Ανάπτυξη Εφαρμογών
σε Προγραμματιστικό Περιβάλλον

Σημειώσεις Θεωρίας

{The way to get started is to quit talking and begin doing.

Walt Disney }

Επιμέλεια Σταυρόπουλος Σταύρος st.stavropoulos@icloud.com

ΚΕΦΑΛΑΙΟ 2: ΒΑΣΙΚΈΣ ΈΝΝΟΙΕΣ ΑΛΓΟΡΊΘΜΩΝ

1. Αλγόριθμος

{Ορισμός} Είναι μία **πεπερασμένη** σειρά ενεργειών, αυστηρά **καθορισμένων** και **εκτελέσιμων** σε πεπερασμένο χρόνο, που **στοχεύουν** στην επίλυση ενός προβλήματος.

Η σειρά – αλληλουχία των ενεργειών δεν είναι μοναδική. Η έννοια του αλγορίθμου δεν συνδέεται αποκλειστικά με έννοιες της πληροφορικής.

2. Απαραίτητα κριτήρια αλγορίθμων

1. **Είσοδος**: **Καμία, μία ή περισσότερες** τιμές δεδομένων πρέπει να δίνονται ως είσοδος. Η περίπτωση που δεν δίνονται τιμές δεδομένων εμφανίζεται όταν: ο αλγόριθμος δημιουργεί και επεξεργάζεται κάποιες **πρωτογενείς** τιμές με την βοήθεια **συναρτήσεων παραγωγής τυχαίων αριθμών** ή με την βοήθεια άλλων **απλών εντολών**.
2. **Έξοδος**: Ο αλγόριθμος πρέπει να δημιουργεί **τουλάχιστον** ένα αποτέλεσμα προς τον **χρήστη** ή προς ένα άλλο **αλγόριθμο**.
3. **Καθοριστικότητα**: Κάθε εντολή πρέπει να καθορίζεται χωρίς **αμφιβολία** για τον τρόπο εκτέλεσής της. Πχ μία εντολή ζ <-- χ / ψ πρέπει να λαμβάνει υπ' όψιν της το γεγονός ότι μπορεί το ψ να είναι μηδέν.
4. **Περατότητα**: Ο αλγόριθμος πρέπει να τελειώνει μετά από **πεπερασμένα** βήματα. Αν δεν τελειώνει μετά από ένα συγκεκριμένο αριθμό βημάτων δεν είναι αλγόριθμος αλλά **υπολογιστική** διαδικασία.
5. **Αποτελεσματικότητα**: Κάθε εντολή ενός αλγορίθμου δεν αρκεί να είναι **ορισμένη** αλλά πρέπει να είναι **απλή** και **εκτελέσιμη**.

3. Περιγραφή και αναπαράσταση αλγορίθμων

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

4. Διάγραμμα ροής

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

Τα κυριότερα σχήματα είναι τα εξής:

1. **Έλλειψη**: Δηλώνει την αρχή και το τέλος του αλγορίθμου.
2. **Ρόμβος**: Δηλώνει τον έλεγχο μίας συνθήκης με δύο εξόδους ανάλογα με την τιμή της. {αληθής - ψευδής}
3. **Ορθογώνιο**: Δηλώνει την εκτέλεση μίας ή περισσοτέρων πράξεων.
4. **Πλάγιο παραλληλόγραμμο**: Η είσοδος δεδομένων και η έξοδος αποτελεσμάτων του αλγορίθμου.

5. Στοιχεία αλγορίθμου

Ένας αλγόριθμος διαμορφώνεται από τα παρακάτω :

- **Τελεσταίοι**
- **Τελεστές**
- **Εκφράσεις**
- **Εντολές**

6. Τελεσταίοι (operands)

Οι τελεσταίοι διακρίνονται σε:

Σταθερές (constants): **Προκαθορισμένες** τιμές που παραμένουν **αμετάβλητες** σε όλη τη διάρκεια της εκτέλεσης ενός αλγορίθμου. *{ προσοχή: Μια σταθερά μπορεί για παράδειγμα να είναι ένας αριθμός. Δηλαδή για το σχολικό βιβλίο ο αριθμος 3 είναι μια σταθερά γιατί δεν μπορεί να μεταβληθεί! Το 3 πάντα θα είναι 3 !! Μπορεί όμως να είναι και μια μονάδα μνήμης του υπολογιστή που έχει κλειδωθεί σε μια τιμή. Το "κλείδωμα" αυτό θα δεις πως γίνεται στο κεφάλαιο 7}*
Μεταβλητές (variables): Είναι ένα **γλωσσικό αντικείμενο** {δηλαδή λέξεις..} που χρησιμοποιούνται για να παραστήσουν στοιχεία δεδομένων. Η διαφορά τους από τις σταθερές είναι ότι δέχονται μία τιμή που μπορεί να αλλάξει κατά την εκτέλεση ενός αλγορίθμου. *{ Οι τελεσταίοι (σταθερές ή μεταβλητές) διακρίνονται σε τρεις κατηγορίες ανάλογα με το είδος των τιμών που μπορούν να λάβουν. Βλέπε για αντιπαραβολή το κεφάλαιο 7 όπου οι κατηγορίες-τύποι μεταβλητών είναι τέσσερις }*

7. Τύποι (ΚΑΤΗΓΟΡΙΕΣ) τελεσταίων

Αριθμητικοί: Είναι ακέραιες τιμές ή πραγματικές πχ 12, 3.14, -19.99 κλπ.

{ προσοχή: η υποδιαστολή στην πληροφορική ακολουθεί το αγγλικό σύστημα και συμβολίζεται με τελεία "." }

Αλφαριθμητικοί: Αποτελούνται από σειρές χαρακτήρων μέσα σε εισαγωγικά. Οι χαρακτήρες μπορεί να είναι γράμματα, ψηφία, σημεία στίξης κλπ πχ "γεια σας", "μεσος1", "1" κλπ

{εδώ το σχολικό μας δηλώνει εμμέσως πλην σαφώς οι αλφαριθμητικοί περικλείονται ανάμεσα σε διπλά εισαγωγικά... στο κεφάλαιο 7 όμως τους περικλείει σε μονά. Μην δίνεις σημασία. Έχεις τόσα άλλα ωραία πράγματα να μάθεις... μην χάνεις τον χρόνο σου να ασχολείσαι με τέτοιες λεπτομέρειες. }

Λογικοί: Η τιμές που μπορούν να πάρουν είναι ακριβώς δύο: αληθής, ψευδής

8. Τελεστές (operators)

Είναι σύμβολα που χρησιμοποιούνται στις διάφορες πράξεις. Διακρίνονται σε τρεις κατηγορίες:

Αριθμητικοί:	**Λογικοί:**	**Συγκριτικοί:**
^	Oxi (άρνηση)	=
		>
* / div mod	Και (σύζευξη)	<
		>=
+ -	Η (διάζευξη)	<=
		<>

9. Εκφράσεις (expressions)

Όταν μία τιμή προκύπτει από υπολογισμό τότε αναφερόμαστε σε εκφράσεις. Οι τελεστές μαζί με τους τελεσταίους διαμορφώνουν τις εκφράσεις. Η διεργασία αποτίμησης μίας έκφρασης συνίσταται στην απόδοση τιμών στις μεταβλητές και στην εκτέλεση των πράξεων. Η εκτέλεση των πράξεων εξαρτάται από την ιεραρχία των πράξεων και την χρήση παρενθέσεων. Μία έκφραση μπορεί να αποτελείται από μία μόνο μεταβλητή ή σταθερά μέχρι μία πολύπλοκη μαθηματική παράσταση. *{ Χωρίζονται σε δύο κατηγορίες : τις αριθμητικές (Σχολ. Κεφ. 7) και τις λογικές (Σχολ. Κεφ. 8) Οι λογικές εκφράσεις ονομάζονται αλλιώς και: Συνθήκες }*

10. Εντολές

Αποκαλείται κάθε μία λέξη που προσδιορίζει μία σαφή ενέργεια.Οι εντολές μπορούν να διακριθούν στις τρεις βασικές δομές του δομημένου προγραμματισμού: *{βλέπε και κεφάλαιο 6ο (δομημένος προγραμματισμός)}*

1. Δομή ακολουθίας
2. Δομή επιλογής
3. Δομή επανάληψης

Οι εντολές διακρίνονται επίσης σε δηλωτικές και σε εκτελέσιμες.

1. Δηλωτικές: αλγοριθμος
2. Εκτελέσιμες: διαβασε χ

11. Δομή ακολουθίας

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

Στη δομή αυτή ανήκουν οι εντολές:

- **Εισόδου**: Διάβασε. Η εντολή αυτή συνοδεύεται με το όνομα μίας ή περισσοτέρων μεταβλητών Η λειτουργία της είναι: μετά την ολοκλήρωσή της η μεταβλητή/μεταβλητές θα έχει λάβει τιμή ως περιεχόμενο. Πχ : διάβασε χ,ψ
- **Εξόδου**: Εμφάνισε – Εκτύπωσε. Οι εντολές αυτές εμφανίζουν τα αποτελέσματα στην οθόνη και στον εκτυπωτή αντίστοιχα. Η σύνταξή της εντολής αυτής είναι ανάλογη με του διάβασε.
- **Εκχώρησης τιμής**: ← Η γενική μορφή της εντολής είναι : Μεταβλητή←έκφραση. Η λειτουργία της είναι: γίνονται οι πράξεις στην έκφραση και το αποτέλεσμά της αποδίδεται , μεταβιβάζεται, εκχωρείται στη μεταβλητή. Ας σημειωθεί ότι δεν πρόκειται για εξίσωση και ότι οι διάφορες γλώσσες προγραμματισμού χρησιμοποιούν διάφορα σύμβολα αντί για το βέλος.
- **Δηλωτικές** : Αλγόριθμος , Τέλος. Ένας αλγόριθμος διατυπωμένος σε ψευδογλώσσα αρχίζει πάντα με τη λέξη Αλγόριθμος συνοδευόμενη με το όνομα του και τελειώνει με την λέξη Τέλος συνοδευόμενη επίσης με το όνομά του.

{Για περισσότερες λεπτομέρειες όσον αφορά τις τρεις πρώτες εντολές ανατρέξτε καλύτερα στο κεφάλαιο 7}

12. Δομή επιλογής

Αποτελείται από ένα σύνολο εντολών που εκτελούνται κατά περίπτωση. Η διαδικασία της επιλογής περιλαμβάνει τον έλεγχο κάποιας συνθήκης με δύο δυνατές τιμές (αληθής, ψευδής) και στη συνέχεια την απόφαση εκτέλεσης κάποιας εντολής ανάλογα με τη συνθήκη. *{Για περισσότερα βλέπε κεφάλαιο 8}*

13. Δομή επανάληψης

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

{βλέπε κεφάλαιο 8}

14. Ολίσθηση

Όταν ο προγραμματιστής ορίζει ένα δεδομένο τότε αυτό αποθηκεύεται στα κυκλώματα του υπολογιστή σε δυαδική μορφή δηλαδή σε ακολουθίες από 0 και 1.

Αν για παράδειγμα ο προγραμματιστής εκτελέσει την εντολή : χ←17 τότε μέσα στην μνήμη του υπολογιστή (η οποία ουσιαστικά είναι ένα κύκλωμα) αποθηκεύεται ο ισοδύναμος αριθμός του δυαδικού συστήματος, ο οποίος είναι το 00010001. Ολίσθηση είναι η μετακίνηση όλων των ψηφίων ενός αριθμού κατά μια θέση.

Διακρίνουμε δύο ολισθήσεις:

Ολίσθηση προς τα αριστερά.
Ισοδυναμεί με πολλαπλασιασμό επί δύο. Μετακινούμε όλα τα ψηφία του προς τα αριστερά προσθέτοντας ένα μηδέν στο τέλος και αγνοώντας το αρχικό μηδέν.
Ολίσθηση προς τα δεξιά.
Ισοδυναμεί με την ακέραια διαίρεση δια δύο. Μετακινούμε όλα τα ψηφία του προς τα δεξιά, αποκόπτοντας το τελευταίο και προσθέτοντας ένα μηδενικό στην αρχή.

15. Πολλαπλασιασμός αλά ρωσικά

Τι είναι;

Η πράξη του πολλαπλασιασμού δύο αριθμών δεν εκτελείται από τον υπολογιστή με τον τρόπο που την εκτελούμε εμείς. Εκτελείται με έναν άλλο τρόπο που λέγεται πολλαπλασιασμός αλά ρωσικά.

Περιγραφή της μεθόδου πολλαπλασιασμού (για θετικούς ακέραιους αριθμούς):

Έστω ότι θέλω να πολλαπλασιάσω τους αριθμούς 45 και 19. Οι αριθμοί που πρέπει να πολλαπλασιαστούν γράφονται δίπλα δίπλα και ο πρώτος διπλασιάζεται ενώ ο δεύτερος υπερδιπλασιάζεται (για την ακρίβεια τον υποδιπλασιάζουμε αγνοώντας το δεκαδικό μέρος). Η διαδικασία αυτή επαναλαμβάνεται ώσπου ο δεύτερος να γίνει μηδέν :

Παράδειγμα:

45	19	45
90	9	90
180	4	
360	2	
720	1	720
	0	

Τελικά το ζητούμενο γινόμενο ισούται με το άθροισμα των στοιχείων της πρώτης στήλης όπου αντίστοιχα στην δεύτερη στήλη υπάρχει περιττός αριθμός. Στο παραπάνω παράδειγμα τα στοιχεία αυτά παρουσιάζονται στην τρίτη στήλη. Δηλαδή είναι 45+90+720=855 !

Αντίστοιχος αλγόριθμος:

```
Αλγόριθμος πολλαπλασιασμός_αλα_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

    τελος_επαναληψης

    Αποτελέσματα // Ρ, το γινόμενο των ακεραίων Μ1,Μ2 //

τελος πολλαπλασιασμός_αλα_ρωσικά
```

{Παρατήρησε την χρήση των εντολών δεδομένα και αποτελέσματα. Η επεξήγησή τους δίνεται παρακάτω. Παρατήρησε επίσης την εντολή Ρ←Ρ+1. Δεν είναι τυπογραφικό λάθος !!! }

Μετατροπή:

Ο παραπάνω αλγόριθμος μπορεί εύκολα να μετατραπεί ώστε να περιλάβει και την περίπτωση πολλαπλασιασμού αρνητικών ακεραίων. *{ωραία άσκηση για τεστάκι... χιχι}*

16. Γιατί ο υπολογιστής εκτελεί τον πολλαπλασιασμό αλά ρωσικά;

- Γιατί υλοποιείται πιο απλά και λιγότερο χρονοβόρα σε σχέση με τον χειρωνακτικό τρόπο πολλαπλασιασμού.
- Απαιτεί μόνο πολλαπλασιαμό επί δύο, διαίρεση δια δύο και πρόσθεση σε αντίθεση με την γνωστή μας διαδικασία πολ/μου που απιτεί πολλαπλασιασμό με οποιοδήποτε ακέραιο και πρόσθεση.
- Σε επίπεδο κυκλωμάτων ο διπλασιασμός υλοποιείται ταχύτατα με εντολή ολίσθησης προς τα αριστερά ενώ ο υποδιπλασιασμός με ολίσθηση προς τα δεξιά , σε αντίθεση με τον πολλαπλασιασμό με οποιονδήποτε ακέραιο που θεωρείται χρονοβόρα διαδικασία.
- Το τελευταίο γεγονός είναι ο λόγος που ο πολλαπλασιασμός αλά ρωσικά είναι προτιμότερος απ' ότι ο χειρωνακτικός τρόπος πολλαπλασιασμού δύο ακεραίων.

17. Δομή ενός αλγόριθμου σε ψευδογλώσσα

Αλγόριθμος όνομα

Δεδομένα // ... //

Εντολές

Αποτελέσματα // ... //

Τέλος όνομα

{ ΣΧΟΛΙΑ: Στα δεδομένα αναγράφουμε ανάμεσα στις κάθετες γραμμές όλες τις μεταβλητές ή/και πίνακες που εμφανίζονται σε εντολές εισόδου, ενώ στα αποτελέσματα όλες τις μεταβλητές ή/και πίνακες που εμφανίζονται σε εντολές εξόδου.
Προσοχή λίγο στο γεγονός ότι όταν δηλώνουμε πίνακες στα δεδομένα ή στα αποτελέσματα ΔΕΝ τοποθετούμε αγκύλες στους πίνακες άρα δεν δηλώνουμε το μέγεθός τους !! Εδώ ίσως αναρωτηθείς ... μα που το ξέρει ο υπολογιστής το μέγεθος του πίνακα; Απάντηση: ο αλγόριθμος δεν είναι πρόγραμμα αλλά μια έκφραση ενός αλγορίθμου σε ψευδογλώσσα άρα απευθύνεται σε έναν άνθρωπο και όχι στην μηχανή!
Πχ Να γραφεί αλγόριθμος σε ψευδογλώσσα που υπολογίζει και εμφανίζει το εμβαδά ενός τριγώνου.

```
Αλγόριθμος Τρίγωνο
        Δεδομένα  // βάση,ύψος //
        Εμβαδό<--βάση*ύψος/2
        Αποτελέσματα          // Εμβαδό //
Τέλος όνομα
```

{ Put your heart, mind, and soul into even your smallest acts. This is the secret of success.

Swami Sivananda }

ΚΕΦΑΛΑΙΟ 3: ΔΟΜΈΣ ΔΕΔΟΜΈΝΩΝ

18. Δομή δεδομένων

{Ορισμός}. Είναι ένα σύνολο αποθηκευμένων δεδομένων που υφίστανται επεξεργασία από ένα σύνολο λειτουργιών.

Τα δεδομένα ενός προβλήματος αποθηκεύονται στον υπολογιστή, είτε στην κύρια μνήμη του είτε στην δευτερεύουσα. Η αποθήκευση δεν γίνεται κατά ένα τυχαίο τρόπο αλλά συστηματικά, δηλαδή χρησιμοποιώντας μία δομή. Κάθε μορφή δομής δεδομένων αποτελείται από ένα σύνολο κόμβων (nodes).

{δηλαδή: ένας πίνακας 100 θέσεων έχει ακριβώς 100 κόμβους}

19. Βασικές λειτουργίες (πράξεις) των δομών δεδομένων

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

20. Γιατί υπάρχουν διαφορετικές δομές;

Στην πράξη σπάνια χρησιμοποιούνται και οι οκτώ λειτουργίες για κάποια δομή. Συνήθως μία δομή είναι αποδοτικότερη από μία άλλη σε κάποια λειτουργία. Άλλη είναι πιο αποδοτική στην αναζήτηση άλλη στην εισαγωγή κλπ. Ανάλογα με την λειτουργία την οποία θέλουμε να κάνουμε επιλέγουμε και την κατάλληλη δομή. Γι' αυτό υπάρχουν τόσες πολλές δομές. Τα παραπάνω εξηγούν το πόσο σπουδαία είναι η επιλογή της κατάλληλης δομής.

21. Εξάρτηση δομής δεδομένων και αλγορίθμου

Υπάρχει μεγάλη εξάρτηση μεταξύ της δομής δεδομένων και του αλγόριθμου που επεξεργάζεται τη δομή. Μάλιστα, το πρόγραμμα πρέπει να θεωρεί τη δομή δεδομένων και τον αλγόριθμο ως μία αδιάσπαστη ενότητα. Η παρατήρηση αυτή δικαιολογεί την εξίσωση που διατυπώθηκε το 1976 από τον Wirth (που σχεδίασε και υλοποίησε τη γλώσσα Pascal):

Αλγόριθμοι + Δομές Δεδομένων = Προγράμματα

{Αν για παράδειγμα θέλεις να κάνεις έναν αλγόριθμο που κάνει αναζήτηση σε έναν μονοδιάστατο πίνακα, το πρόγραμμα που θα προκύψει θα είναι εντελώς διαφορετικό από το

να γράψεις έναν αλγόριθμο που κάνει αναζήτηση σε έναν δισδιάστατο πίνακα. Συμπέρασμα: το πρόγραμμα που γράφουμε είναι το αποτέλεσμα του αλγορίθμου που έχουμε στο κεφάλι μας και της δομής δεδομένων που χρησιμοποιούμε}

22. Κατηγορίες δομών δεδομένων

Οι δομές διακρίνονται σε δύο μεγάλες κατηγορίες :

- Δυναμικές και
- στατικές

23. Στατικές δομές

- Το ακριβές **μέγεθός** τους είναι σταθερό και καθορίζεται κατά την στιγμή του προγραμματισμού τους και όχι κατά την στιγμή της εκτέλεσης του προγράμματος.
- Οι κόμβοι τους αποθηκεύονται σε **συνεχόμενες** θέσεις μνήμης.
- Στην πράξη οι στατικές δομές υλοποιούνται με **πίνακες** .

24. Δυναμικές δομές

- Οι δομές αυτές δεν έχουν σταθερό **μέγεθος** αλλά ο αριθμός των κόμβων τους μεγαλώνει καθώς εισάγονται νέα δεδομένα, και μικραίνει καθώς διαγράφονται δεδομένα από αυτές.
- Δεν αποθηκεύονται σε **συνεχόμενες** θέσεις μνήμης αλλά στηρίζονται στην τεχνική της δυναμικής παραχώρησης μνήμης (DMA : Dynamic Memory Allocation).
- Όλες οι **σύγχρονες** γλώσσες προγραμματισμού τις υποστηρίζουν.
- Δυναμικές δομές είναι το **αρχείο**, η **εγγραφή** κ.α.
  {Σημείωση Οι πρώτες δύο παράγραφοι περιέχονται στην σελίδα 72 του νέου σχολικού βιβλίου ενώ οι υπόλοιπες τεσσερις στην σελίδα 57.}

25. Πίνακες

Είναι μία δομή που περιέχει στοιχεία του ίδιου τύπου (δηλαδή ακέραιους , πραγματικούς).

Η αναφορά στα στοιχεία του γίνεται με την χρήση του συμβολικού ονόματος του πίνακα ακολουθούμενου από την τιμή ενός ή περισσοτέρων δεικτών. ΠΧ Α[3,4]

Ο τρόπος δήλωσης του πίνακα και η μέθοδος αναφοράς του εξαρτάται από την γλώσσα προγραμματισμού που χρησιμοποιείται.

{ΠΧ στην γλώσσα Java ένας πίνακας Α 3 γραμμών και 4 στηλών δηλώνεται ως: A[3] [4] }

Ένας πίνακας μπορεί να είναι μονοδιάστατος , δισδιάστατος , τρισδιάστατος και γενικά ν-διάστατος. Ειδικά για τους δισδιάστατους πίνακες αν το μέγεθος των δύο διαστάσεων είναι

ίδιο τότε ο πίνακας ονομάζεται τετραγωνικός. Οι πίνακες χρησιμεύουν για την αποθήκευση και διαχείριση δύο σπουδαίων δομών της στοίβας και της ουράς

26. Αναζήτηση

Το πρόβλημα της αναζήτησης έχει ιδιαίτερο ενδιαφέρον λόγω της χρησιμότητάς του σε πλήθος εφαρμογών. Υπάρχουν αρκετοί αλγόριθμοι (μέθοδοι) αναζήτησης σε πίνακα. Η επιλογή του πιο κατάλληλου εξαρτάται κυρίως από το

- Αν ο πίνακας είναι ταξινομημένος ή όχι.
- Αν όλα τα στοιχεία του πίνακα είναι διαφορετικά μεταξύ τους ή όχι.

Τα στοιχεία του πίνακα στον οποίο γίνεται αναζήτηση μπορεί να είναι οποιουδήποτε τύπου: αριθμητικά (ακέραιοι πραγματικοί), αλφαριθμητικά (χαρακτήρες) ακόμη και λογικά.

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

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

Η χρήση της δικαιολογείται σε περιπτώσεις όπου ο πίνακας:

- είναι μη ταξινομημένος,
- είναι μικρού μεγέθους (ν<=20),
- η αναζήτηση γίνεται σπάνια.

Ο αλγόριθμος της γραμμικής αναζήτησης είναι:

```
Αλγόριθμος σειριακή_αναζήτηση
        Δεδομένα //n, table, key//
        done←ψευδής
        i←1
        position←0
        Όσο i<=n και done=ψευδής επανάλαβε
                αν table[i]=key τότε
                        done← αληθής
                        position←i
                αλλιώς
                        i←i+1
                τελος_αν
        τελος_επανάληψης
        αποτελέσματα //position , done //
τελος σειριακή_αναζήτηση
```

Μια αποτελεσματικότερη μέθοδος αναζήτησης σε ταξινομημένο πίνακα είναι η δυαδική αναζήτηση.

{χμχμ λίγο προσοχή εδώ: να μελετήσεις το κεφάλαιο 9 όπου αναφέρει ξανά την έννοια της σειριακής αναζήτησης}

28. Ταξινόμηση

{Ορισμός}: Δοθέντων των στοιχείων α1, α2, ... αn η ταξινόμηση ορίζεται ως μετάθεση της θέσης των στοιχείων ώστε να τοποθετηθούν σε μια σειρά ακ1, ακ2, ... ακn έτσι ώστε: Αν δοθεί συνάρτηση διάταξης (ordering function) F να ισχύει : F(ακ1) ≤ F(ακ2) ≤ ... ≤ F(ακn). Η παραπάνω συνάρτηση διάταξης μπορεί να τροποποιηθεί ώστε να καλύπτει και την φθίνουσα ταξινόμηση.

{Αν η συνάρτηση διάταξης είναι για παράδειγμα η F(x)= x (η οποία αν θυμάσαι/γνωρίζεις είναι γνησίως αύξουσα) τότε η παραπάνω διάταξη είναι αύξουσα. Αν η συνάρτηση διάταξης είναι η F(x)=-x (γνησίως φθίνουσα) τότε η παραπάνω διάταξη είναι φθίνουσα.}

Ταξινόμηση είναι η διαδικασία κατά την οποία τα στοιχεία μιας δομής διατάσσονται κατά φθίνουσα ή αύξουσα σειρά.

Σκοπός της ταξινόμησης είναι η **εύκολη αναζήτηση** ενός στοιχείου.

29. Ταξινόμηση ευθείας ανταλλαγής-φυσαλίδας

Η μέθοδος της φυσαλίδας βασίζεται στην σύγκριση και ανταλλαγή ζευγών γειτονικών στοιχείων μέχρις ότου διαταχθούν όλα τα στοιχεία.

Αν θεωρήσουμε ένα πίνακα μονοδιάστατο σε κατακόρυφη θέση, σύμφωνα με την μέθοδο αυτή γίνονται διαδοχικές **προσπελάσεις** στον πίνακα και μετακινείται το μικρότερο στοιχείο του πίνακα προς τα άνω (αύξουσα ταξινόμηση).

Ο αντίστοιχος αλγόριθμος είναι:

```
Αλγόριθμος φυσαλίδα
      Δεδομένα // table,n //
      για i από 2 μέχρι n
          για j από n μέχρι i με_βήμα -1
              αν table[j]>table [j-1] τότε
                 αντιμετάθεσε table[j], table [j-1]
              τελος_αν
          τελος_επανάληψης
      τελος_επανάληψης
      αποτελέσματα // table //
τελος φυσαλίδα
```

Η εντολή αντιμετάθεσε ανταλλάσσει το περιεχόμενο δυο θέσεων μνήμης με την βοήθεια μιας τρίτης θέσης. Αυτό μπορεί να γίνει με τις εξής τρεις εντολές:

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

table [j-1] ← tmp

{χμχμ λίγο προσοχή εδώ: να μελετήσεις το κεφάλαιο 9 όπου αναφέρει ξανά την έννοια της φυσσαλίδας }

30. Άλλοι αλγόριθμοι ταξινόμησης

Υπάρχουν πολλοί αλγόριθμοι ταξινόμησης.

Αλλοι σχετικά απλοί αλγόριθμοι είναι

- η ταξινόμηση με επιλογή και
- η ταξινόμηση με παρεμβολή.
- η γρήγορη ταξινόμηση (ο ποιό γρήγορος αλγόριθμος ταξινόμησης).

Η ταξινόμηση φυσσαλίδας είναι ο πιο απλός και ταυτόχρονα ο πιο αργός αλγόριθμος ταξινόμησης.

31. Δομές Δεδομένων δευτερεύουσας μνήμης

Σε μεγάλες εφαρμογές το μέγεθος της κύριας μνήμης (RAM) δεν είναι αρκετό για την αποθήκευση όλων των δεδομένων. Στην περίπτωση αυτή χρησιμοποιούνται ειδικές δομές που αποθηκεύουν τα δεδομένα στην **δευτερεύουσα** μνήμη που είναι συνήθως ο **σκληρός δίσκος**. Οι ειδικές αυτές δομές ονομάζονται **αρχεία**.

Μια σημαντική διαφορά μεταξύ της κύριας μνήμης και της δευτερεύουσας είναι ότι τα δεδομένα δεν χάνονται όταν τερματίσει το πρόγραμμα ή ακόμα και όταν διακοπεί η ηλεκτρική παροχή. Έτσι τα δεδομένα των αρχείων αποθηκεύονται μόνιμα σε σχέση με τις δομές της κύριας μνήμης (όπως οι πίνακες) που αποθηκεύονται προσωρινά.

32. Τι είναι τα αρχεία και από τι αποτελούνται

Τα **αρχεία** είναι μια ειδική δομή δεδομένων της δευτερεύουσας μνήμης (συνήθως ο σκληρός δίσκος) και αποτελούνται από **εγγραφές**. Κάθε μια εγγραφή αποτελείται από ένα η περισσότερα **πεδία** που περιγράφουν διάφορα χαρακτηριστικά της εγγραφής. Ένα πεδίο της εγγραφής που ταυτοποιεί την εγγραφή ονομάζεται **πρωτεύον** κλειδί ή απλά κλειδί. Πολλά αρχεία έχουν και δεύτερο πεδίο που ταυτοποιεί μια εγγραφή και ονομάζεται **δευτερεύων** κλειδί

{ Perfection is not attainable, but if we chase perfection we can catch excellence.

Vince Lombardi }

ΚΕΦΑΛΑΙΟ 6: ΕΙΣΑΓΩΓΉ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΌ

33. Από τι προσδιορίζεται μια γλώσσα

Οι φυσικές γλώσσες αλλά και οι γλώσσες προγραμματισμού προσδιορίζενται από **4** στοιχεία:

1 Το αλφάβητο.

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

2 Το λεξιλόγιο.

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

3 Την γραμματική.

Η γραμματική αποτελείται από το τυπικό και το συντακτικό.

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

Συντακτικό: Είναι το σύνολο των κανόνων που καθορίζει αν η **διάταξη** και η **σύνδεση** των λέξεων σε μία πρόταση είναι σωστή. Η γνώση συντακτικού στις φυσικές γλώσσες επιτρέπει την δημιουργία σωστών προτάσεων ενώ στις γλώσσες προγραμματισμού επιτρέπει την δημιουργία σωστών εντολών

{*ΠΧ:}*

Φυσική γλωσσα:	*Γλωσσα Προγραμματισμού:*
Σωστό: Πάω να φάω	*Σωστό: Χ←3*
Λάθος: Φάω να πάω	*Λάθος: 3←Χ*

4 Την σημασιολογία.

Είναι το σύνολο των κανόνων που καθορίζει το νόημα των **λέξεων** άρα και το νόημα των **προτάσεων** που δημιουργούνται. *{Πχ Απογειώθηκε το ντροπαλό κατσαβίδι. (δεν έχει νόημα..)}*

Στις γλώσσες προγραμματισμού ο δημιουργός τους αποφασίζει για την σημασιολογία των λέξεων της γλώσσας.

34. Διαφορές φυσικών – τεχνητών γλωσσών

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

- Οι γλώσσες προγραμματισμού αλλάζουν σε:
  - επίπεδο **διαλέκτου**. Πχ Basic ␣ QuickBasic
  - σε επίπεδο **επέκτασης** Πχ Basic ␣ Visual Basic

35. Τεχνική της ιεραρχικής σχεδίασης προγράμματος.

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

Χρησιμοποιούνται διάφορες διαγραμματικές τεχνικές για την υποβοήθηση της σχεδίασης.Υλοποιείται με **τμηματικό** προγραμματισμό.

Τι περιλαμβάνει η τεχνική της ιεραρχικής σχεδίασης;

- Τον **καθορισμό** των **βασικών** λειτουργιών ενός προγράμματος σε **ανώτερο** επίπεδο,
- την **διάσπαση** των λειτουργιών αυτών σε όλο και **μικρότερες** λειτουργίες,
- μέχρι το **τελευταίο** επίπεδο όπου οι λειτουργίες είναι πολύ **απλές**.

36. Τμηματικός προγραμματισμός

{ Μελέτησέ τον από το κεφάλαιο 10. }

37. Δομημένος προγραμματισμός

Δεν είναι απλώς ένα είδος προγραμματισμού αλλά μια **μεθοδολογία σύνταξης** προγραμμάτων. **Προήλθε** από μια προσπάθεια περιορισμού της ανεξέλεγκτης **χρήσης** της εντολής **goto**. Όλες οι **σύγχρονες** γλώσσες προγραμματισμού υποστηρίζουν τον δομημένο προγραμματισμό και διαθέτουν εντολές που καθιστούν την χρήση του goto **περιττή**. Ο

δομημένος προγραμματισμός βοηθάει την **ανάλυση** ενός προγράμματος σε **τμήματα** έτσι **περιέχει** τόσο την **ιεραρχική** σχεδίαση όσο και τον **τμηματικό προγραμματισμό**.

38. Ποιές δομές χρησιμοποιεί ο δομημένος προγραμματισμός

Σύμφωνα με τον δομημένο προγραμματισμό, όλα τα προγράμματα μπορούν να γραφούν χρησιμοποιώντας **μόνο** τις τρεις παρακάτω λογικές δομές καθώς και συνδυασμών τους.

1. Δομή ακολουθίας.
2. Δομή επιλογής.
3. Δομή επανάληψης.

Κάθε πρόγραμμα όπως και κάθε ενότητα προγράμματος έχει μία είσοδο και μόνο μία έξοδο

{Εδώ εννοεί ότι έχει μόνο μία αρχή και μόνο ένα τέλος_προγράμματος !!!}

39. Γιατί η εντολή GOTO κρίνεται ακατάλληλη

Επειδή η χρήση της εντολής **αλλάζει** την **ροή** ενός προγράμματος, τα προγράμματα γίνονται **δύσκολα** στην παρακολούθηση την κατανόηση και την συντήρηση.

40. Πλεονεκτήματα δομημένου προγραμματισμού

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

41. Πηγαίο ή αρχικό πρόγραμμα (source)

- Είναι το **αρχικό** πρόγραμμα που γράφει ο προγραμματιστής με την βοήθεια ενός προγράμματος που ονομάζεται **συντάκτης.**
- Δεν είναι **κατανοητό** από τον υπολογιστή. (εκτός αν είναι γραμμένο απ’ ευθείας σε γλώσσα μηχανής)

42. Λάθη στο πηγαίο πρόγραμμα

- **Λογικά** λάθη.
    - Οφείλονται σε σφάλματα κατά την υλοποίηση του αλγορίθμου.
    - Εντοπίζονται παρά μόνο κατά την εκτέλεση του προγράμματος.
    - Είναι τα πλέον σοβαρά και δύσκολα στην διόρθωση.
- Συντακτικά λάθη.
    - Οφείλονται σε αναγραμματισμούς γραμμάτων εντολών, παράληψη δήλωσης δεδομένων κλπ.

- Εντοπίζονται από τον μεταγλωττιστή ή τον διερμηνευτή.
- Πρέπει να διορθωθούν ώστε να δημιουργηθεί το εκτελέσιμο πρόγραμμα.

{πχ Γραεψ αντί για Γράψε}

43. Τι είναι ο συντάκτης

Είναι: ένας μικρός επεξεργαστής κειμένου που χρησιμοποιείται για την **συγγραφή** του πηγαίου προγράμματος, την **διόρθωση** των λαθών του και έχει δυνατότητες που διευκολύνουν την **γρήγορη γραφή** των εντολών.

44. Μετατροπή προγραμμάτων σε γλώσσα μηχανής

Γίνεται με την χρήση ειδικών μεταφραστικών προγραμμάτων. Υπάρχουν δύο μεγάλες κατηγορίες μεταφραστικών προγραμμάτων: Οι μεταγλωττιστές και οι διερμηνευτές.

45. Μεταγλωττιστής

- Δέχεται σαν **είσοδο** ένα πρόγραμμα γραμμένο σε μία γλώσσα προγραμματισμού (υψηλού επιπέδου).
- Ανιχνεύει τα τυχόν συντακτικά λάθη.
- Αν βρεθούν λάθη ο προγραμματιστής τα διορθώνει (με τον συντάκτη) και υποβάλλει το πρόγραμμα **ξανά** προς μεταγλώττιση μέχρι να παραχθεί το σωστό.
- Αν δεν υπάρχουν λάθη και μόνο τότε, παράγει το **αντικείμενο** πρόγραμμα (ή **τελικό** πρόγραμμα), το οποίο είναι ισοδύναμο με το πηγαίο αλλά εκφρασμένο πλέον σε γλώσσα μηχανής. Αυτό είναι πλέον τελείως **ανεξάρτητο** από το αρχικό πρόγραμμα, αλλά **δεν** είναι ακόμη εκτελέσιμο.
- Η διαδικασία μέσω της οποίας καταλήγουμε στο εκτελέσιμο πρόγραμμα (γλώσσα μηχανής) είναι συνοπτικά:
  Πηγαίο→μεταγλωττιστής→αντικείμενο(τελικό)→συνδέτης-φορτωτής→Εκτελέσιμο.

46. Συνδέτης φορτωτής

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

47. Διερμηνευτής

- Δέχεται ως **είσοδο** ένα πρόγραμμα γραμμένο σε γλώσσα υψηλού επιπέδου και **εκτελεί μία μία** τις εντολές του αρχικού προγράμματος.
- Η διαδικασία εκτέλεσης είναι η εξής:
  - Διαβάζει μία από τις εντολές του αρχικού προγράμματος,
  - **ανιχνεύει** τα (τυχόν) **συντακτικά** λάθη της κάθε εντολής,
  - την μεταφράζει σε γλώσσα μηχανής

    - την εκτελεί.
- Κατόπιν διαβάζει την επόμενη εντολή και **επαναλαμβάνει** την ίδια διαδικασία !

48. Ομοιότητες διερμηνευτή-μεταγλωττιστή

{ η παράγραφος αυτή δεν αναφέρεται ξεκάθαρα στο σχολικό εγχειρίδιο} Και οι δύο **μεταφράζουν** το πηγαίο πρόγραμμα (υψηλού επιπέδου) σε γλώσσα μηχανής. Και οι δύο **ανιχνεύουν** τα **συντακτικά** λάθη.

49. Διαφορές διερμηνευτή-μεταγλωττιστή

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

Διαφορές ως απόρροια των παραπάνω είναι:

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

50. (Σύγχρονα) προγραμματιστικά περιβάλλοντα

Για την δημιουργία - εκτέλεση ενός προγράμματος απαιτούνται τουλάχιστον 3 προγράμματα:

- Συντάκτης
- Μεταγλωττιστής
- Συνδέτης

Τα σύγχρονα προγραμματιστικά περιβάλλοντα παρέχουν τα προγράμματα αυτά με **ενιαίο** τρόπο. Το κάθε προγραμματιστικό περιβάλλον έχει διαφορετικά **εργαλεία** και ιδιότητες (ανάλογα με την γλώσσα προγραμματισμού). Για παράδειγμα ένα περιβάλλον **οπτικού** προγραμματισμού παρέχει και **ειδικό** συντάκτη για την δημιουργία **γραφικών** (μενού διαλόγου κλπ)

{ Not trying is a habit that develops into a fear of failure. If you want to succeed, try !

Michael Jordan }

ΚΕΦΑΛΑΙΟ7: ΒΑΣΙΚΆ ΣΤΟΙΧΕΊΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΎ

*{Στο κεφάλαιο 7ο και 8ο το βιβλίο περιγράφει την γλώσσα προγραμματισμού **ΓΛΩΣΣΑ**. Η γλώσσα αυτή είναι ένα υβρίδιο της pascal και της basic μεταφρασμένο στα ελληνικά}*

51. Από τι αποτελείται το αλφάβητο της ΓΛΩΣΣΑΣ

Αποτελείται από:

- Γράμματα
- Κεφαλαία και μικρά ελληνικού αλφαβήτου (Α-Ω , α-ω)
- Κεφαλαία και μικρά αγγλικού αλφαβήτου (A-Z, a-z)
- Ψηφία 0-9
- Ειδικοί χαρακτήρες

+ - * / ^ = > < , . ! & [ ] ( ) _

Και ο κενός χαρακτήρας

{Μια υπενθύμιση: Άλλο ο κενός χαρακτήρας (το space) και άλλο η κάτω παύλα (το underscore)}

52. Ποιοί είναι οι τύποι δεδομένων της ΓΛΩΣΣΑΣ

- **Ακέραιος**. Ο τύπος αυτός περιλαμβάνει τους ακέραιους (το σύνολο Z) που είναι γνωστοί από τα μαθηματικά. (δηλαδή **θετικούς** και **αρνητικούς** καθώς και το **0**)
- **Πραγματικός**. Ο τύπος αυτός περιλαμβάνει τους πραγματικούς (το σύνολο R) που είναι γνωστοί από τα μαθηματικά. (δηλαδή αριθμούς με **δεκαδικό** μέρος **θετικούς** και **αρνητικούς** καθώς και το **0**)

*{Πρακτικά: Η διαφορά μεταξύ τους είναι μια μεταβλητή πραγματικού τύπου μπορεί να πάρει **και** δεκαδικές τιμές }*

- **Λογικός**. Ο τύπος αυτός δέχεται μόνο δύο τιμές **ΑΛΗΘΗΣ** και **ΨΕΥΔΗΣ**.
- **Χαρακτήρας**. Ο τύπος αυτός αναφέρεται τόσο σε **ένα χαρακτήρα** όσο και σε μία **σειρά** χαρακτήρων. Περιέχει οποιοδήποτε χαρακτήρα μπορεί να γραφεί στο πληκτρολόγιο. Οι χαρακτήρες πρέπει να βρίσκονται υποχρεωτικά ανάμεσα σε **μονά** εισαγωγικά. Επειδή μπορεί να περιέχει και αριθμούς ονομάζεται συχνά και **αλφαριθμητικός** τύπος.

{Πχ ‘ αριθμός ’ ‘ μαθητής 3ος’ }

53. (συμβολικές) Σταθερές

{Ορισμός} Είναι **προκαθορισμένες** τιμές που δεν **μεταβάλλονται** κατά την διάρκεια **εκτέλεσης** του προγράμματος. Μπορεί να είναι τύπου ακέραιες, πραγματικές, λογικές, χαρακτήρες.

Η χρήση σταθερών κάνει το πρόγραμμα πιο κατανοητό, άρα πιο εύκολο να συντηρηθεί και να διορθωθεί.

54. Δήλωση Σταθερών

Η δήλωση των σταθερών γίνεται στο τμήμα δήλωσης σταθερών:

- **Σύνταξη**:

  Σταθερές

  Όνομα-1=σταθερή-τιμή-1

  Όνομα-2=σταθερή-τιμή-2

  …

  Ονομα-ν=σταθερή-τιμή-ν

- **Παράδειγμα:**
  Σταθερές
  Π=3.14
  Όνομα=’Κώστας’
- **Λειτουργία** του τμήματος δήλωσης σταθερών:
  Αποδίδει ονόματα σε σταθερές τιμές. Κάθε μία από αυτές τις σταθερές μπορεί να χρησιμοποιηθεί οπουδήποτε στο πρόγραμμα αλλά δεν είναι δυνατή η μεταβολή της τιμής της.

55. Μεταβλητές

{Ορισμός} Μια μεταβλητή παριστάνει μία **ποσότητα** που η **τιμή** της μπορεί να **μεταβάλλεται**.

Οι μεταβλητές που χρησιμοποιούνται σε ένα πρόγραμμα αντιστοιχούνται από τον **μεταγλωττιστή** σε συγκεκριμένες **θέσεις** μνήμης.

Η τιμή της μεταβλητής είναι η τιμή της **αντίστοιχης** θέσης μνήμης.

Ενώ η τιμή της μεταβλητής (θέσης μνήμης) μπορεί να αλλάξει, ο **τύπος** της **δεν** αλλάζει ποτέ.

Η **ΓΛΩΣΣΑ** επιτρέπει την χρήση **τεσσάρων** τύπων δεδομένων (ακέραιες λογικές πραγματικές χαρακτήρες)

Το **όνομα** κάθε μεταβλητής ακολουθεί τους **κανόνες** δημιουργίας ονομάτων.

Είναι καλή πρακτική να χρησιμοποιούμε ονόματα μεταβλητών που **υπονοούν** το περιεχόμενό τους. Πχ «εμβαδόν» και όχι σκέτο «Ε»

56. Δήλωση Μεταβλητών

(Η δήλωση των μεταβλητών γίνεται στο τμήμα δήλωσης μεταβλητών)

- **Σύνταξη** του τμήματος δήλωσης:
  Μεταβλητές
  Ακέραιες: Λίστα-μεταβλητών-1
  Πραγματικές: Λίστα-μεταβλητών-2
  Λογικές: Λίστα-μεταβλητών-3
  Χαρακτήρες: Λίστα-μεταβλητών-4
- **Λειτουργία** του τμήματος δήλωσης:

Δηλώνει τον **τύπο όλων** των μεταβλητών που χρησιμοποιούνται στο πρόγραμμα

57. Ονόματα προγραμμάτων και δεδομενων

Κάθε πρόγραμμα όπως και τα δεδομένα (μεταβλητές και σταθερές) που χρησιμοποιεί έχουν ένα όνομα με το οποίο αναφερόμαστε σε αυτό.

Τα ονόματα αυτά μπορούν να αποτελούνται από:

- Γράμματα:
  Κεφαλαία και μικρά ελληνικού αλφαβήτου (Α-Ω , α-ω)
  Κεφαλαία και μικρά αγγλικού αλφαβήτου (A-Z, a-z)
- Ψηφία:
  0-9
- Ειδικούς χαρακτήρες:
  _ (κάτω παύλα ή underscore)
- Περιορισμοί:
  - Απαγορεύεται η χρήση δεσμευμένων λέξεων ως ονόματα (πχ γράψε διάβασε)
  - Τα ονόματα υποχρεωτικά αρχίζουν από γράμμα (όχι ψηφίο ούτε underscore)

58. Αριθμητικοί τελεστές

Τελεστής	Αριθμητική πράξη
^	Ύψωση στην δύναμη
+	Πρόσθεση
-	Αφαίρεση
*	Πολλαπλασιασμός
/	Διαίρεση
mod	**Υπόλοιπο** ακέραιας διαίρεσης
div	**Πηλίκο** ακέραιας διαίρεσης

59. Ιεραρχία αριθμητικών τελεστών

1. ^
2. * / div mod
3. + -

Προσοχή:

- Αν συναντάμε **παρενθέσεις** προηγούνται οι τελεστές εντός των παρενθέσεων.
- Όσοι τελεστές έχουν **ίδια** ιεραρχία εκτελούνται από **αριστερά** προς τα δεξιά (όπως στα μαθηματικά).

60. Ενσωματομένες (εγγενείς) συναρτήσεις της ΓΛΩΣΣΑΣ

ΗΜ(x)	ΣΥΝ(x)	ΕΦ(x)	ΛΟΓ(x)	Ε(χ)	Τ_Ρ(x)	Α_Μ(x)	Α_Τ(x)
Ημίτονο	Συνημίτονο	Εφαπτομένη	Φυσικός Λογάριθμος	Εκθετική	Ρίζα	Ακέραιο μέρος	Απόλυτη τιμή

{ Προσοχή:
Η ΛΟΓ(χ) δεν βρίσκει τον δεκαδικό λογάριθμο αλλά τον φυσικό, δηλαδή το lnx
Οι τριγωνομετρικές συναρτήσεις δέχονται ως είσοδο μοίρες και όχι ακτίνια
Οι συναρτήσεις αυτές είναι υποπρογράμματα τύπου συνάρτηση. (Βλέπε κεφάλαιο 10).
Θεωρούνται ενσωματωμένες στην ΓΛΩΣΣΑ, άρα μπορούμε να τις χρησιμοποιούμε οπουδήποτε τις χρειαζόμαστε }

61. Αριθμητικές εκφράσεις.

{βλέπε και κεφάλαιο 2ο για εκφράσεις γενικότερα}

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

Για την **σύνταξη** μιας αριθμητικής έκφρασης χρησιμοποιούνται:

- Αριθμητικές **σταθερές**. *{ πχ 3 }*
- Αριθμητικές **μεταβλητές**. { πχ x }
- Αριθμητικοί **τελεστές** *{ πχ + }*
- **Συναρτήσεις**. *{ πχ Α_Τ( ) }*
- Παρενθέσεις.

Κάθε έκφραση παριστάνει μία αριθμητική τιμή η οποία βρίσκεται **μετά** την εκτέλεση των πράξεων. Γι' αυτό είναι **απαραίτητο** όλες οι μεταβλητές που εμφανίζονται σε μία έκφραση να έχουν πάρει **προηγούμενα** κάποια τιμή.

Η εκτέλεση των πράξεων γίνεται με βάση την **ιεραρχία** των αριθμητικών τελεστών.

Οι παρενθέσεις πάντα χρησιμοποιούνται σε **ζεύγη**. **Διαφορετικός** αριθμός αριστερών από δεξιές παρενθέσεις στην ίδια έκφραση είναι ένα από τα πιο συνηθισμένα **συντακτικά** λάθη.

62. Εντολή εκχώρησης

Χρησιμοποιείται για την **απόδοση** τιμών στις **μεταβλητές** κατά την **εκτέλεση** του προγράμματος.

Σε μία εντολή εκχώρησης η μεταβλητή και η έκφραση πρέπει να είναι του **ιδίου τύπου**.

Το σύμβολο "←" διαφοροποιείται από το = το οποίο χρησιμοποιείται ως **συγκριτικός** τελεστής καθώς και ως τελεστής απόδοσης τιμών στο τμήμα δηλώσεων των σταθερών.

- Σύνταξη: Όνομα_μεταβλητής ← έκφραση
- Λειτουργία: Υπολογίζεται η έκφραση δεξιά του ← και η τιμή της εκχωρείται στην μεταβλητή που βρίσκεται αριστερά.
- Παράδειγμα: Α←'όμορφα'

63. Εντολή εισόδου (διάβασε)

- Σύνταξη: Διάβασε λίστα-μεταβλητών
- Λειτουργία: Η εντολή οδηγεί στην είσοδο τιμών από το πληκτρολόγιο και την εκχώρηση τους στις μεταβλητές που αναφέρονται στην λίστα.
- Παράδειγμα: διάβασε χ,ψ

Ποιό αναλυτικά:

Η εντολή διάβασε ακολουθείται πάντα από τουλάχιστον ένα **όνομα μεταβλητής**. Αν υπάρχουν περισσότερες τότε αυτές χωρίζονται με **κόμμα** (,). **Διακόπτεται** η εκτέλεση του προγράμματος καθώς το πρόγραμμα **περιμένει** την εισαγωγή τιμών από το **πληκτρολόγιο**. Μόλις εισαχθούν οι τιμές συνεχίζεται η εκτέλεση του προγράμματος στην **επόμενη** εντολή. Εξυπηρετεί την **επικοινωνία** του **χρήστη** με τον υπολογιστή.

64. Εντολή εξόδου (γράψε)

- Σύνταξη: Γράψε λίστα-στοιχείων
- Λειτουργία: Εμφανίζει σταθερές τιμές καθώς και τιμές μεταβλητών (ότι αναγράφεται στην λίστα)
- Παράδειγμα: Γράψε ' Ο φόρος είναι ' , φπα, ' € '

Ποιό αναλυτικά: Έχει ως αποτέλεσμα την εμφάνιση τιμών στην **μονάδα εξόδου**. Η μονάδα εξόδου μπορεί να είναι η **οθόνη**, ο **εκτυπωτής**, ή γενικά οποιαδήποτε μονάδα εξόδου έχει **οριστεί**. Η λίστα στοιχείων μπορεί να περιέχει **σταθερές** τιμές και **ονόματα** μεταβλητών. Όταν κάποιο όνομα μεταβλητής περιέχεται στην λίστα τότε αρχικά **ανακτάται** η τιμή της

και μετά η τιμή αυτή **εμφανίζεται** στην μονάδα εξόδου. Η χρήση της αναφέρεται κυρίως στην εμφάνιση **μηνυμάτων** και **αποτελεσμάτων.** Εξυπηρετεί την **επικοινωνία** του **υπολογιστή** με τον χρήστη.

65. Δομή προγράμματος σε ΓΛΩΣΣΑ

- **Επικεφαλίδα**: Πρόγραμμα Όνομα-προγράμματος Το όνομα πρέπει να ακολουθεί τους κανόνες δημιουργίας ονομάτων
- **Τμήμα δήλωσης σταθερών**: Εδώ δηλώνονται οι τυχόν σταθερές που χρησιμοποιεί το πρόγραμμα.
- **Τμήμα δήλωσης μεταβλητών**:Δηλώνονται υποχρεωτικά τα ονόματα όλων των μεταβλητών μαζί με τον τύπο τους.
- **Κύριο μέρος**: Περιλαμβάνει όλες τις εκτελέσιμες εντολές ανάμεσα στην αρχή και στο τέλος_προγράμματος.

Παρατηρήσεις:

Κάθε εντολή γράφεται σε **ξεχωριστή γραμμή**. Αν κάποια εντολή πρέπει να συνεχιστεί σε επόμενη γραμμή τότε ο πρώτος χαρακτήρας της επόμενης πρέπει να είναι ο: **&.**

Αν ο **πρώτος** χαρακτήρας μίας γραμμής είναι ο **!** τότε η γραμμή αυτή αποτελεί **σχόλιο** και **δεν εκτελείται**

Αν το πρόγραμμα χρησιμοποιεί **υποπρογράμματα** τότε αυτά γράφονται **μετά** το **τέλος_προγράμματος**

{ Education is what remains after one has forgotten what one has learned in school.

Albert Einstein }

ΚΕΦΑΛΑΙΟ 8: ΕΠΙΛΟΓΉ ΚΑΙ ΕΠΑΝΆΛΗΨΗ

66. Τι είναι λογική έκφραση:

Μία λογική έκφραση είναι μία έκφραση που έχει λογική τιμή δηλαδή αληθής ή ψευδής. Για την σύνταξη μιας λογικής έκφρασης χρησιμοποιούνται: Σταθερές, μεταβλητές, αριθμητικές εκφράσεις, συγκριτικοί τελεστές, λογικοί τελεστές, και παρενθέσεις

67. Ποιοί είναι οι συγκριτικοί τελεστές

Τελεστής	Ελεγχόμενη σχέση
=	Ισότητα
<>	Ανισότητα
>	Μεγαλύτερο
<	Μικρότερο
>=	Μεγαλύτερο ή ίσο
<=	Μικρότερο ή ίσο

68. Συγκρίσεις

Οι συγκρίσεις έχουν ως αποτέλεσμα μία λογική τιμή (Αληθής-ψευδής)

Γίνονται σε δεδομένα όλων των τύπων της ΓΛΩΣΣΑΣ :

- Σύγκριση **αριθμών**: Η σύγκριση μεταξύ δύο αριθμών γίνεται με **προφανή** τρόπο.
- Σύγκριση **χαρακτήρων**: Η σύγκριση μεταξύ **ατομικών** χαρακτήρων βασίζεται στην **αλφαβητική** σειρά ΠΧ το “α” < “β” : αληθής Η σύγκριση **σειράς** χαρακτήρων βασίζεται στην σύγκριση χαρακτήρα προς χαρακτήρα σε κάθε θέση μέχρι να βρεθεί κάποια **διαφορά**.
  { Πχ “κακός” < “καλός” : αληθής (γιατί το κ είναι μικρότερο του λ) }
- Σύγκριση **Λογικών**: Η σύγκριση μεταξύ λογικών έχει **νόημα** μόνο στην περίπτωση του **=** και του **<>**

69. Σύνθετες λογικές εκφράσεις

Σχηματίζονται από **απλές** λογικές εκφράσεις με την χρήση των λογικών τελεστών: Όχι, Και,Η

70. Σχετική ιεραρχία των τελεστών

1. αριθμητικοί τελεστές,
2. συγκριτικοί,
3. λογικοί.

71. Δομή επιλογής

Η δομή επιλογής υλοποιείται με την εντολή “Αν”. Η εντολή “Αν” έχει τρεις διαφορετικές μορφές:

Απλή επιλογή	Σύνθετη επιλογή	Πολλαπλή επιλογή
Αν .. τότε	Αν..τότε αλλιώς	Αν..τότε αλλιώς_αν

Προσοχή: Κάθε εντολή Αν πρέπει να κλείνει με τέλος_αν

*{προσοχή όμως εδώ: Σύμφωνα με το σχολικό σου (κεφάλαιο 2) σε μια ψευδογλώσσα όταν οι εντολές εντός μιας απλής επιλογής είναι μια και **μόνο** μία, δικαιούμαστε να μην βάλουμε τελος_αν αρκεί να γράψουμε την εντολή δίπλα από την λέξη τότε.*
Παράδειγμα:
Οι δυο αλγόριθμοι είναι ισοδύναμοι:

```
αλγόριθμος ααα
        Διάβασε χ
        αν χ > 100 τότε
                γράψε 'μεγάλος'
        τελος_αν
        γράψε χ
τελος ααα
αλγόριθμος ααα
        Διάβασε χ
        αν χ > 100 τότε γράψε 'μεγάλος'
τελος ααα
}
```

72. Δομή απλής επιλογής (Αν..τότε)

- **Σύνταξη**:

```
Αν συνθήκη τότε
    Εντολή-1
    Εντολή-2
    ..
    Εντολή-ν
τελος_αν
```

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

73. Δομή σύνθετης επιλογής (Αν..τότε.αλλιώς)

- **Σύνταξη**:

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

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

74. Δομή πολλαπλής επιλογής (Αν..τότε.αλλιώς_αν)

- **Σύνταξη**:

```
Αν συνθήκη-1 τότε
      Ομάδα εντολών 1
Αλλιώς_αν συνθήκη-2 τότε
      Ομάδα εντολών 2
Αλλιώς
      Ομάδα εντολών ν
τελος_αν
```

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

75. Εμφωλευμένες δομές επιλογής

{Ορισμός:} Ονομάζονται **δύο** ή **περισσότερες** εντολές Αν που περιέχονται η μία μέσα στην άλλη.

Η χρήση εμφωλευμένων αν οδηγεί **συνήθως** σε **πολύπλοκες** δομές που αυξάνουν την **πιθανότητα** λάθους και κάνουν το πρόγραμμα **δυσνόητο**.

{εμφωλευμένες μπορεί να είναι και εντολές επανάληψης αλλά και συνδυασμός δομών επιλογής και επανάληψης}

76. Δομή επανάληψης

Η δομή επανάληψης έχει τρεις μορφές-εντολές:

Την δομή «Για» την δομή «όσο» και την δομή «μέχρις_ότου».

Όλες ελέγχονται από μία συνθήκη ή οποία καθορίζει την έξοδο από τον **βρόχο**. *{ Προσοχή: βρόχος γενικά σημαίνει θηλειά αλλά στην πληροφορική σημαίνει δομή επανάληψης }*

77. Δομή οσο..επανάλαβε

- **Σύνταξη**:

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

- **Λειτουργία**
  - **Ελέγχεται** η συνθήκη και αν είναι **αληθής** εκτελούνται οι εντολές που βρίσκονται **ανάμεσα** στις όσο και τέλος_επανάληψης.
  - Στην συνέχεια **ελέγχεται πάλι** η συνθήκη και αν ισχύει εκτελούνται πάλι οι ίδιες εντολές.
  - Όταν η συνθήκη γίνει **ψευδής** τότε **σταματά** η επανάληψη και εκτελείται η **εντολή** μετά το τέλος_επανάληψης. *{αν βέβαια υπάρχει}*
  - Με την δομή ΟΣΟ μπορούν να εκφραστούν **όλες** οι άλλες δομές επανάληψης
  - Χαρακτηριστικό της είναι ότι ο αριθμός των επαναλήψεων **δεν** είναι γνωστός.
  - Για να σταματήσει η επανάληψη πρέπει **υποχρεωτικά** μέσα στον βρόχο να υπάρχει μία **εντολή** η οποία κάνει την συνθήκη **ψευδή**.

78. Δομή μέχρις_οτου

- **Σύνταξη**:

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

- **Λειτουργία**
  - Εκτελούνται οι εντολές **μεταξύ** των αρχή_επανάληψης και μέχρις_οτου.
  - Στη συνέχεια **ελέγχεται** η συνθήκη και αν είναι **ψευδής** τότε εκτελούνται **πάλι** οι εντολές.
  - Όταν η συνθήκη βρεθεί **αληθής** τότε **σταματά** η επανάληψη και εκτελείται η εντολή **μετά** το μέχρις_ότου..
  - Η δομή αυτή εκτελείται **τουλάχιστον** μια φορά.

79. Δομή για

- **Σύνταξη**:

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

- **Λειτουργία**
  - Οι εντολές του βρόχου εκτελούνται για **όλες** τις τιμές της μεταβλητής από την τιμή1 μέχρι την τιμή2 αυξανόμενες κατά την τιμή3.
  - Αν το βήμα είναι **1** τότε **παραλείπεται**
  - Χρησιμοποιείται όταν γνωρίζουμε εκ των **προτέρων** τον αριθμό των επαναλήψεων.

80. Τι ονομάζουμε τιμή φρουρό;

Η χρήση τιμών για τον **τερματισμό** μιας 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.

81. Εμφωλευμένοι βρόχοι και κανόνες χρήσης

{Ορισμός} Ονομάζονται

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

82. Κανόνες χρήσης εμφωλευμένων βρόχων

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

{ Just believe in yourself. Even if you don't, pretend that you do and at some point you will.

Venus Williams }

ΚΕΦΑΛΑΙΟ 9: ΠΊΝΑΚΕΣ

83. Τι είναι πινάκας

{Ορισμός} Είναι

- ένα **σύνολο** αντικειμένων **ιδίου τύπου** τα οποία
- αναφέρονται με ένα **κοινό όνομα**.

Παρατηρήσεις

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 **[ ]**

Δήλωση 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.

Στοιχείο:

Η αναφορά σε ένα στοιχείο το πίνακα γίνεται με το **όνομά** του πίνακα ακολουθούμενο από ένα **δείκτη**.

Κάθε ένα από τα **αντικείμενα** που τον αποτελούν ονομάζεται **στοιχείο**.

Κάθε **συγκεκριμένη θέση μνήμης** καλείται **στοιχείο** του πίνακα και προσδιορίζεται από την τιμή **ενός** δείκτη αν ο πίνακας είναι **μονοδιάστατος**. Στην περίπτωση που είναι **δισδιάστατος** προσδιορίζεται από **δύο** δείκτες. Αν είναι **ν-διαστατος** από **ν δείκτες**.

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.

Πολυδιάστατοι πίνακες

Αν ο καθορισμός των στοιχείων ενός πίνακα απαιτεί παραπάνω από ένα δείκτη τότε μιλάμε για πολυδιάστατους πίνακες: πχ 3 δείκτες τρισδιάστατος 2 δείκτες δισδιάστατος κλπ

Χρήση πινάκων:

Η ανάγνωση, επεξεργασία και εμφάνιση των στοιχείων των πινάκων γίνεται πάντοτε από βρόχους. Συνήθως με την χρήση της δομής “Για” αφού είναι προκαθορισμένο το πλήθος των στοιχείων

84. Μειονεκτήματα από την χρήση πινάκων

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

85. Πότε πρέπει να χρησιμοποιούνται πίνακες

1. Όταν τα **δεδομένα** που εισάγονται σε ένα πρόγραμμα πρέπει να διατηρούνται στην μνήμη μέχρι το **τέλος** της εκτέλεσης.
2. Η **απόφαση** για την χρήση ή όχι πίνακα είναι θέμα **εμπειρίας** στον προγραμματισμό.

86. Τυπικές επεξεργασίες πινάκων

*{ Καλό είναι να τις μελετήσεις σε συνδυασμό με τις τυπικές επεξεργασίες των **δομών δεδομένων** βλ. Κεφάλαιο 3 }*

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

{ Do not be anxious about tomorrow, for tomorrow will be anxious for itself. Let the day's own trouble be sufficient for the day.

Jesus Christ }

ΚΕΦΑΛΑΙΟ 10: ΥΠΟΠΡΟΓΡΆΜΜΑΤΑ

87. Τμηματικός προγραμματισμός

{ ορισμός: } Ονομάζεται η **τεχνική σχεδίασης** και **ανάπτυξης** των προγραμμάτων ως ένα **σύνολο** από **απλούστερα τμήματα** προγραμμάτων.

Η **ιεραρχική** σχεδίαση, υλοποιείται με τον **τμηματικό** προγραμματισμό. Αφού **αναλυθεί** ένα πρόβλημα σε **υποπροβλήματα**, κάθε υποπρόβλημα γράφεται σαν **ανεξάρτητη οντότητα** (module)

{ κάθε module είναι ένα υποπρόγραμμα }

Αποτελεί ένα από τα **βασικά συστατικά** του **δομημένου** προγραμματισμού.

88. Υποπρόγραμμα

{ ορισμός: } Ονομάζεται ένα **τμήμα** προγράμματος το οποίο επιτελεί ένα **αυτόνομο έργο** και έχει **γραφεί χωριστά** από το **υπόλοιπο** πρόγραμμα.

89. Χαρακτηριστικά των υποπρογραμμάτων

1) **Κάθε υπ/μα πρέπει να έχει μόνο μία είσοδο και μια έξοδο.**

Η **είσοδος** σε αυτό γίνεται μόνο από την **αρχή** του και αφού εκτελέσει κάποιες ενέργειες, **απενεργοποιείται** στην **έξοδο** από αυτό η οποία γίνεται μόνο στο **τέλος** του.

{ Προσοχή. Αυτό δεν σημαίνει ότι μια διαδικασία , για παράδειγμα, έχει μόνο μια είσοδο ή μόνο μία έξοδο. Στην πραγματικότητα γνωρίζουμε ότι μπορεί να έχει πολλές εισόδους και εξόδους. Αυτό που εννοεί εδώ το σχολικό είναι ότι δεν είναι δυνατόν να ενεργοποιηθεί το υποπρόγραμμα πουθενά αλλού εκτός από την αρχή του. Όμοια σταματά μόνο όταν εκτελεστεί η εντολή: τέλος_διαδικασίας.. Σε κανένα άλλο σημείο δεν μπορεί να σταματήσει. Άρα έχει μία και μόνο έξοδο. }

2) **Κάθε υπ/μα πρέπει να είναι ανεξάρτητο από τα άλλα.**

Κάθε υπ/μα μπορεί να **σχεδιαστεί**, να **αναπτυχθεί** και να **συντηρηθεί αυτόνομα**,χωρίς να **επηρεαστούν** άλλα υποπρογράμματα. Στην **πράξη** όμως, η απόλυτη ανεξαρτησία είναι **δύσκολο** να γίνει.

3) **Κάθε υπ/μα πρέπει να μην είναι πολύ μεγάλο.**

Η έννοια του μεγάλου είναι **υποκειμενική**. Γενικά κάθε υπ/μα πρέπει να εκτελεί **μόνο μια λειτουργία**. Αν εκτελεί **παραπάνω** από μία, τότε συνήθως μπορεί να **διασπαστεί** σε μικρότερα υποπρογράμματα.

90. Πλεονεκτήματα του τμηματικού προγραμματισμού.

1) **Διευκολύνει την ανάπτυξη του προγράμματος.**

- Αυτό γιατί επιτρέπει την εξέταση και επίλυση **απλών** προβλημάτων και όχι την κατά- μέτωπο αντιμετώπιση του **συνολικού** προβλήματος.
- Με την **σταδιακή** αντιμετώπιση των **υποπροβλημάτων** επιλύεται τελικά και το **συνολικό**

2) **Διευκολύνει την κατανόηση και διόρθωση του προγράμματος.**

- Η **διόρθωση** γίνεται πιο γρήγορα και εύκολα αφού διορθώνεται μόνο ένα συγκεκριμένο **τμήμα** του προγράμματος, χωρίς οι αλλαγές αυτές να **επηρεάσουν** το υπόλοιπο πρόγραμμα.
- Η **κατανόηση** του προγράμματος από **τρίτους** είναι πολύ πιο εύκολη

{... είναι πιο εύκολο να εξετάζεις πολλά μικρά κομματάκια αλγορίθμων που κάνουν μια συγκεκριμένη ενέργεια, παρά ένα τεράστιο αλγόριθμο που τα κάνει όλα! }

3) **Απαιτεί λιγότερο κόπο και χρόνο στην συγγραφή του προγράμματος.**

- Πολύ συχνά σε **διαφορετικά** σημεία ενός προγράμματος χρειάζεται να εκτελεστεί η **ίδια** λειτουργία.
- Από την στιγμή, όμως που ένα υπ/μα έχει γραφεί, μπορεί να χρησιμοποιηθεί **πολλές** φορές.
- Έτσι, **χρησιμοποιώντας** στο **ίδιο** πρόγραμμα το **ίδιο** υποπρόγραμμα πολλές φορές μειώνεται το **μέγεθος** του προγράμματος.
- Μειώνεται παράλληλα ο **κόπος** για την συγγραφή του και ο **χρόνος** και οι **πιθανότητες λάθους**.

4) **Επεκτείνει τις δυνατότητες των γλωσσών προγραμματισμού.**

- Η **χρήση** ενός υποπρογράμματος δεν διαφέρει από την χρήση των **ενσωματωμένων** συναρτήσεων (A_T , T_P, κλπ).
- Έτσι αν χρειάζεται κάποια λειτουργία που δεν υποστηρίζεται **απευθείας** από την γλώσσα προγραμματισμού τότε μπορεί να **γραφεί** το αντίστοιχο υπ/μα.
- Η συγγραφή **πολλών** υποπρογραμμάτων και η δημιουργία **βιβλιοθηκών** με αυτά **επεκτείνουν** την ίδια την γλώσσα προγραμματισμού.

91. Παράμετροι

{Ορισμός:} Μια παράμετρος είναι

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

{Είναι σαν τις κοινές μεταβλητές ενός προγράμματος με μια ουσιώδη διαφορά: χρησιμοποιούνται για να περνούν τιμές από και προς τα υποπρογράμματα.}

92. Σύντομη περιγραφή της λειτουργίας ενός υποπρογράμματος.

Κάθε υποπρόγραμμα για να ενεργοποιηθεί **καλείται** , όπως λέγεται, από ένα **άλλο** υποπρόγραμμα ή από το **αρχικό** πρόγραμμα. (Το αρχικό πρόγραμμα ονομάζεται **κυρίως πρόγραμμα**.)

Το υπ/μα είναι **αυτόνομο** και **ανεξάρτητο** αλλά πρέπει να **επικοινωνεί** με το υπόλοιπο πρόγραμμα.

Συνήθως **δέχεται** τιμές από το πρόγραμμα που το **καλεί** και μετά την εκτέλεση του **επιστρέφει** σε αυτόν που το κάλεσε **νέες τιμές- αποτελέσματα.**

Οι τιμές αυτές **περνούν** από το ένα υπ/μα στο **άλλο** με την βοήθεια των **παραμέτρων.**

93. Τι είναι συνάρτηση.

{Ορισμός:} Είναι ένας **τύπος** υποπρογράμματος που υπολογίζει και επιστρέφει **μία** μόνο τιμή με το **όνομά** της.

94. Τι είναι διαδικασία.

(Ορισμός:) Είναι ένας **τύπος** υποπρογράμματος που μπορεί να **εκτελεί όλες** τις λειτουργίες ενός **προγράμματος**.

95. Δομή συνάρτησης:

Κάθε συνάρτηση έχει την ακόλουθη δομή και ορίζεται **μετά** από το τέλος_προγράμματος:

```
Συνάρτηση όνομα (λίστα τυπικών παραμέτρων): τύπος συνάρτησης
τμήμα δηλώσεων
Αρχή
...
όνομα←έκφραση
```

Ονομα: Το όνομα της συνάρτησης είναι οποιοδήποτε έγκυρο όνομα της ΓΛΩΣΣΑΣ.

Η λίστα τυπικών παραμέτρων: είναι μια λίστα μεταβλητών των οποίων οι τιμές μεταβιβάζονται στην συνάρτηση. *{Αποτελείται από τις μεταβλητές εισόδου της συνάρτησης}*

Ο τύπος της συνάρτησης: είναι ο τύπος της τιμής που υπολογίζει η συνάρτηση. Μπορεί να είναι ακέραια, λογική, πραγματική, ή χαρακτήρας. *{θυμήσου ότι η συνάρτηση επιστρέφει μία μόνο τιμή}*

Εντολές: Στις εντολές του σώματος της συνάρτησης πρέπει υποχρεωτικά να υπάρχει μια εντολή εκχώρησης τιμής στο όνομα της συνάρτησης. (όνομα←έκφραση) *{Αυτή η εκχώρηση αποτελεί ουσιαστικά τον μηχανισμό εξόδου της συνάρτησης}*

96. Δομή διαδικασίας:

Κάθε διαδικασία έχει την ακόλουθη δομή και ορίζεται **μετά** από το τέλος_προγράμματος:

```
Διαδικασία όνομα (λίστα τυπικών παραμέτρων)
τμήμα δηλώσεων
Αρχή
...
```

Ονομα: Το όνομα της διαδικασίας είναι οποιοδήποτε έγκυρο όνομα της γλώσσας.

Η λίστα τυπικών παραμέτρων: είναι μια λίστα μεταβλητών, οι τιμές των οποίων μεταβιβάζονται προς την διαδικασία ή/και επιστρέφονται στο κυρίως πρόγραμμα μετά το τέλος της διαδικασίας. *{Πρόσεχε το ή/και.. κρύβονται δύο προτάσεις σε μία}*

Εντολές: Στο σώμα της διαδικασίας μπορούν να υπάρχουν οποιεσδήποτε εντολές της ΓΛΩΣΣΑΣ.

97. Κλήση συναρτήσεων.

Κάθε συνάρτηση καλείται όπως ακριβώς καλούνται οι ενσωματωμένες συναρτήσεις της ΓΛΩΣΣΑΣ (A_T, T_P κλπ.)
Απλώς αναφέρεται το όνομά της σε μια έκφραση ή σε μια εντολή και επιστρέφεται η τιμή της.
Παράδειγμα:
Ε←Εμβαδό_κύκλου(R)

Το κύριο πρόγραμμα πριν την κλήση της συνάρτηση γνωρίζει την τιμή της μεταβλητής R. Κατά την κλήση της συνάρτηση μεταβιβάζεται η τιμή του R στην αντίστοιχη μεταβλητή της συνάρτησης. Η συνάρτηση υπολογίζει το εμβαδόν και το αποτέλεσμα εκχωρείται στο όνομά της. Με το τέλος_συνάρτησης γίνεται επιστροφή στο κυρίως πρόγραμμα όπου η τιμή του εμβαδού εκχωρείται στην μεταβλητή Ε

98. Κλήση διαδικασιών

Η κλήση μιας διαδικασίας από ένα πρόγραμμα ή από ένα άλλο υποπρόγραμμα γίνεται με την εντολή κάλεσε.

- **Σύνταξη:** Κάλεσε όνομα_διαδικασίας(λίστα πραγματικών παραμέτρων)
- **Λειτουργία:** Η εκτέλεση του προγράμματος **διακόπτεται**. **Εκτελούνται** οι εντολές της διαδικασίας. Μετά το τέλος της διαδικασίας η εκτέλεση του προγράμματος **συνεχίζεται** με την εντολή που βρίσκεται **μετά** την κάλεσε.
- **Λίστα παραμέτρων:** Μπορεί να περιλαμβάνει **καμία**, **μία** ή **περισσότερες** παραμέτρους. Ορίζει τις τιμές **εισόδου** προς την διαδικασία αλλά **και** τις τιμές β της διαδικασίας. *{παρατήρησε ότι η λίστα παραμέτρων μπορεί να μην περιλαμβάνει και καμία παράμετρο}*

99. Πραγματικές παράμετροι.

{Ορισμός} Η λίστα των πραγματικών παραμέτρων **καθορίζει** τις παραμέτρους στην **κλήση** ενός υποπρογράμματος.

100. Τυπικές παράμετροι.

{ορισμος} Η λίστα των τυπικών παραμέτρων **καθορίζει** τις παραμέτρους στην **δήλωση** ενός υποπρογράμματος. **Μερικές** γλώσσες προγραμματισμού τις ονομάζουν και **ορίσματα**.

101. Κανόνες παραμέτρων.

1. Ο **αριθμός** των πραγματικών και τυπικών παραμέτρων πρέπει να είναι ο ίδιος.
2. Κάθε πραγματική παράμετρος **αντιστοιχεί** στην αντίστοιχη τυπική.
3. Η πραγματική παράμετρος και η **αντίστοιχή** της τυπική πρέπει να είναι του **ίδιου τύπου**.

{ Εξαίρεση στον τρίτο κανόνα αποτελεί το σενάριο όπου η πραγματική παράμετρος είναι ακέραια και η τυπική παράμετρος είναι πραγματική. Το σχολικό εγχειρίδιο όμως δεν αναφέρει πουθενά κάτι τέτοιο. Για τον λόγο αυτό μην το αναφέρεις :) }

102. Χρήση στοίβας στην κλήση διαδικασιών.

Όλες οι **εντολές** ενός προγράμματος έχουν μια μοναδική "**διεύθυνση**". Όταν μια **διαδικασία ή μία συνάρτηση** καλείται από το κύριο πρόγραμμα, τότε η **διεύθυνση** της **αμέσως επόμενης εντολής** του κυρίου προγράμματος (που ονομάζεται **διεύθυνση επιστροφής**) **αποθηκεύεται** από τον **μεταφραστή** σε μια **στοίβα**. Η στοίβα αυτή ονομάζεται **στοίβα χρόνου εκτέλεσης**. **Μετά** την **εκτέλεση** της διαδικασίας ή της συνάρτησης, τότε η "**διεύθυνση επιστροφής**" **απωθείται** από την στοίβα και το πρόγραμμα εκτελεί την **αντίστοιχη** εντολή. Η **τεχνική** αυτή εφαρμόζεται **γενικότερα**, δηλαδή **οποτεδήποτε** μια διαδικασία η συνάρτηση **καλεί μια άλλη** διαδικασία ή συνάρτηση.

103. Τοπικές μεταβλητές

- Κάθε **κύριο** πρόγραμμα όπως και κάθε **υποπρόγραμμα** περιλαμβάνει τις **δικές** του **μεταβλητές** και **σταθερές**. **Ολες** οι μεταβλητές και **όλες** οι σταθερές στη ΓΛΩΣΣΑ είναι **γνωστές** στο αντίστοιχο προγραμμα/υποπρόγραμμα που

δηλώνονται και **μόνο** σε αυτό. Είναι δηλαδή **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 **διαμέσου των παραμέτρων** κατά το στάδιο της **κλήσης** του υποπρογράμματος και μετά το **τέλος** της εκτέλεσης του υποπρογράμματος.

104. Τι ονομάζεται εμβέλεια μεταβλητών

{Ορισμός} Εμβέλεια (scope) μεταβλητών λέγεται το **τμήμα** του προγράμματος που **ισχύουν** οι **μεταβλητές**

Πολλές γλώσσες προγραμματισμού επιτρέπουν τη **χρήση** των μεταβλητών και των σταθερών, όχι **μόνο** στο τμήμα προγράμματος που **δηλώνονται**, αλλά και σε **άλλα υποπρογράμματα** ή ακόμη και σε **όλα** τα υπόλοιπα υποπρογράμματα. Αυτό που **καθορίζει** την **περιοχή** που ισχύουν οι μεταβλητές και οι σταθερές είναι η **εμβέλεια** των μεταβλητών της γλώσσας.

105. Απεριόριστη εμβέλεια.

Σύμφωνα με αυτή την αρχή **όλες** οι μεταβλητές και **όλες** οι σταθερές είναι γνωστές και μπορούν να χρησιμοποιούνται σε οποιοδήποτε **τμήμα** του προγράμματος, **άσχετα που δηλώθηκαν**. **Όλες** οι μεταβλητές είναι **καθολικές**.

Μειονεκτήματα:

1. καταστρατηγεί την **αρχή της αυτονομίας** των υποπρογραμμάτων,
2. δημιουργεί **πολλά προβλήματα** και τελικά
3. είναι **αδύνατη** για μεγάλα προγράμματα με **πολλά υποπρογράμματα**, (αφού ο καθένας που γράφει κάποιο υποπρόγραμμα πρέπει να γνωρίζει τα ονόματα όλων των μεταβλητών που χρησιμοποιούνται στα υπόλοιπα υποπρογράμματα.)

106. 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 **ΓΛΩΣΣΑ** έχουμε περιορισμένη εμβέλεια.)

Πλεονεκτήματα:

1. **Απόλυτη αυτονομία όλων** των υποπρογραμμάτων και
2. η **δυνατότητα** να χρησιμοποιείται **οποιοδήποτε όνομα** μεταβλητής/σταθεράς, (χωρίς να ενδιαφέρει αν το ίδιο χρησιμοποιείται σε άλλο υποπρόγραμμα.)

107. Μερικώς περιορισμένη εμβέλεια.

Σύμφωνα με αυτή την αρχή άλλες μεταβλητές είναι τοπικές και άλλες καθολικές. Κάθε γλώσσα προγραμματισμού έχει τους δικούς της κανόνες και μηχανισμούς για τον τρόπο και τις προϋποθέσεις που ορίζονται οι μεταβλητές ως τοπικές ή καθολικές.

Πλεονεκτήματα/ μειονεκτήματα:

1. Η μερικώς περιορισμένη εμβέλεια προσφέρει **μερικά** πλεονεκτήματα στον πεπειραμένο προγραμματιστή, αλλά για τον **αρχάριο περιπλέκει** το πρόγραμμα δυσκολεύοντας την ανάπτυξή του.

{ How strange to use "You only live once" as an excuse to throw it away.

Bill Copeland }