων συναρτήσεων που παρέχει η γλώσσα προγραμματισμού, όπως για τον υπολογισμό του ημίτονου ή του συνημίτονου.

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.

Παράμετροι 10.4

Τα υποπρογράμματα ενεργοποιούνται από κάποιο άλλο πρόγραμμα ή υποπρόγραμμα για να εκτελέσουν συγκεκριμένες λειτουργίες.
Κάθε υποπρόγραμμα για να ενεργοποιηθεί καλείται, όπως λέγεται, από ένα άλλο υποπρόγραμμα ή το αρχικό πρόγραμμα, το οποίο ονομάζεται κύριο πρόγραμμα.

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

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

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

Διαδικασίες και συναρτήσεις 10.5

Υπάρχουν δύο ειδών υποπρογράμματα, οι **διαδικασίες** και οι **συναρτήσεις**.

Οι διαδικασίες μπορούν να εκτελέσουν οποιαδήποτε λειτουργία από αυτές που μπορεί να εκτελέσει ένα πρόγραμμα. Να εισάγουν δεδομένα, να εκτελέσουν υπολογισμούς, να μεταβάλλουν τις τιμές των μεταβλητών και να τυπώσουν αποτελέσματα.

Με τη χρήση των παραμέτρων αυτές τις τιμές μπορούν να τις μεταφέρουν και στα άλλα υποπρογράμματα. Αντίθετα η λειτουργία των συναρτήσεων είναι πιο περιορισμένη.

Οι συναρτήσεις υπολογίζουν μόνο μία τιμή, αριθμητική, χαρακτήρα ή λογική και μόνο αυτήν επιστρέφουν στο υποπρόγραμμα που την κάλεσε.

Οι συναρτήσεις μοιάζουν με τις συναρτήσεις των μαθηματικών και η χρήση τους είναι όμοια με τη χρήση των ενσωματωμένων συναρτήσεων που υποστηρίζει η γλώσσα προγραμματισμού.

Ο τρόπος κλήσης καθώς και ο τρόπος σύνταξης των δύο αυτών τύπων των υποπρογραμμάτων είναι διαφορετικός.
Τόσο οι συναρτήσεις όσο και οι διαδικασίες τοποθετούνται μετά το τέλος του κυρίου προγράμματος.

Οι συναρτήσεις εκτελούνται απλά με την εμφάνιση του ονόματος τους σε οποιαδήποτε έκφραση
Για να εκτελεστούν οι διαδικασίες χρησιμοποιείται η ειδική εντολή ΚΑΛΕΣΕ και το όνομα της διαδικασίας.

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

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

Ορισμός και κλήση συναρτήσεων

Κάθε συνάρτηση έχει την ακόλουθη δομή.

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

Το όνομα της συνάρτησης είναι οποιοδήποτε έγκυρο όνομα της ΓΛΩΣΣΑΣ.
Η λίστα παραμέτρων είναι μια λίστα μεταβλητών, των οποίων οι τιμές μεταβιβάζονται στη συνάρτηση κατά την κλήση.

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

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

Στο παρακάτω πρόγραμμα αυτό γίνεται με το
Εμβαδό_κύκλου <- Π*R^2

Κάθε συνάρτηση εκτελείται, όπως ακριβώς εκτελούνται οι ενσωματωμένες συναρτήσεις της γλώσσας.

Απλώς αναφέρεται το όνομα της σε μια έκφραση ή σε μία εντολή και επιστρέφεται η τιμή της.
Στο πρόγραμμα η συνάρτηση εκτελείται με την εντολή
Ε<-Εμβαδό_κύκλου(R).

Κάθε διαδικασία έχει την ακόλουθη δομή.

```
ΔΙΑΔΙΚΑΣΙΑ Όνομα (λίστα παραμέτρων)
Τμήμα δηλώσεων
ΑΡΧΗ
        Εντολές…
ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ
```

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

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

Η λίστα παραμέτρων δεν είναι υποχρεωτική.

Να γραφεί πρόγραμμα, που να υπολογίζει το εμβαδόν του κύκλου από την ακτίνα του.
Λύση

Αν και το πρόγραμμα είναι πολύ απλό και μπορεί κάλλιστα να γραφεί χωρίς τη χρήση υποπρογραμμάτων, ας το διασπάσουμε σε τρία υποπρογράμματα που εκτελούν τις τρεις παρακάτω λειτουργίες.

a) Διαβάζει τα δεδομένα που είναι η ακτίνα

b) Υπολογίζει το εμβαδόν ($E = \pi.r^2$)

c) Τυπώνει το αποτέλεσμα, που είναι το εμβαδόν Ε

```
ΠΡΟΓΡΑΜΜΑ Κύκλος
ΜΕΤΑΒΛΗΤΕΣ
        ΠΡΑΓΜΑΤΙΚΕΣ : R, E
ΑΡΧΗ
        ΚΑΛΕΣΕ Είσοδος_δεδομένων(R)
        E <- Εμβαδό_κύκλου(R)
        ΚΑΛΕΣΕ Εκτύπωση (E)
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Το κύριο πρόγραμμα καλεί όλα τα υποπρογράμματα. Η ροή του προγράμματος διακόπτεται και εκτελούνται οι εντολές του υποπρογράμματος. Αφού τελειώσει το υποπρόγραμμα συνεχίζεται το κύριο πρόγραμμα με την επόμενη εντολή

```
ΔΙΑΔΙΚΑΣΙΑ Είσοδος_δεδομένων (Αριθμός)
ΜΕΤΑΒΛΗΤΕΣ
        ΠΡΑΓΜΑΤΙΚΕΣ: Αριθμός
ΑΡΧΗ
        ΑΡΧΗ ΕΠΑΝΑΛΗΨΗΣ
                ΓΡΑΨΕ 'Δώσε την ακτίνα'
                ΔΙΑΒΑΣΕ Αριθμός
        ΜΕΧΡΙΣ_ΟΤΟΥ Αριθμός > 0
ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ
```

Η ΔΙΑΔΙΚΑΣΙΑ αυτή διαβάζει την ακτίνα και την επιστρέφει μέσω της παραμέτρου Αριθμός στο κύριο πρόγραμμα. Αφού το υποπρόγραμμα πρέπει να διαβάζει δεδομένα, υλοποιείται με διαδικασία και όχι με συνάρτηση

```
ΣΥΝΑΡΤΗΣΗ Εμβαδό_κύκλου(R) : ΠΡΑΓΜΑΤΙΚΗ
ΣΤΑΘΕΡΕΣ
        Π=3.14
ΜΕΤΑΒΛΗΤΕΣ
        ΠΡΑΓΜΑΤΙΚΕΣ: R
ΑΡΧΗ
        Εμβαδό_κύκλου <- Π*R^2
ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ
```

Το κύριο πρόγραμμα πριν την κλήση της συνάρτησης γνωρίζει την τιμή της μεταβλητής R. Κατά την κλήση μεταβιβάζεται αυτή η τιμή στην αντίστοιχη μεταβλητή R της συνάρτησης. Η συνάρτηση υπολογίζει το εμβαδόν του κύκλου και το αποτέλεσμα αυτό εκχωρείται στο όνομα της συνάρτησης. Με το τέλος της συνάρτησης γίνεται επιστροφή στο κύριο πρόγραμμα, όπου η τιμή του εμβαδού εκχωρείται στη μεταβλητή Ε.

```
ΔΙΑΔΙΚΑΣΙΑ Εκτύπωση (Αποτέλεσμα)
ΜΕΤΑΒΛΗΤΕΣ
        ΠΡΑΓΜΑΤΙΚΕΣ : Αποτέλεσμα
ΑΡΧΗ
        ΓΡΑΨΕ 'Το εμβαδόν είναι :', Αποτέλεσμα
ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ
```

Η ΔΙΑΔΙΚΑΣΙΑ αυτή τυπώνει το αποτέλεσμα το οποίο παίρνει σαν είσοδο από το κύριο πρόγραμμα μέσω της παραμέτρου Ε. Εφόσον απαιτείται από αυτό το υποπρόγραμμα εκτύπωση, πρέπει να υλοποιηθεί με Διαδικασία και όχι με συνάρτηση.

Πραγματικές και τυπικές παράμετροι 10.5.3

Να γραφεί μια διαδικασία η οποία δέχεται στην είσοδο δύο τιμές και υπολογίζει και επιστρέφει το άθροισμα και τη διαφορά τους.

```
ΠΡΟΓΡΑΜΜΑ Παράδειγμα
……..
Α ← 5
Β ← 7
ΚΑΛΕΣΕ Πράξεις( Α, Β, Διαφ1, Αθρ1 )
…….
Α ← 9
Β ← 6
ΚΑΛΕΣΕ Πράξεις(Α, Β, Διαφ2, Αθρ2)
……….

ΔΙΑΔΙΚΑΣΙΑ Πράξεις( Χ, Υ, Διαφορά, Άθροισμα )
ΜΕΤΑΒΛΗΤΕΣ
  ΠΡΑΓΜΑΤΙΚΕΣ : Χ, Υ, Διαφορά, Άθροισμα
ΑΡΧΗ
Διαφορά ← Χ -  Υ
Άθροισμα ← Χ + Υ
ΤΕΛΟΣ ΔΙΑΔΙΚΑΣΙΑΣ
```

Οι μεταβλητές Α, Β, Διαφ1, Αθρ1, Διαφ2, Αθρ2 είναι μεταβλητές του προγράμματος Παράδειγμα και αποτελούν τις **πραγματικές** παραμέτρους, ενώ οι μεταβλητές Χ,Υ, Διαφορά, Άθροισμα είναι μεταβλητές της διαδικασίας Πράξεις, και ονομάζονται **τυπικές** παράμετροι.
Οι μεταβλητές του προγράμματος Παράδειγμα δεν είναι γνωστές στη διαδικασία

Πράξεις και αντίστοιχα όλες οι μεταβλητές της διαδικασίας Πράξεις είναι άγνωστες στο πρόγραμμα Παράδειγμα.
Τα ονόματα των τυπικών και των πραγματικών παραμέτρων μπορούν να είναι οποιαδήποτε. Αφού είναι ονόματα μεταβλητών σε διαφορετικά τμήματα προγράμματος, είναι υποχρεωτικά διαφορετικές μεταβλητές, άσχετα αν έχουν το ίδιο όνομα.

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

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.

Ας παρακολουθήσουμε πώς γίνεται η επικοινωνία ανάμεσα στο πρόγραμμα Παράδειγμα και τη διαδικασία Πράξεις.

Οι τιμές που υπάρχουν στις μεταβλητές του προγράμματος Α, Β, Διαφ1 και Αθρ1 δίνονται κατά την κλήση στις μεταβλητές της διαδικασίας Χ, Υ, Διαφορά, Άθροισμα.

Έτσι η μεταβλητή Χ παίρνει την τιμή 5 κι η Υ την τιμή 7. Οι μεταβλητές Διαφορά και Άθροισμα δεν παίρνουν καμία τιμή, αφού οι αντίστοιχες μεταβλητές Διαφ1 και Αθρ1 δεν έχουν συγκεκριμένη τιμή.

Μετά την εκτέλεση των εντολών της διαδικασίας, όταν εκτελεστεί η εντολή ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ, οι μεταβλητές της διαδικασίας που αναφέρονται στη δήλωση της διαδικασίας δίνουν τις τιμές που περιέχουν στις αντίστοιχες μεταβλητές που περιλαμβάνονται στην κλήση της διαδικασίας Πράξεις. Έτσι η Α παίρνει την τιμή της Χ (=5), η Β την τιμή της Υ (= 7), η Διαφ1 της Διαφορά (=-2) και η μεταβλητή Άθρ1 της Άθροισμα (=12).

Με την επιστροφή στο κύριο πρόγραμμα όλες οι θέσεις μνήμης που είχαν δοθεί στη διαδικασία απελευθερώνονται.

Οι λίστες των παραμέτρων πρέπει να ακολουθούν τους εξής κανόνες:

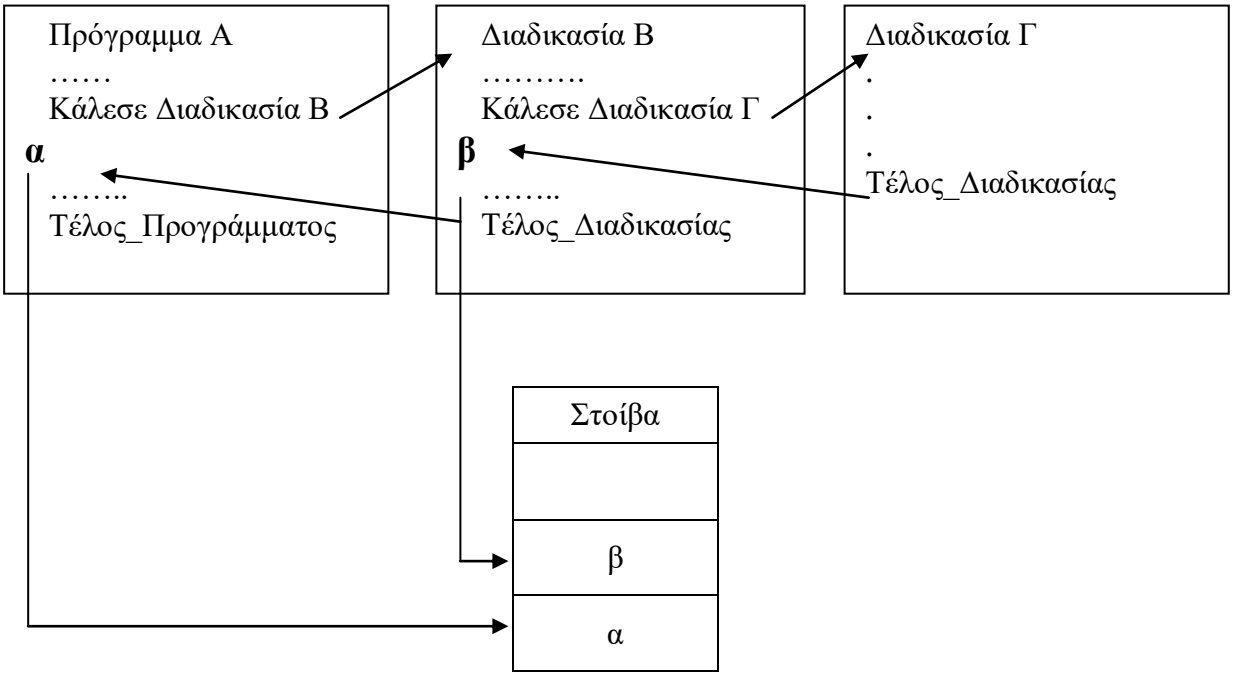
1. Ο αριθμός των πραγματικών και των τυπικών παραμέτρων πρέπει να είναι ίδιος.
2. Κάθε πραγματική παράμετρος αντιστοιχεί στην τυπική παράμετρο που βρίσκεται στην αντίστοιχη θέση. Για παράδειγμα η πρώτη της λίστας των τυπικών παραμέτρων στην πρώτη της λίστας των πραγματικών παραμέτρων κοκ.
3. Η τυπική παράμετρος και η αντίστοιχη της πραγματική πρέπει να είναι του ιδίου τύπου.

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

ζεται στοίβα χρόνου εκτέλεσης.

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

Για παράδειγμα, έστω ότι ένα πρόγραμμα Α καλεί τη διαδικασία Β, που με τη σειρά της καλεί τη διαδικασία Γ κοκ. Στην περίπτωση αυτή οι διευθύνσεις επιστροφής εμφανίζονται στη στοίβα
Το παράδειγμα αυτό δείχνει μια από τις πολλές χρησιμότητες της LIFO ιδιότητας της στοίβας.



Εμβέλεια μεταβλητών – σταθερών 10.6

Κάθε κύριο πρόγραμμα και κάθε υποπρόγραμμα έχει τις δικές του μεταβλητές και σταθερές. Οι μεταβλητές αυτές στη γλώσσα είναι γνωστές στο υποπρόγραμμα που δηλώνονται και μόνο σε αυτό.
Λέμε λοιπόν ότι όλες οι μεταβλητές (και οι σταθερές) είναι τοπικές. Ο μόνος τρόπος για να περάσει μια τιμή από ένα υποπρόγραμμα σε ένα άλλο η από το κυρίως πρόγραμμα σε ένα υποπρόγραμμα είναι δια μέσου παραμέτρων.

Το τμήμα του προγράμματος που ισχύουν οι μεταβλητές ονομάζεται εμβέλεια μεταβλητών.

Πολλές γλώσσες προγραμματισμού επιτρέπουν τη χρήση μεταβλητών και σταθερών, όχι μόνο στο τμήμα προγράμματος που δηλώνονται, αλλά και σε άλλα τμήματα η ακόμα και σε όλα τα υπόλοιπα υποπρογράμματα.

Έτσι λοιπόν μπορεί να έχουμε:

ΑΠΕΡΙΟΡΙΣΤΗ ΕΜΒΕΛΕΙΑ
Εδώ όλες οι μεταβλητές και 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, δημιουργεί πολλά προβλήματα ειδικά σε μεγάλα προγράμματα με πολλά υποπρογράμματα, αφού πρέπει ο καθένας να γνωρίζει τα ονόματα όλων των μεταβλητών που χρησιμοποιούνται στα υπόλοιπα υποπρογράμματα.

ΠΕΡΙΟΡΙΣΜΕΝΗ ΕΜΒΕΛΕΙΑ
Εδώ όλες οι μεταβλητές και σταθερές που χρησιμοποιούνται σε ένα τμήμα προγράμματος, πρέπει να δηλώνονται σε αυτό το τμήμα. Όλες οι μεταβλητές είναι τοπικές, ισχύουν δηλαδή μόνο στα υποπρογράμματα που δηλώθηκαν.

ΣΤΗ ΓΛΩΣΣΑ ΕΧΟΥΜΕ ΠΕΡΙΟΡΙΣΜΕΝΗ ΕΜΒΕΛΕΙΑ.

Τα πλεονεκτήματα στη περιορισμένη εμβέλεια είναι η απόλυτη αυτονομία των υποπρογραμμάτων και ότι έχουμε τη δυνατότητα να χρησιμοποιούμε οποιοδήποτε όνομα, χωρίς να ενδιαφέρει αν το ίδιο χρησιμοποιείται και σε άλλο υποπρόγραμμα.

ΜΕΡΙΚΩΣ ΠΕΡΙΟΡΙΣΜΕΝΗ ΕΜΒΕΛΕΙΑ
Εδώ άλλες μεταβλητές είναι τοπικές και άλλες καθολικές. Κάθε γλώσσα προγραμματισμού έχει τους δικούς της κανόνες και μηχανισμούς για τη διαχείριση των τοπικών και καθολικών μεταβλητών. Υπάρχουν πλεονεκτήματα για τους πεπειραμένους προγραμματιστές, αλλά οι αρχάριοι έχουν δυσκολίες.