ΘΕΩΡΙΑ του μαθήματος

Ανάπτυξη Εφαρμογών σε Προγραμματιστικό Περιβάλλον

(δεν υποκαθιστά σε καμία περίπτωση το σχολικό εγχειρίδιο,

απλά δρα απλουστευτικά με τη βοήθεια ερωτήσεων)

Πίνακας περιεχομένων

ΚΕΦΑΛΑΙΟ 2 – ΒΑΣΙΚΕΣ ΕΝΝΟΙΕΣ ΑΛΓΟΡΙΘΜΩΝ.

§2.1 Τι είναι αλγόριθμος

1. **Δώστε ένα ορισμό της έννοιας αλγόριθμος.**

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

2. **Ποια είναι τα κριτήρια που πρέπει να ικανοποιεί ένας αλγόριθμος;**

Κάθε αλγόριθμος πρέπει να ικανοποιεί τα επόμενα κριτήρια:

- **Είσοδος (input).** Καμία, μία ή περισσότερες τιμές δεδομένων πρέπει να δίνονται ως είσοδοι στον αλγόριθμο. Η περίπτωση που δεν δίνονται τιμές δεδομένων εμφανίζεται, όταν ο αλγόριθμος δημιουργεί και επεξεργάζεται κάποιες πρωτογενείς τιμές με τη βοήθεια συναρτήσεων παραγωγής τυχαίων αριθμών ή με τη βοήθεια άλλων απλών εντολών.
- **Έξοδος (output).** Ο αλγόριθμος πρέπει να δημιουργεί τουλάχιστον μία τιμή δεδομένων ως αποτέλεσμα προς το χρήστη ή προς έναν αλλο αλγόριθμο.
- ***Καθοριστικότητα (definiteness).** Κάθε εντολή πρέπει να καθορίζεται χωρίς καμία αμφιβολία για τον τρόπο εκτέλεσής της. Λόγου χάριν, μία εντολή διαίρεσης πρέπει να θεωρεί και την περίπτωση, όπου ο διαιρέτης λαμβάνει μηδενική τιμή.
- ***Περατότητα (finiteness).** Ο αλγόριθμος να τελειώνει μετά από πεπερασμένα βήματα εκτέλεσης των εντολών του. Μία διαδικασία που δεν τελειώνει μετά από ένα συγκεκριμένο αριθμό βημάτων δεν αποτελεί αλγόριθμο, αλλά λέγεται απλά υπολογιστική διαδικασία (computational procedure).
- **Αποτελεσματικότητα (effectiveness).** Κάθε μεμονωμένη εντολή του αλγορίθμου να είναι απλή. Αυτό σημαίνει ότι μία εντολή δεν αρκεί να έχει ορισθεί, αλλά πρέπει να είναι και εκτελέσιμη.

§2.3 Περιγραφή και αναπαράσταση αλγορίθμων

3. **Με ποιους τρόπους αναπαρίσταται ένας αλγόριθμος;**

- **με ελεύθερο κείμενο:** αποτελεί τον πιο ανεπεξέργαστο και αδόμητο τρόπο παρουσίασης αλγορίθμου, υπό μορφή οδηγιών σε φυσική γλώσσα (σαν μια εκθεσούλα). Μπορεί να οδηγήσει σε παραβίαση του κριτηρίου της αποτελεσματικότητας.
- **με διαγραμματικές τεχνικές (diagramming techniques):** είναι ένας γραφικός τρόπος παρουσίασης του αλγορίθμου. Από τις διάφορες διαγραμματικές τεχνικές, η πιο παλιά και πιο γνωστή είναι το διάγραμμα ροής (flow chart), που όμως δεν αποτελεί την καλύτερη λύση για την παρουσίαση αλγορίθμων και γι'αυτό εμφανίζεται όλο και σπανιότερα στη βιβλιογραφία και στην πράξη .
- **με φυσική γλώσσα (natural language) κατά βήματα:** είναι ο τρόπος παρουσίασης ενός αλγορίθμου με βήματα ενεργειών σε φυσική γλώσσα. Στην περίπτωση αυτή χρειάζεται προσοχή, γιατί μπορεί να παραβιασθεί το κριτήριο του καθορισμού.

- **με κωδικοποίηση (coding):** είναι ο τρόπος παρουσίασης αλγορίθμου ως ένα πρόγραμμα γραμμένο είτε σε μία ψευδογλώσσα είτε σε κάποια γλώσσα προγραμματισμού που όταν εκτελεσθεί θα δώσει τα ίδια αποτελέσματα με τον αλγόριθμο.

4. Ποια είναι τα χρησιμοποιούμενα γεωμετρικά σχήματα στα διαγράμματα ροής;

- **έλλειψη**, που δηλώνει την αρχή και το τέλος του κάθε αλγορίθμου
- **ρόμβος,** που δηλώνει μια ερώτηση/έλεγχο/συνθήκη με δύο ή περισσότερες εξόδους για απάντηση
- **ορθογώνιο**, που δηλώνει την εκτέλεση μίας ή περισσότερων πράξεων (εντολών εκχώρησης ή κλήσεις διαδικασιών)
- **πλάγιο παραλληλόγραμμο**, που δηλώνει είσοδο ή έξοδο στοιχείων (διάβασε, γράψε, εμφάνισε, εκτύπωσε, τύπωσε)
- **βέλη,** που συνδέουν τα παραπάνω γεωμετρικά σχήματα και δηλώνουν τη σειρά εκτέλεσης των ενεργειών που υποδηλώνουν τα παραπάνω σχήματα.

§2.4 Βασικές συνιστώσες/εντολές ενός αλγορίθμου (αλγοριθμικές δομές)

§2.4.1 Δομή ακολουθίας

5. Τι γνωρίζετε για τη δομή της ακολουθίας;

Η δομή της ακολουθίας είναι η αλγοριθμική δομή εκείνη στην οποία είναι δεδομένη η σειρά εκτέλεσης ενός συνόλου ενεργειών. Με αποκλειστική χρήση της δομής ακολουθίας αντιμετωπίζονται μόνο πολύ απλά προβλήματα.

6. Ποια είναι τα στοιχεία από τα οποία συντίθενται οι εντολές σε έναν αλγόριθμο;

Τα στοιχεία από τα οποία συντίθενται οι εντολές ενός αλγορίθμου είναι οι σταθερές, οι μεταβλητές, οι τελεστές, οι εκφράσεις, οι δεσμευμένες λέξεις που συνδέουν τα παραπάνω και τα σχόλια.

7. Τι γνωρίζετε για τις σταθερές;

Με τον όρο σταθερές αναφερόμαστε σε προκαθορισμένες τιμές που παραμένουν αμετάβλητες σε όλη τη διάρκεια εκτέλεσης ενός αλγορίθμου. Οι τύποι των σταθερών είναι ακέραιες, πραγματικές, χαρακτήρες ή αλφαριθμητικές και λογικές.

Οι ακέραιες σχηματίζονται με τη χρήση αριθμητικών χαρακτήρων-ψηφίων και των συμβόλων + και – ενώ οι πραγματικές με τη χρήση επιπρόσθετα της τελείας '.' ως σημείου υποδιαστολής.

Οι χαρακτήρες ή αλφαριθμητικές σχηματίζονται από οποιουσδήποτε χαρακτήρες μέσα σε απλά ή διπλά εισαγωγικά.

Οι λογικές είναι δύο, οι Αληθής και Ψευδής.

Παραδείγματα σταθερών: 5, 6.5, -3.7, +1, "Γιώργος", Αληθής

8. Τι είναι η μεταβλητή;

Με τον όρο μεταβλητή αναφερόμαστε σε ένα γλωσσικό αντικείμενο (λέξη) που χρησιμοποιείται για να παραστήσει ένα στοιχείο δεδομένου. Κάθε μεταβλητή αντιστοιχίζεται από το μεταγλωττιστή σε μια θέση μνήμης όπου περιέχεται η τιμή της. Σε μια μεταβλητή εκχωρείται μια τιμή, που μπορεί να μεταβάλλεται κατά τη διάρκεια εκτέλεσης ενός αλγορίθμου. Οι μεταβλητές ανάλογα με το είδος της τιμής που μπορούν να λάβουν διακρίνονται σε ακέραιες, πραγματικές, χαρακτήρες και λογικές. Ο τύπος μιας μεταβλητής παραμένει αμετάβλητος κατά τη διάρκεια εκτέλεσης του αλγορίθμου.

Για τη σύνθεση των ονομάτων των μεταβλητών χρησιμοποιούνται οι αλφαβητικοί χαρακτήρες-γράμματα πεζά ή κεφαλαία-λατινικά ή ελληνικά, τα ψηφία και ο χαρακτήρας _ (underscore). Ένα όνομα μεταβλητής ξεκινάει πάντα από γράμμα. Συνίσταται τα ονόματα των μεταβλητών και των σταθερών να ανάγουν στο περιεχόμενο τους. Το όνομα κάθε μεταβλητής είναι μοναδικό για κάθε αλγόριθμο-πρόγραμμα.

Παραδείγματα ονομάτων μεταβλητών: αριθμός_μαθητών, x, x1, βρέθηκε, όνομα, μέσος_όρος

9. Τι γνωρίζετε για τους τελεστές;

Τελεστές είναι τα γνωστά σύμβολα που χρησιμοποιούνται στις διάφορες πράξεις. Διακρίνονται σε αριθμητικούς (+, -, *, /, ^, mod, div) , συγκριτικούς (<, <= ή ≤ , >, >= ή ≥, <> ή ≠) και λογικούς (ΟΧΙ, ΚΑΙ, Ή).

10. Τι γνωρίζετε για τις εκφράσεις;

Οι εκφράσεις σε έναν αλγόριθμο διαμορφώνονται από τους τελεστέους (μεταβλητές, σταθερές, συναρτήσεις), τους τελεστές και τη χρήση παρενθέσεων. Η αποτίμηση μιας έκφρασης συνίσταται στην απόδοση τιμών στις μεταβλητές και στην εκτέλεση πράξεων. Η τελική τιμή της έκφρασης εξαρτάται από την ιεραρχία των πράξεων και τη χρήση παρενθέσεων. Μια έκφραση μπορεί να είναι από μια απλή σταθερά ή μεταβλητή μέχρι μια πολύπλοκη μαθηματική παράσταση.

11. Ποια είναι η γενική μορφή των εντολών εισόδου, εξόδου και εκχώρησης τιμής σε έναν αλγόριθμο και ποια η λειτουργία που επιτελούν;

Εισόδου:**Διάβασε λίστα_μεταβλητών** *(λίστα μεταβλητών: 1 ή περισσότερες μεταβλητές χωριζόμενες με κόμμα)*

Κατά την εκτέλεσή της ο αλγόριθμος περιμένει τον χρήστη να δώσει τόσες τιμές χωριζόμενες με Enter όσες και οι μεταβλητές της λίστας_μεταβλητών και αποδίδει αντίστοιχα τις τιμές σε αυτές τις μεταβλητές.

Εξόδου: **Γράψε λίστα_στοιχείων** *(λίστα στοιχείων: 1 ή περισσότερες εκφράσεις χωριζόμενες με κόμμα)*

Κατά την εκτέλεσή της ο αλγόριθμος εμφανίζει στην οθόνη με την ίδια σειρά τις τιμές των εκφράσεων που περιέχονται στη λίστα_στοιχείων ύστερα από την αποτίμησή τους. Αντίστοιχες εντολές εξόδου, μόνο για την ψευδογλώσσα των αλγορίθμων και όχι για τη ΓΛΩΣΣΑ είναι και οι Εμφάνισε, Εκτύπωσε, Τύπωσε (οι 2 τελευταίες τυπώνουν σε εκτυπωτή)

Εκχώρησης τιμής: **Μεταβλητή←Έκφραση**

Κατά την εκτέλεσή της γίνεται αποτίμηση της Έκφρασης και το αποτέλεσμα εκχωρείται στη μεταβλητή. Προσοχή δεν πρόκειται για εξίσωση, αν και πολλές φορές σε κάποιες γλώσσες προγραμματισμού χρησιμοποιείται το σύμβολο "=" για την εντολή εκχώρησης. Σε μια εντολή εκχώρησης η μεταβλητή και η έκφραση πρέπει να είναι του ίδιου τύπου.

Οι παραπάνω εντολές είναι εκτελεστέες εντολές σε έναν αλγόριθμο ή πρόγραμμα. Εντολές που δεν εκτελούνται είτε σε αλγόριθμο είτε σε πρόγραμμα όπως: *Αλγόριθμος άσκηση, ΠΡΟΓΡΑΜΜΑ ΑΣΚΗΣΗ, ΜΕΤΑΒΛΗΤΕΣ, ΑΚΕΡΑΙΕΣ:μτβλ1,μτβλ2, ΑΡΧΗ κλπ* ονομάζονται δηλωτικές εντολές.

12. Τι είναι τα σχόλια σε έναν αλγόριθμο;

Τα σχόλια σε έναν αλγόριθμο είναι επεξηγηματικές φράσεις που βοηθούν στην κατανόηση του αλγορίθμου από κάποιον που τον μελετάει. Για να διαχωριστούν αυτές οι φράσεις, από τις λέξεις-κλειδιά του αλγορίθμου προτάσσεται το σύμβολο !. Παράδειγμα σχολίου: ! εδώ υπολογίζεται ο μεγαλύτερος

13. Τι αντιπροσωπεύουν οι μεταβλητές που εμπεριέχονται στις δηλωτικές εντολές Δεδομένα // // και Αποτελέσματα // //;

Οι μεταβλητές που εμπεριέχονται στις δηλωτικές εντολές Δεδομένα // // αντιπροσωπεύουν δεδομένα που έχουν εισαχθεί με κάποιο τρόπο πριν την έναρξη εκτέλεσης των εντολών του αλγορίθμου που ακολουθούν. Στην ουσία αν δεν υπήρχε η δηλωτική εντολή αυτή θα έπρεπε να υπάρχουν αντίστοιχες εντολές ***Διάβασε*** για να αποκτήσουν τιμές οι αντίστοιχες μεταβλητές.

Οι μεταβλητές που εμπεριέχονται στις δηλωτικές εντολές Αποτελέσματα // // αντιπροσωπεύουν τις τιμές δεδομένων (πληροφορίες) που αποτελούν την έξοδο του αλγορίθμου προς τον χρήστη ή προς κάποιον άλλο αλγόριθμο. Αν δεν υπήρχε αυτή η δηλωτική εντολή θα έπρεπε να υπάρχουν αντίστοιχες εντολές ***Γράψε*** για την εμφάνιση των τιμών των αντίστοιχων μεταβλητών προς το χρήστη.

§2.4.2 Δομή επιλογής

14. Τι γνωρίζετε για τη δομή της επιλογής;

Στις περιπτώσεις που σε έναν αλγόριθμο πρέπει να λαμβάνονται κάποιες αποφάσεις με βάση κάποια δεδομένα κριτήρια, που μπορεί να είναι διαφορετικά για κάθε στιγμιότυπο του προβλήματος που επιλύει ο αλγόριθμος, χρησιμοποιείται η δομή της επιλογής. Η διαδικασία της επιλογής περιλαμβάνει τον έλεγχο κάποιας συνθήκης (λογικής έκφρασης) που μπορεί να έχει δύο τιμές (Αληθής ή Ψευδής) και ακολουθεί η απόφαση εκτέλεσης κάποιας ενέργειας με βάση την τιμή της λογικής αυτής συνθήκης. Η γενική μορφή της δομής επιλογής– απλής, σύνθετης – αλλά και η διαγραμματική τους αναπαράσταση φαίνεται στον παρακάτω πίνακα.

§2.4.3 Διαδικασίες πολλαπλών επιλογών

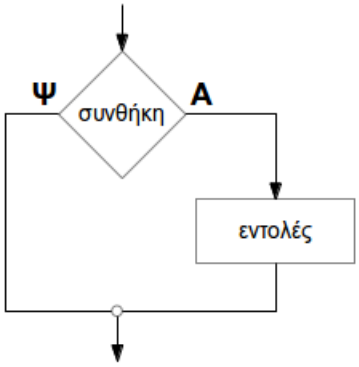
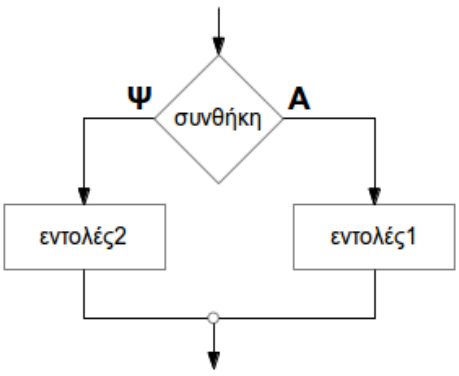
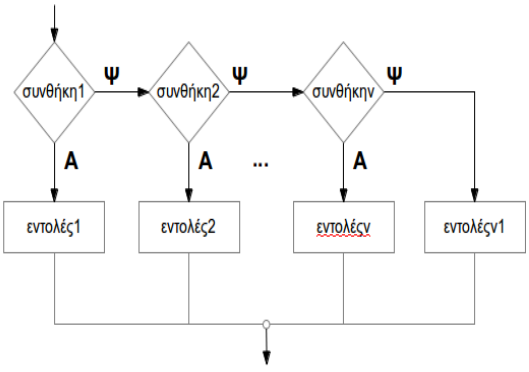
15. Τι γνωρίζετε για τη δομή της πολλαπλής επιλογής;

Σε προβλήματα που μπορούν να ληφθούν περισσότερες από 2 διαφορετικές αποφάσεις ανάλογα με την τιμή που παίρνει μια μεταβλητή ή μια έκφραση χρησιμοποιείται η δομή της πολλαπλής επιλογής. Η γενική μορφή της δομής πολλαπλής επιλογής και η διαγραμματικής της αναπαράσταση φαίνεται στον παρακάτω πίνακα.

Προσοχή: όταν διαπιστωθεί ως Αληθής μια από τις συνθήκες της πολλαπλής επιλογής, οι υπόλοιπες από εκεί και κάτω δεν εξετάζονται. Άρα, όταν εξετάζεται μια από τις συνθήκες της πολλαπλής επιλογής όλες οι προηγούμενες αυτής έχουν διαπιστωθεί ως ψευδείς....

Απλή επιλογή	Σύνθετη επιλογή	Πολλαπλή επιλογή

Αν συνθήκη τότε
  εντολές
Τέλος_αν
	Αν συνθήκη τότε
	  εντολές1
	αλλιώς
	  εντολές2
	Τέλος_αν
		Αν συνθήκη1 τότε
		  εντολές1
		αλλιώς_αν συνθήκη2 τότε
		  εντολές2
		....
		αλλιώς_αν συνθήκην τότε
		  εντολέςν
		αλλιώς
		  εντολέςν1
		Τέλος_αν

§2.4.4 Εμφωλευμένες διαδικασίες

16. Τι γνωρίζετε για τις εμφωλευμένες επιλογές;

Οι εμφωλευμένες επιλογές είναι ένας άλλος τρόπος να γραφούν οι πολλαπλές επιλογές περιλαμβάνοντας εντολές Αν μέσα σε άλλες εντολές Αν. Οι εμφωλευμένες επιλογές συνήθως οδηγούν σε πολύπλοκες δομές που αυξάνουν την πιθανότητα λάθους και τη δυσκολία κατανόησης του προγράμματος. Αντικατάστασή τους από σύνθετες λογικές εκφράσεις ή πολλαπλές επιλογές απλοποιούν τον κώδικα τις περισσότερες φορές.

17. Γράψτε τον πίνακα αληθείας για 2 λογικές προτάσεις Α και Β.

Πρόταση Α	Πρόταση Β	Α και Β	Α ή Β	όχι Α
Αληθής	Αληθής	Αληθής	Αληθής	Ψευδής
Αληθής	Ψευδής	Ψευδής	Αληθής	Ψευδής
Ψευδής	Αληθής	Ψευδής	Αληθής	Αληθής
Ψευδής	Ψευδής	Ψευδής	Ψευδής	Αληθής

§2.4.5 Δομή Επανάληψης

18. Τι γνωρίζετε για τη δομή της επανάληψης;

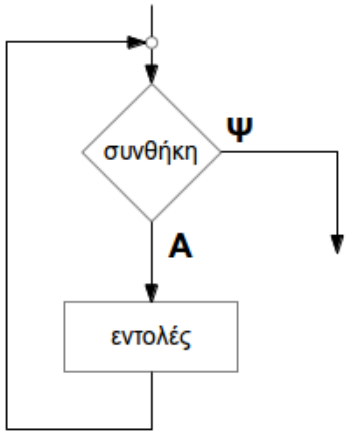
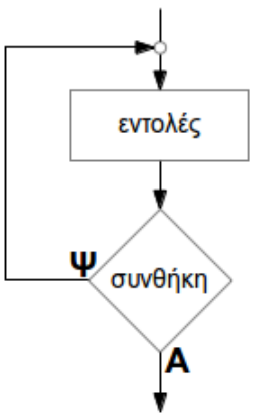
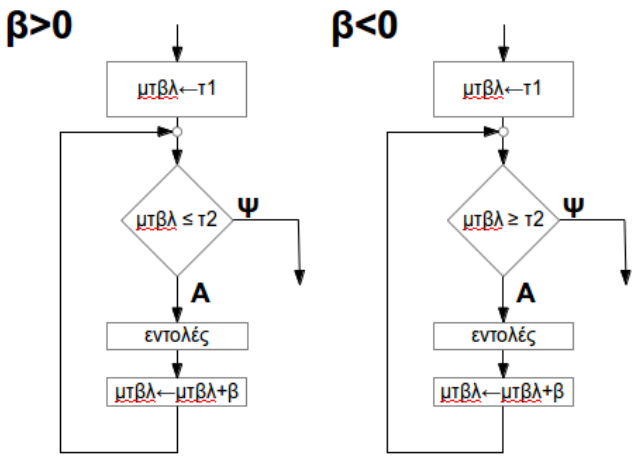
Η δομή της επανάληψης χρησιμοποιείται στις περιπτώσεις προβλημάτων όπου ακολουθίες εντολών εφαρμόζονται σε ένα σύνολο περιπτώσεων που έχουν κάτι κοινό. Οι δομές επανάληψης μπορεί να έχουν διάφορες μορφές και συνήθως εμπεριέχουν μέσα τους και δομές επιλογής κατά την επίλυση προβλημάτων. Οι γενικές μορφές της δομής επανάληψης και τα διαγράμματα ροής τους φαίνονται στον παρακάτω πίνακα.

19. Τι αποκαλείται βρόχος σε ένα αλγόριθμο:

Βρόχος είναι ένα τμήμα αλγορίθμου ή αλλιώς ένα σύνολο εντολών που επαναλαμβάνεται.

Όσο...επανάλαβε	Αρχή_επανάληψης ... Μέχρις_Όσου	Για ... από ... μέχρι ... με_βήμα ...

`Όσο συνθήκη επανάλαβε`
`  εντολές`
`Τέλος_επανάληψης`
	`Αρχή_επανάληψης`
	`  εντολές`
	`Μέχρις_όσου συνθήκη`
		`Για μτβλ από τ1 μέχρι τ2 με_βήμα β`
		`  εντολές`
		`Τέλος_επανάληψης`

20. Σε ποιες επαναληπτικές διαδικασίες είναι προτιμότερη η κάθε μορφή της δομής επανάληψης;

Όταν είναι γνωστός εκ των προτέρων ο αριθμός των φορών που θα εκτελεστεί μια επαναληπτική διαδικασία είναι προτιμότερο να χρησιμοποιηθεί η δομή επανάληψης **Για ... από ... μέχρι ... με_βήμα ...** . Μπορούν να χρησιμοποιηθούν και οι Όσο, Αρχή , αλλά η Για οδηγεί σε πιο συμπαγή μορφή αλγορίθμου και μειώνει τις πιθανότητες λάθους από απροσεξία, όπως παραλείψεις αρχικοποιήσεων και άθροισης βήματος. Σημειωτέον ότι αν παραληφθεί το **με_βήμα ....** τότε εννοείται ότι το βήμα είναι 1.

Όταν είναι άγνωστος ο αριθμός των επαναλήψεων (π.χ. όταν ο τερματισμός εκτέλεσης του βρόχου εξαρτάται από τιμές που δίνει ο χρήστης) αλλά ταυτόχρονα είναι δυνατόν η επαναληπτική διαδικασία να μην εκτελεστεί καμία φορά τότε είναι προτιμότερη η χρήση της **Όσο... επανάλαβε** . Μπορεί να χρησιμοποιηθεί και η Αρχή, αλλά απαιτείται επιπρόσθετος έλεγχος.

Όταν είναι άγνωστος ο αριθμός των επαναλήψεων αλλά είναι και σίγουρο ότι η επαναληπτική διαδικασία θα εκτελεστεί τουλάχιστο μία φορά τότε είναι προτιμότερη η χρήση της **Αρχή_Επανάληψης... Μέχρις_Ότου** . Χαρακτηριστικές τέτοιες περιπτώσεις είναι ο έλεγχος αποδεκτών τιμών καθώς και η επιλογή από προκαθορισμένες απαντήσεις ή μενού. Φυσικά και εδώ μπορεί να χρησιμοποιηθεί η ΟΣΟ.

21. Τι είναι η τιμή φρουρός σε μια επαναληπτική διαδικασία;

Τιμή φρουρός είναι μια σταθερά τιμή που ορίζει ο προγραμματιστής ότι όταν εισαχθεί από τον χρήστη σε ένα εκτελούμενο αλγόριθμο-πρόγραμμα θα τερματιστεί μια επαναληπτική διαδικασία. Η τιμή φρουρός επιλέγεται από τον προγραμματιστή έτσι ώστε να μην είναι λογικά σωστή για το πρόβλημα. Ο χρήστης πρέπει να ενημερώνεται από τον προγραμματιστή, με κάποιο μήνυμα, για το ποια είναι αυτή η τιμή για να καταφέρει να τερματίσει όταν το θέλει την επαναληπτική διαδικασία π.χ. μια επαναληπτική είσοδο στοιχείων.

Παράδειγμα1: κατά την επαναληπτική είσοδο ηλικιών για υπολογισμό μέσου όρου ηλικίας τιμή φρουρός θα μπορούσε να είναι η σταθερά -1 (αφού δεν υπάρχει τέτοια ηλικία).

Παράδειγμα2: κατά την επαναληπτική είσοδο πληθυσμών κρατών η τιμή φρουρός θα μπορούσε να είναι 10^10 αφού δεν υπάρχει περίπτωση να υπάρχει κράτος με πληθυσμό 10^10 αφού όλη η γη έχει πληθυσμό ~7*10^9

22. Τι τύπου μπορεί να είναι η μεταβλητή μτβλ και οι τιμές τ1, τ2, β στη δομή επανάληψης *"Για μτβλ από τ1 μέχρι τ2 με_βήμα β"*;

Η μεταβλητή μτβλ και οι τιμές τ1, τ2 και β σε αυτή τη μορφή δομής επανάληψης μπορεί να είναι είτε ακέραιες είτε πραγματικές για τη ΓΛΩΣΣΑ που διδάσκεται στο σχολικό βιβλίο.

23. Πώς εκτελείται η πράξη του πολλαπλασιασμού στα ηλεκτρονικά κυκλώματα του υπολογιστή; Περιγράψτε αναλυτικά και δώστε τον αλγόριθμο που την υλοποιεί.

Ο υπολογιστής στο εσωτερικό του υλοποιεί τον λεγόμενο πολλαπλασιασμό αλά ρωσικά για να πολλαπλασιάσει ακεραίους αριθμούς, θετικούς ή αρνητικούς. Ο πολλαπλασιασμός αλά ρωσικά απαιτεί πολλαπλασιασμό επί 2,

διαίρεση διά 2, πρόσθεση και σύγκριση, λειτουργίες οι οποίες μπορούν να υλοποιηθούν ταχύτατα σε επίπεδο κυκλωμάτων στον ηλεκτρονικό υπολογιστή. Αυτές οι λειτουργίες είναι από τις στοιχειώδεις λειτουργίες που είδαμε ότι μπορεί να εκτελεί ένας υπολογιστής στο κεφ. 1 του σχολικού βιβλίου. Αντίθετα η γνωστή μας διαδικασία χειρωνακτικού πολλαπλασιασμού, όπου απαιτείται πολλαπλασιασμός με οποιονδήποτε ακέραιο θα ήταν μια χρονοβόρα διαδικασία, δύσκολα υλοποιήσιμη σε επίπεδο ηλεκτρονικών κυκλωμάτων. Παρακάτω παρατίθεται ο αλγόριθμος που υλοποιεί τον πολλαπλασιασμό σε επίπεδο κυκλωμάτων.

```
Αλγόριθμος πολλαπλασιασμός_αλαρωσικα
Δεδομένα // Μ1, Μ2 //
Σ ← 0
Όσο Μ2 > 0 επανάλαβε
   Αν Μ2 mod 2 = 1 τότε
      Σ ← Σ + Μ1
   Τέλος_αν
   Μ1 ← Μ1*2
   Μ2 ← Μ2 div 2
Τέλος_επανάληψης
Αποτελέσματα // Σ //
Τέλος πολλαπλασιασμός_αλαρωσικα
```

Συγκρίσεις: από τις στοιχειώδεις λειτουργίες που μπορεί να εκτελεί ένας υπολογιστής. Μάλιστα η M2 mod 2=1 απαιτεί την εξέταση μόνο του πιο δεξιού bit ενός καταχωρητή.

Πρόσθεση: από τις στοιχειώδεις λειτουργίες που μπορεί να εκτελεί ένας υπολογιστής.

Λειτουργίες μεταφοράς δεδομένων:
Ολίσθηση αριστερά (shift left): όταν εφαρμόζεται σε ένα καταχωρητή, πολλαπλασιάζει το περιεχόμενό του επί 2.
Ολίσθηση δεξιά (shift right): όταν εφαρμόζεται σε ένα καταχωρητή, υλοποιεί την ακέραια διαίρεση του περιεχομένου του με το 2.

Παραδείγματα ολισθήσεων:

$17_{10} = 00010001_2$ → Ολίσθηση αριστερά → $00100010_2 = 34_{10}$ (ισοδυναμεί με πολλαπλασιασμό *2)

$27_{10} = 00011011_2$ → Ολίσθηση δεξιά → $00001101_2 = 13_{10}$ (ισοδυναμεί με div 2)

ΚΕΦΑΛΑΙΟ 3 – ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΑΛΓΟΡΙΘΜΟΙ.

§3.2 Αλγόριθμοι + Δομές Δεδομένων = προγράμματα

24. Τι είναι μια δομή δεδομένων;

Δομή Δεδομένων είναι ένα σύνολο συστηματικά και οργανωμένα αποθηκευμένων δεδομένων που υφίστανται επεξεργασία από ένα σύνολο λειτουργιών. Κάθε μορφή δομής δεδομένων αποτελείται από ένα σύνολο κόμβων (nodes).

25. Ποιες είναι οι βασικές λειτουργίες επί των δομών δεδομένων; Εφαρμόζονται όλες αυτές οι λειτουργίες σε κάθε δομή; Εξηγήστε, δίνοντας και ένα παράδειγμα.

- ✔ **Προσπέλαση** (access), πρόσβαση σε ένα κόμβο με σκοπό να εξετασθεί ή να τροποποιηθεί το περιεχόμενό του.
- ✔ **Εισαγωγή** (insertion), δηλαδή η προσθήκη νέων κόμβων σε μία υπάρχουσα δομή.
- ✔ **Διαγραφή** (deletion), που αποτελεί το αντίστροφο της εισαγωγής, δηλαδή ένας κόμβος αφαιρείται από μία δομή.
- ✔ **Αναζήτηση** (searching), κατά την οποία προσπελαύνονται οι κόμβοι μιας δομής, προκειμένου να εντοπιστούν ένας ή περισσότεροι που έχουν μια δεδομένη ιδιότητα.
- ✔ **Ταξινόμηση** (sorting), όπου οι κόμβοι μιας δομής διατάσσονται κατά αύξουσα ή φθίνουσα σειρά.
- ✔ **Αντιγραφή** (copying), κατά την οποία όλοι οι κόμβοι ή μερικοί από τους κόμβους μίας δομής αντιγράφονται σε μία άλλη δομή.
- ✔ **Συγχώνευση** (merging), κατά την οποία δύο ή περισσότερες δομές συνενώνονται σε μία ενιαία δομή.
- ✔ **Διαχωρισμός** (separation), που αποτελεί την αντίστροφη πράξη της συγχώνευσης.

Σπάνια χρησιμοποιούνται και οι οκτώ λειτουργίες σε μια δομή δεδομένων. Στην πράξη αυτό που συμβαίνει είναι μια δομή δεδομένων να είναι αποδοτικότερη από κάποια άλλη ως προς μια λειτουργία, αλλά λιγότερο αποδοτική ως προς κάποια άλλη. Η επιλογή της κατάλληλης εξαρτάται από τη συχνότητα με την οποία εκτελείται καθεμιά από αυτές τις λειτουργίες.

Παράδειγμα: Ας πούμε ότι ένας καθηγητής αφού διορθώσει τα διαγωνίσματα των μαθητών του, τους τα μοιράζει να τα δουν και τα ξαναμαζεύει. Κατά το μάζεμα μπορεί να τα βάζει ταχύτατα το ένα πάνω στο άλλο τυχαία (εισαγωγή σε λίστα). Θα μπορούσε να τα μαζεύει όμως και να τα τοποθετεί με αρκετά πιο αργό ρυθμό έτσι ώστε να είναι αλφαβητικά ταξινομημένα (εισαγωγή σε ταξινομημένο ευρετήριο). Το τι θα επιλεγεί εξαρτάται από το αν χρειάζεται ο καθηγητής να έχει πρόσβαση σε αυτά συχνά. Σκεφτείτε να έρχεται αμέσως μετά και κάθε 5 λεπτά ένας γονέας να ενημερωθεί για το παιδί του και να πρέπει να αναζητεί ο καθηγητής το γραπτό συγκεκριμένου μαθητή στην τυχαία λίστα με π.χ. 30 γραπτά. Άρα σε μια τέτοια περίπτωση θα ήταν

προτιμητέο κατά το μάζεμα να τοποθετούνται τα γραπτά σε ταξινομημένο ευρετήριο.

26. Ποια είναι η εξίσωση του Wirth και τι τονίζει;

Η εξίσωση του Wirth είναι η εξής: **Αλγόριθμοι + Δομές Δεδομένων = Προγράμματα**
Η εξίσωση αυτή τονίζει τη μεγάλη εξάρτηση μεταξύ των δομών δεδομένων και των αλγορίθμων που τις επεξεργάζονται. Στην ουσία κάθε πρόγραμμα εμπεριέχει αλγόριθμο/ους και δομή/ές δεδομένων ως αδιάσπαστη ενότητα.

27. Σε ποιες κατηγορίες διακρίνονται οι δομές δεδομένων; Ποιες οι διαφορές τους;

Οι δομές δεδομένων διακρίνονται σε δύο μεγάλες κατηγορίες: τις στατικές (static) και τις δυναμικές (dynamic).
Οι **δυναμικές δομές δεδομένων** δεν αποθηκεύονται σε συνεχόμενες θέσεις μνήμης αλλά στηρίζονται στην τεχνική της λεγόμενης δυναμικής παραχώρησης μνήμης (dynamic memory allocation). Οι δυναμικές δομές λοιπόν δεν έχουν σταθερό μέγεθος, αλλά ο αριθμός των κόμβων τους μεγαλώνει και μικραίνει καθώς στη δομή εισάγονται νέα δεδομένα ή διαγράφονται κάποια δεδομένα αντίστοιχα.
Οι **στατικές δομές δεδομένων** περιέχουν στοιχεία αποθηκευμένα σε συνεχόμενες θέσεις μνήμης. Το ακριβές μέγεθος της κύριας μνήμης που απαιτούν καθορίζεται κατά τη στιγμή του προγραμματισμού τους, και κατά συνέπεια κατά τη στιγμή της μετάφρασής τους και όχι κατά τη στιγμή της εκτέλεσης τους προγράμματος. Οι στατικές δομές υλοποιούνται με πίνακες που υποστηρίζονται από όλες τις γλώσσες προγραμματισμού.

§3.3 Πίνακες

28. Τι είναι πίνακας;

Πίνακας είναι μια δομή που περιέχει ένα σύνολο αντικειμένων του ίδιου τύπου (δηλαδή ακέραιους, πραγματικούς, χαρακτήρες κλπ) τα οποία αναφέρονται με ένα κοινό όνομα. Κάθε ένα από τα αντικείμενα που απαρτίζουν τον πίνακα λέγεται στοιχείο του πίνακα. Η δήλωση των στοιχείων ενός πίνακα και η μέθοδος αναφοράς τους εξαρτάται από τη συγκεκριμένη γλώσσα υψηλού επιπέδου που χρησιμοποιείται. Πάντως η αναφορά σε ατομικά στοιχεία του πίνακα γίνεται με τη χρήση του συμβολικού ονόματος του πίνακα ακολουθούμενου από την τιμή ενός (για μονοδιάστατους πίνακες), δύο (για δισδιάστατους) ή ν δεικτών-indexes (για ν-διάστατους) σε παρένθεση ή αγκύλη. Όσον αφορά στους δισδιάστατους πίνακες σημειώνεται ότι αν το μέγεθος των δύο διαστάσεων είναι ίσο, τότε ο πίνακας λέγεται τετραγωνικός (square) και γενικά συμβολίζεται ως πίνακας n x n.

§3.6 Αναζήτηση

29. Σε ποιες περιπτώσεις δικαιολογείται η χρήση της μεθόδου σειριακής αναζήτησης;

Η μέθοδος της σειριακής αναζήτησης είναι η απλούστερη αλλά και λιγότερο αποτελεσματική μέθοδος αναζήτησης. Η χρήση της δικαιολογείται μόνο στις περιπτώσεις όπου

- ✔ ο πίνακας είναι μη ταξινομημένος
- ✔ ο πίνακας είναι μικρού μεγέθους
- ✔ η αναζήτηση γίνεται σπάνια

30. Περιγράψτε τις παραλλαγές της σειριακής αναζήτησης στις περιπτώσεις:

α) μη ταξινομημένου πίνακα με μοναδικά στοιχεία

β) μη ταξινομημένου πίνακα με μη μοναδικά στοιχεία.

γ) ταξινομημένου πίνακα με μη μοναδικά στοιχεία.

α) Στην περίπτωση που κάθε στοιχείο εμφανίζεται μόνο μια φορά στον πίνακα αναζήτησης τότε εφόσον κατά την προσπέλαση εντοπιστεί το ζητούμενο στοιχείο η επαναληπτική διαδικασία της αναζήτησης μπορεί να τερματιστεί. Αυτό επιτυγχάνεται με τη βοήθεια μιας λογικής μεταβλητής (π.χ. βρέθηκε) η οποία αρχικοποιείται κατάλληλα, τίθεται ως Αληθής κατά τον εντοπισμό και ελέγχεται στη συνθήκη της επανάληψης.

```
Αλγόριθμος σειριακή_αναζήτηση
Δεδομένα // Τ, n, key //
θ ← 0
βρεθηκε ← Ψευδής
i ← 1
Όσο i ≤ n και βρεθηκε = Ψευδής επανάλαβε
  Αν Τ[i] = key τότε
    βρεθηκε ← Αληθής
    θ ← i
  αλλιώς
    i ← i + 1
  Τέλος_αν
Τέλος_επανάληψης
Αποτελέσματα // βρεθηκε, θ //
Τέλος σειριακή_αναζήτηση
```

β) Στην περίπτωση που κάποια στοιχεία μπορεί να εμφανίζονται περισσότερες από μία φορές στον πίνακα αναζήτησης τότε η επαναληπτική διαδικασία της αναζήτησης θα συνεχιστεί για όλες τις θέσεις του πίνακα και δεν απαιτείται λογική μεταβλητή για τον τερματισμό της επανάληψης παρά μόνο ο έλεγχος της συνθήκης i<=n που μπορεί να γίνει μέσω της δομής Για.

```
Αλγόριθμος σειριακή_αναζήτηση2
Δεδομένα // Τ, n, key //
Βρέθηκε ← Ψευδής
Για i από 1 μέχρι n
  Αν Τ[i] = key τότε
    Βρέθηκε ← Αληθής
    Γράψε 'Βρέθηκε στη θέση ', i
  Τέλος_αν
Τέλος_επανάληψης
Αν Βρέθηκε = Ψευδής τότε
  Γράψε 'Δεν υπάρχει το στοιχείο ', key
Τέλος_αν
Τέλος σειριακή_αναζήτηση2
```

γ) στην περίπτωση ταξινομημένου πίνακα, αν κατά τη σειριακή αναζήτηση εντοπίσουμε μία ή περισσότερες φορές το προς αναζήτηση στοιχείο το εμφανίζουμε. Αν όμως προσπελάσουμε στοιχείο που είναι μεγαλύτερο από το αναζητούμενο, η αναζήτηση πρέπει να τερματιστεί. Για το σκοπό αυτό πρέπει να χρησιμοποιηθεί σίγουρα μια λογική μεταβλητή (π.χ. ΒρέθηκεΜεγαλύτερο) η οποία να αρχικοποιείται και να τίθεται ανάλογα αν και όταν βρεθεί μεγαλύτερο στοιχείο από το αναζητούμενο. Μπορεί να χρησιμοποιηθεί και άλλη λογική μεταβλητή (π.χ. Βρέθηκε) για να κρατιέται η πληροφορία αν βρέθηκε το προς αναζήτηση στοιχείο.

```
Αλγόριθμος σειριακή_αναζήτηση3
Δεδομένα // T, n, key //
Βρέθηκε ← Ψευδής
ΒρέθηκεΜεγαλύτερο ← Ψευδής
i ← 1
Όσο i ≤ n και ΒρέθηκεΜεγαλύτερο = Ψευδής επανάλαβε
  Αν T[i] = key τότε
    Γράψε 'Βρέθηκε στη θέση ', i
    Βρέθηκε ← Αληθής
  αλλιώς_αν T[i] > key τότε
    ΒρέθηκεΜεγαλύτερο ← Αληθής
  Τέλος_αν
  i ← i + 1
Τέλος_επανάληψης
Αν Βρέθηκε = Ψευδής τότε
  Γράψε 'Δεν υπάρχει το στοιχείο ', key
Τέλος_αν
Τέλος σειριακή_αναζήτηση3
```

31. Ποια μέθοδος αναζήτησης ενδείκνυται στις περιπτώσεις που ο πίνακας είναι ταξινομημένος;

Η μέθοδος της σειριακής αναζήτησης είναι η απλούστερη αλλά και λιγότερο αποτελεσματική μέθοδος αναζήτησης. Η αποτελεσματικότερη μέθοδος που μειώνει κατά πολύ το χρόνο αναζήτησης στους ταξινομημένους πίνακες είναι η μέθοδος της **δυαδικής αναζήτησης**, η οποία εξετάζει το μεσαίο στοιχείο του πίνακα και αν αυτό είναι το αναζητούμενο σταματάει, αλλιώς αν είναι μικρότερο από το αναζητούμενο τότε επαναλαμβάνει την αναζήτηση στο δεξιό μισό του πίνακα αλλιώς αν είναι μεγαλύτερο τότε επαναλαμβάνει την αναζήτηση στο αριστερό μισό του πίνακα.

§3.7 Ταξινόμηση

32. Τι γνωρίζετε για τη λειτουργία της ταξινόμησης; Που χρησιμεύει; Δώστε ένα ορισμό αυτής.

Η ταξινόμηση ή διάταξη είναι η λειτουργία της τακτοποίηση των κόμβων μιας δομής δεδομένων με μια ιδιαίτερη σειρά (π.χ. αύξουσα ή φθίνουσα). Η ταξινόμηση γίνεται για τη διευκόλυνση της διαδικασίας της αναζήτησης σε μια δομή δεδομένων. Η χρησιμότητά της αποδεικνύεται σε αναρίθμητες περιπτώσεις αναζήτησης αριθμητικών ή αλφαβητικών δεδομένων σε λεξικά, τηλ. καταλόγους, βιβλιοθηκονομικά συστήματα κλπ. Ακολουθεί ένας τυπικός ορισμός της ταξινόμησης.

Ορισμός. *Δοθέντων των στοιχείων* $a_1a_2...,a_n$ *η ταξινόμηση συνίσταται στη μετάθεση (permutation) της θέσης των στοιχείων, ώστε να τοποθετηθούν σε μία σειρά* $a_{k1},a_{k2},...,a_{kn}$ *έτσι ώστε, δοθείσης μίας συνάρτησης διάταξης (ordering function), f, να ισχύει:* $f(a_{k1})<f(a_{k2})<...<f(a_{kn})$

Σημ. Η συνάρτηση διάταξης στην περίπτωση της φθίνουσας ταξινόμησης είναι η f(x)= -x !

33. Περιγράψτε τη διαδικασία της ταξινόμησης ευθείας ανταλλαγής. Πώς αλλιώς ονομάζεται και γιατί; Δώστε έναν αλγόριθμο για την υλοποίηση αυτής.

Η ταξινόμηση ευθείας ανταλλαγής στηρίζεται σε διαδοχικές σαρώσεις του προς ταξινόμηση πίνακα, κατά τις οποίες συγκρίνονται ζευγάρια γειτονικών στοιχείων, αρχίζοντας από το ζευγάρι τελευταίο-προτελευταίο, και αντιμεταθέτοντάς τα εφόσον δεν ακολουθούν τη ζητούμενη διάταξη. Με αυτό τον τρόπο όταν ολοκληρωθεί η 1η σάρωση του πίνακα το πιο μικρό στοιχείο (εφόσον πρόκειται για αύξουσα ταξινόμηση) θα τοποθετηθεί σωστά στην 1η θέση του πίνακα. Με τη δεύτερη σάρωση του πίνακα τοποθετείται το 2ο πιο μικρό στοιχείο στη 2η θέση του πίνακα κ.ο.κ. μέχρι να τοποθετηθούν όλα στη σωστή τους θέση.

```
Αλγόριθμος φυσσαλίδα
Δεδομένα // Τ, n //
Για i από 2 μέχρι n
  Για j από n μέχρι i με_βήμα -1
    Αν Τ[j - 1] > Τ[j] τότε
      temp ← Τ[j - 1]
      Τ[j - 1] ← Τ[j]
      Τ[j] ← temp
    Τέλος_αν
  Τέλος_επανάληψης
Τέλος_επανάληψης
Αποτελέσματα // Τ //
Τέλος φυσσαλίδα
```

Κατά τη διάρκεια των σαρώσεων και εφόσον δούμε τον πίνακα σαν πίνακα-στήλη, μπορούμε να παρατηρήσουμε στοιχεία να μετακινούνται προς τα πάνω ως φυσσαλίδες σε υγρό που ανεβαίνουν ώσπου να συναντήσουν πιο ελαφριές φυσσαλίδες οι οποίες με τη σειρά τους ανεβαίνουν πιο πάνω κ.ο.κ. Για το λόγο αυτό ο αλγόριθμος αυτός ονομάζεται και αλγόριθμος φυσσαλίδα (bubblesort). Υλοποίηση του αλγορίθμου παρατίθεται στο διπλανό σχήμα.:

34. Σε τι είδους δεδομένα μπορεί να εφαρμοστεί η ταξινόμηση φυσσαλίδας;

Μπορεί να εφαρμοστεί σε αριθμητικά ή αλφαριθμητικά δεδομένα (χαρακτήρες). Όχι σε λογικές τιμές οι οποίες δεν μπορούν να συγκριθούν!

35. α)Πώς εφαρμόζεται στον αλγόριθμο του σχήματος η ταξινόμηση βάσει συνάρτησης διάταξης f ; β)Πώς αλλάζει η συνθήκη στην περίπτωση που θέλουμε φθίνουσα ταξινόμηση στοιχείων;

α) Εφαρμόζεται απλά μετασχηματίζοντας την συνθήκη Αν του αλγορίθμου φυσσαλίδας σε: `Αν f(T[j-1])>f(T[j]) τότε`. β) Στην περίπτωση που θέλουμε φθίνουσα ταξινόμηση αυτό ουσιαστικά οδηγεί στη συνθήκη *`Αν T[j-1]) < T[j])`* (αλλαγή της φοράς της ανισότητας στη συνθήκη).

36. Ποιοι αλγόριθμοι έχετε ακούσει ότι έχουν εκπονηθεί για την ταξινόμηση δεδομένων;

Για την ταξινόμηση δεδομένων έχουν εκπονηθεί πάρα πολλοί αλγόριθμοι. Σχετικά απλοί είναι οι ταξινόμηση με επιλογή ελαχίστου και ταξινόμηση με παρεμβολή ενώ ο πιο γρήγορος αλγόριθμος ταξ/σης είναι η quicksort. Η ταξινόμηση φυσσαλίδας είναι ο πιο απλός και ταυτόχρονα ο πιο αργός αλγόριθμος ταξινόμησης.

ΚΕΦΑΛΑΙΟ 6 – ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ.

§6.3 Φυσικές και τεχνητές γλώσσες

37. Ποιες γλώσσες αναφέρονται ως φυσικές και ποιες ως τεχνητές;.

Φυσικές είναι οι γλώσσες που χρησιμοποιούνται για την επικοινωνία μεταξύ ανθρώπων, ενώ τεχνητές είναι οι γλώσσες που χρησιμοποιούνται για την επικοινωνία ανθρώπου-υπολογιστή. Οι γλώσσες προγραμματισμού είναι τεχνητές γλώσσες που ακολουθούν τις βασικές έννοιες και αρχές της γλωσσολογίας αλλά δεν εξελίσσονται.

38. Ποια χαρακτηριστικά προσδιορίζουν μια γλώσσα φυσική ή τεχνητή;

Μία γλώσσα προσδιορίζεται από το αλφάβητο της, το λεξιλόγιο της, τη γραμματική της και τέλος τη σημασιολογία της.

- **Αλφάβητο** μίας γλώσσας καλείται το σύνολο των στοιχείων που χρησιμοποιείται από τη γλώσσα, γράμματα πεζά κεφαλαία, ψηφία και σημεία στίξης.
- **Λεξιλόγιο** είναι το υποσύνολο των αποδεκτών ακολουθιών που δημιουργούνται από τα στοιχεία του αλφαβήτου, δηλαδή οι αποδεκτές λέξεις της γλώσσας.
- **Γραμματική,** η οποία αποτελείται από:
    - Το **τυπικό,** δηλαδή των σύνολο των κανόνων που ορίζει με ποιες μορφές είναι μια λέξη αποδεκτή από τη γλώσσα. Π.χ. Από τη λέξη γλώσσα παράγονται οι λέξεις γλώσσας, γλώσσες, γλωσσών όχι όμως και η λέξη ~~γλώσσον~~
    - Το **συντακτικό,** δηλαδή το σύνολο των κανόνων που ορίζει τη νομιμότητα της διάταξης και σύνδεσης των λέξεων της γλώσσας για τη δημιουργία σωστών προτάσεων. π.χ. “Είμαι πολύ καλά σήμερα δεν” δεν είναι συντακτικά σωστή πρόταση
- **Σημασιολογία,** είναι το σύνολο των κανόνων που καθορίζει το νόημα των λέξεων και κατ' επέκταση των προτάσεων και εκφράσεων. Στις τεχνητές γλώσσες τη σημασιολογία την καθορίζει ο δημιουργός της γλώσσας προγραμματισμού.

39. Ποιες οι διαφορές φυσικών και τεχνητών γλωσσών ;

Η βασική διαφορά μεταξύ φυσικών και τεχνητών γλωσσών είναι η δυνατότητα εξέλιξής τους.

Οι φυσικές γλώσσες εξελίσσονται διαρκώς, εμφανίζονται νέες λέξεις, αλλάζουν κανόνες γραμματικής και σύνταξης κι αυτά επειδή οι άνθρωποι εξελίσσονται, οι εποχές και ο κοινωνικός περίγυρος αλλάζει.

Οι τεχνητές γλώσσες, χαρακτηρίζονται από στασιμότητα αφού κατασκευάζονται συνειδητά για συγκεκριμένο σκοπό. Ωστόσο και οι γλώσσες προγραμματισμού συχνά βελτιώνονται και μεταβάλλονται από τους δημιουργούς τους για να διορθώσουν αδυναμίες, να καλύψουν μεγαλύτερο εύρος εφαρμογών και να ακολουθήσουν τις εξελίξεις.

§6.4 Τεχνικές σχεδίασης προγραμμάτων

§6.4.1 Ιεραρχική σχεδίαση προγράμματος

40. Τι γνωρίζετε για την ιεραρχική σχεδίαση προγράμματος;

Η **ιεραρχική σχεδίαση ή "από επάνω προς τα κάτω προγραμματισμός"** περιλαμβάνει τον καθορισμό των βασικών λειτουργιών ενός προγράμματος, σε ανώτερο επίπεδο, 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 λοιπόν του προβλήματος σε μια σειρά από απλούστερα υποπροβλήματα, τα οποία να είναι εύκολο να επιλυθούν οδηγώντας στην επίλυση του αρχικού προβλήματος.

Η παράσταση της ιεραρχικής σχεδίασης γίνεται με διάφορες διαγραμματικές τεχνικές.

§6.4.2 Τμηματικός προγραμματισμός

41. Τι γνωρίζετε για τον τμηματικό προγραμματισμό;

Τμηματικός προγραμματισμός ονομάζεται η τεχνική σχεδίασης και ανάπτυξης των προγραμμάτων ως ένα σύνολο από απλούστερα τμήματα προγραμμάτων. Ο τμηματικός προγραμματισμόςυλοποιεί την ιεραρχική σχεδίαση προγράμματος. Το πρόβλημα αναλύεται σε επιμέρους υποπροβλήματα (συνεχώς μέχρι να γίνουν πλύ απλά) καθένα από τα οποία αποτελεί ανεξάρτητη ενότητα που γράφεται ξεχωριστά από τα υπόλοιπα τμήματα προγράμματος. Ο τμηματικός προγραμματισμός διευκολύνει τη δημιουργία του προγράμματος, μειώνει τα λάθη και επιτρέπει την ευκολότερη παρακολούθηση, κατανόηση και συντήρησή του από τρίτους.

§6.4.3 Δομημένος προγραμματισμός

42. Τι γνωρίζετε για το δομημένο προγραμματισμό;

Ο **δομημένος προγραμματισμός** είναι μια μεθοδολογία σύνταξης προγραμμάτων που που έχει σκοπό να βοηθήσει τον προγραμματιστή στην ανάπτυξη σύνθετων προγραμμάτων, να μειώσει τα λάθη, να εξασφαλίσει την εύκολη κατανόηση των προγραμμάτων και να διευκολύνει τις διορθώσεις και τις αλλαγές σε αυτά.

Ο δομημένος προγραμματισμός στηρίζεται στη χρήση τριών και μόνο στοιχειωδών λογικών δομών, τη δομή της ακολουθίας, τη δομή της επιλογής και τη δομή της επανάληψης. Όλα τα προγράμματα μπορούν να γραφούν χρησιμοποιώντας μόνο αυτές τις τρεις δομές καθώς και συνδυασμό τους. Κάθε πρόγραμμα όπως και κάθε ενότητα προγράμματος έχει μόνο μία είσοδο και μόνο μία έξοδο. Αυτό αντίθετα δε συνέβαινε στην αρχική ιστορία του προγραμματισμού όπου η υπερβολική χρήση μιας εντολής, της εντολής GOTO, η οποία άλλαζε τη ροή ενός προγράμματος, δημιουργούσε δυσκολίες στην αρχική σχεδίαση της λύσης, αλλά κυρίως στην παρακολούθηση, κατανόηση και συντήρηση ενός προγράμματος.

Ο δομημένος προγραμματισμός ενθαρρύνει και βοηθάει την ανάλυση του προγράμματος σε επί μέρους

τμήματα, έτσι ώστε σήμερα ο όρος δομημένος προγραμματισμός περιέχει τόσο την ιεραρχική σχεδίαση όσο και τον τμηματικό προγραμματισμό.

43. Ποια είναι τα πλεονεκτήματα του 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.

§6.7 Προγραμματιστικά περιβάλλοντα

44. Πώς γίνεται η μετατροπή ενός προγράμματος από γλώσσα υψηλού επιπέδου σε γλώσσα μηχανής;

Η μετατροπή ενός προγράμματος από γλώσσα υψηλού επιπέδου σε εντολές γλώσσας μηχανής (κατανοητές από τον υπολογιστή) επιτυγχάνεται με τη χρήση ειδικών μεταφραστικών προγραμμάτων που κατατάσσονται σε δύο μεγάλες κατηγορίες, στους **μεταγλωττιστές (compilers)** και τους **διερμηνευτές (interpreters)**.

45. Τι γνωρίζετε για τους μεταγλωττιστές και τους διερμηνευτές; Ποιες οι διαφορές τους;

Οι μεταγλωττιστές και οι διερμηνευτές ανήκουν στα ειδικά μεταφραστικά προγράμματα που μετατρέπουν προγράμματα από γλώσσα υψηλού επιπέδου σε γλώσσα μηχανής.

Μεταγλωττιστής: Δέχεται στην είσοδο ένα πρόγραμμα γραμμένο σε μια γλώσσα υψηλού επιπέδου και παράγει ένα ισοδύναμο πρόγραμμα σε γλώσσα μηχανής. Το τελευταίο μπορεί να εκτελείται οποτεδήποτε από τον υπολογιστή και είναι τελείως ανεξάρτητο από το αρχικό πρόγραμμα.

Διερμηνευτής: Διαβάζει μία προς μία τις εντολές του αρχικού προγράμματος και για κάθε μια εκτελεί αμέσως μια ισοδύναμη ακολουθία εντολών μηχανής.

Η χρήση μεταγλωττιστή έχει το μειονέκτημα, ότι προτού χρησιμοποιηθεί ένα πρόγραμμα, πρέπει να περάσει από τη διαδικασία της μεταγλώττισης και σύνδεσης δηλαδή να διορθωθούν όλα τα συντακτικά λάθη. Από την άλλη μεριά η χρήση διερμηνευτή έχει το πλεονέκτημα της άμεσης εκτέλεσης και συνεπώς και της άμεσης διόρθωσης.

Με το διερμηνευτή όμως η εκτέλεση του προγράμματος καθίσταται πιο αργή, σημαντικά μερικές φορές, από εκείνη του ισοδύναμου εκτελέσιμου προγράμματος που παράγει ο μεταγλωττιστής.

Τα σύγχρονα προγραμματιστικά περιβάλλοντα παρουσιάζονται συνήθως με μεικτές υλοποιήσεις, όπου χρησιμοποιείται διερμηνευτής κατά τη φάση δημιουργίας του προγράμματος και μεταγλωττιστής για την τελική έκδοση και εκμετάλλευση του προγράμματος.

46. Ποιο πρόγραμμα ονομάζεται πηγαίο και ποιο αντικείμενο;

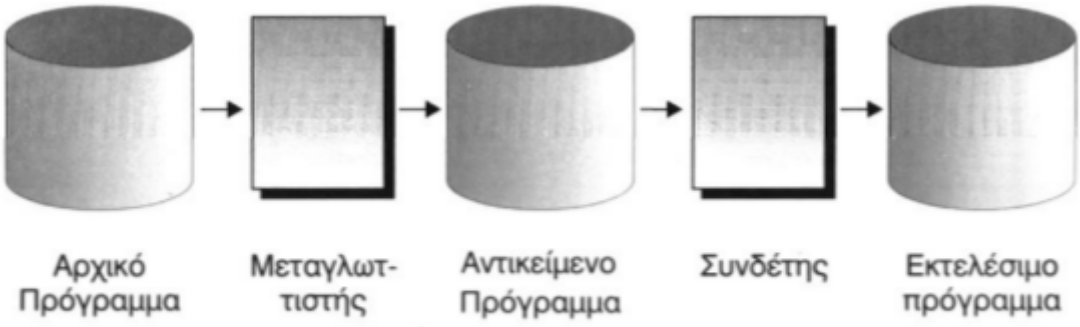
Το αρχικό πρόγραμμα που γράφεται από τον προγραμματιστή (σε γλώσσα υψηλού επιπέδου συνήθως) ονομάζεται πηγαίο πρόγραμμα.

Το πρόγραμμα που παράγεται από τον μεταγλωττιστή ονομάζεται αντικείμενο πρόγραμμα (object code). Είναι σε μορφή κατανοητή από τον υπολογιστή αλλά δεν είναι σε θέση να εκτελεστεί πριν συνδεθεί με άλλα τμήματα προγράμματος που είναι απαραίτητα για την εκτέλεσή του τα οποία υπάρχουν στις βιβλιοθήκες της γλώσσας ή τα γράφει ο προγραμματιστής.

47. Τι είναι ο συνδέτης-φορτωτής (linker) και ποιος ο ρόλος του;

Ο συνδέτης-φορτωτής είναι το ειδικό εκείνο πρόγραμμα που αναλαμβάνει τη σύνδεση του αντικειμένου προγράμματος που παράγεται από το μεταγλωττιστή με άλλα τμήματα προγράμματος που είναι απαραίτητα για την εκτέλεσή του τα οποία υπάρχουν στις **βιβλιοθήκες (libraries)** της γλώσσας ή τα γράφει ο προγραμματιστής. Αποτέλεσμα του συνδέτη είναι η παραγωγή του **εκτελέσιμου προγράμματος (executable)** το οποίο είναι το τελικό πρόγραμμα που εκτελείται από τον υπολογιστή. Η συνολική διαδικασία παραγωγής ενός εκτελέσιμου προγράμματος ονομάζεται μεταγλώττιση και σύνδεση.

48. Περιγράψετε τη διαδικασία για τη μετάφραση και εκτέλεση ενός προγράμματος.



Μεταγλώττιση και σύνδεση προγράμματος

49. Ποια λάθη μπορεί να εμφανίζονται σε ένα πρόγραμμα και με ποιον τρόπο ανιχνεύονται;

Σε ένα πρόγραμμα μπορούν να εμφανιστούν γενικά δύο είδη λαθών, τα συντακτικά και τα λογικά.

Τα **συντακτικά λάθη** εμφανίζονται κατά το στάδιο της μεταγλώττισης και ανιχνεύονται από το μεταγλωττιστή ή το διερμηνευτή εμφανίζοντας κατάλληλα διαγνωστικά μηνύματα. Οφείλονται σε αναγραμματισμούς ονομάτων εντολών-μεταβλητών, παραλήψεις δηλώσεων κλπ. και πρέπει να εξαλειφθούν πλήρως ώστε να παραχθεί το τελικό εκτελέσιμο πρόγραμμα.

Τα **λογικά λάθη** είναι τα πλέον σοβαρά και δύσκολα στην ανίχνευση και διόρθωσή τους και οφείλονται σε σφάλματα κατά την υλοποίηση του αλγορίθμου. Τα προγράμματα που τα έχουν δεν βγάζουν τα αναμενόμενα αποτελέσματα για το πρόγραμμα που υποτίθεται ότι επιλύουν.

50. Τι είναι ο συντάκτης προγραμμάτων;.

Ο συντάκτης προγραμμάτων είναι ένας μικρός ειδικός επεξεργαστής κειμένου που χρησιμοποιείται για την αρχική σύνταξη και διόρθωση των προγραμμάτων μιας γλώσσα προγραμματισμού. Ο συντάκτης μιας γλώσσας έχει ορισμένες δυνατότητες που διευκολύνουν τη γρήγορη γραφή των εντολών των προγραμμάτων σε αυτή τη γλώσσα.

51. Ποια προγράμματα και εργαλεία περιέχει ένα σύγχρονο ολοκληρωμένο προγραμματιστικό περιβάλλον;.

Τα σύγχρονα ολοκληρωμένα προγραμματιστικά περιβάλλοντα περιέχουν όλα τα προγράμματα και εργαλεία που απαιτούνται και βοηθούν τη συγγραφή, την εκτέλεση και κύρια τη διόρθωση των προγραμμάτων. Πρέπει να περιέχουν απαραίτητα

- Συντάκτη
- Μεταγλωττιστή και
- Συνδέτη-Φορτωτή

Το κάθε προγραμματιστικό περιβάλλον έχει φυσικά διαφορετικά εργαλεία και ιδιότητες. Για παράδειγμα ένα περιβάλλον οπτικού (visual) προγραμματισμού πρέπει να περιέχει οπωσδήποτε και ειδικό συντάκτη που να διευκολύνει τη δημιουργία γραφικών αντικειμένων (για παράδειγμα φόρμες, λίστες, παράθυρα διαλόγου) παρέχοντας στον προγραμματιστή τα αντίστοιχα γραφικά εργαλεία.

ΚΕΦΑΛΑΙΟ 7 – ΒΑΣΙΚΑ ΣΤΟΙΧΕΙΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ.

§7.1 Το αλφάβητο της ΓΛΩΣΣΑΣ

52. Τι περιλαμβάνει το αλφάβητο της ΓΛΩΣΣΑΣ;

Γράμματα ελληνικά - λατινικά, κεφαλαία και πεζά, ψηφία 0-9, ειδικούς χαρακτήρες + - * / ^ _ < <= > >= = <> ( ) & ! ' " . ,

§7.2 Τύποι δεδομένων

53. Ποιοι είναι οι τύποι δεδομένων της ΓΛΩΣΣΑΣ;

Ακέραιος (περιλαμβάνει τους γνωστούς από τα μαθηματικά ακεραίους), **Πραγματικός** (περιλαμβάνει τους πραγματικούς αριθμούς), **Χαρακτήρας ή αλφαριθμητικός** (περιλαμβάνει οποιαδήποτε ακολουθία χαρακτήρων μέσα σε ' ' , **Λογικός** (περιλαμβάνει μόνο τις τιμές ΑΛΗΘΗΣ και ΨΕΥΔΗΣ).

§7.3 Σταθερές

54. Τι είναι οι συμβολικές σταθερές στη ΓΛΩΣΣΑ;

Οι συμβολικές σταθερές είναι σταθερές τιμές στις οποίες έχουν αποδοθεί ονόματα και οι δηλώσεις τους γίνονται στο τμήμα δήλωσης σταθερών πριν από το τμήμα δήλωσης μεταβλητών και μετά τη δήλωση του ονόματος του προγράμματος. Οι συμβολικές σταθερές μπορούν να χρησιμοποιηθούν σε εκφράσεις οπουδήποτε στο πρόγραμμα, αλλά δεν μπορεί να μεταβληθεί η τιμή τους κατά τη διάρκεια της εκτέλεσής του. Η χρήση τους κάνει το πρόγραμμα πιο κατανοητό και ευκολότερο να διορθωθεί και να συντηρηθεί. Παράδειγμα δήλωσης σταθερών:

ΣΤΑΘΕΡΕΣ

π=3.14

§7.4 Μεταβλητές

55. Τι είναι οι μεταβλητές στη ΓΛΩΣΣΑ; (βλ. ερ. 8)

56. Που δηλώνονται οι μεταβλητές στη ΓΛΩΣΣΑ;

Δηλώνονται πριν τη δεσμευμένη λέξη **ΑΡΧΗ** και μετά τη δήλωση σταθερών. Το τμήμα δηλώσεων μεταβλητών ξεκινάει με τη δεσμευμένη λέξη **ΜΕΤΑΒΛΗΤΕΣ** και στην επόμενη γραμμή γράφεται ο τύπος που δηλώνεται, : , και τα ονόματα μεταβλητών αυτού του τύπου χωριζόμενα με κόμμα. Στην επόμενη γραμμή ο επόμενος τύπος που εμφανίζεται στο πρόγραμμα και τα ονόματα μεταβλητών ομοίως .

Π.χ. ***ΜΕΤΑΒΛΗΤΕΣ***

ΠΡΑΓΜΑΤΙΚΕΣ: Μέσος_Όρος

ΑΚΕΡΑΙΕΣ: i, πλήθος

§7.5 Αριθμητικοί τελεστές

57. Τι γνωρίζετε για τους τελεστές της ΓΛΩΣΣΑΣ; (βλ. ερ. 9)

§7.6 Συναρτήσεις

58. Ποιες γνωστές συναρτήσεις των μαθηματικών χρησιμοποιούνται στη ΓΛΩΣΣΑ;

Χρησιμοποιούνται οι: **Α_Τ(Χ)** (απόλυτη τιμή) , **Τ_Ρ(Χ)** (τετραγωνική ρίζα), **Α_Μ(Χ)** (ακέραιο μέρος), **ΗΜ(Χ)** (ημίτονο), **ΣΥΝ(Χ)** (συνημίτονο), **ΕΦ(Χ)** (εφαπτομένη), **Ε(Χ)** (το e^x) και **ΛΟΓ(Χ)** (το lnx)

§7.7 Εκφράσεις

59. Τι γνωρίζετε για τις εκφράσεις της ΓΛΩΣΣΑΣ;

(βλ. ερ. 10)

60. Ποια η ιεραρχία των πράξεων στις αριθμητικές εκφράσεις της ΓΛΩΣΣΑΣ;

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

Όταν η ιεραρχία είναι ίδια (όπως πολλαπλασιασμός-διαίρεση-mod-div, πρόσθεση-αφαίρεση) οι πράξεις εκτελούνται από αριστερά προς τα δεξιά. Πράξεις χαμηλότερης ιεραρχίας αποκτούν μεγαλύτερη όταν μπουν σε παρενθέσεις.

§7.8 Εκχώρηση

61. Τι γνωρίζετε για την εντολή εκχώρησης;

(βλ. ερ. 11)

§7.9 Εντολές εισόδου-εξόδου

62. Τι γνωρίζετε για τις εντολές εισόδου-εξόδου της ΓΛΩΣΣΑΣ;

(βλ. ερ.11)

§7.10 Δομή προγράμματος

63. Ποια είναι η δομή ενός προγράμματος σε ΓΛΩΣΣΑ;

Παράδειγμα όπου φαίνεται η δομή προγράμματος σε ΓΛΩΣΣΑ εικονίζεται στο διπλανό σχήμα

Εφόσον υπάρχουν

Ό,τι διαφορετικοί τύποι μτβλ. υπάρχουν (εφόσον υπάρχουν) τους γράφουμε με όποια σειρά θέλουμε

```
ΠΡΟΓΡΑΜΜΑ όνομα_προγράμματος
ΣΤΑΘΕΡΕΣ
   π=3.14
   g=10
ΜΕΤΑΒΛΗΤΕΣ
   ΛΟΓΙΚΕΣ: βρεθηκε
   ΠΡΑΓΜΑΤΙΚΕΣ: Μεσος_όρος, κλιση
ΑΡΧΗ
   !εντολές
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ

! Εδώ δηλώνονται διαδικασίες
! και συναρτήσεις κεφ.10
```

ΚΕΦΑΛΑΙΟ 8 – ΕΠΙΛΟΓΗ ΚΑΙ ΕΠΑΝΑΛΗΨΗ.

§8.1 Εντολές επιλογής

64. Τι περιλαμβάνουν οι λογικές εκφράσεις της ΓΛΩΣΣΑΣ (απλές και σύνθετες);

Περιλαμβάνουν σταθερές, μεταβλητές, αριθμητικές παραστάσεις, παρενθέσεις και οπωσδήποτε τουλάχιστο ένα συγκριτικό τελεστή (απλές λογικές εκφράσεις). Αν περιλαμβάνονται και λογικοί τελεστές τότε πρόκειται για σύνθετες λογικές εκφράσεις.

65. Ποια η ιεραρχία των τελεστών σε μια έκφραση στη ΓΛΩΣΣΑ;

Αριθμητικοί τελεστές (1. ^ 2. * / mod div 3. + -) , συγκριτικοί τελεστές (όλοι ίδια προτεραιότητα), λογικοί τελεστές (1. ΟΧΙ, 2. ΚΑΙ Η)

66. Μεταξύ ποιων τύπων δεδομένων μπορούν να γίνουν συγκρίσεις στη ΓΛΩΣΣΑ;

Οι συγκρίσεις γίνονται σε αριθμητικά δεδομένα (ακέραιους, πραγματικούς) με προφανή τρόπο .

Οι συγκρίσεις αλφαριθμητικών δεδομένων βασίζονται στην αλφαβητική τους κατάταξη. Έτσι π.χ. η έκφραση 'Γιώργος' > 'Γιάννης' αποτιμάται ως ΑΛΗΘΗΣ αφού ο 'Γιώργος' έπεται του 'Γιάννη' λεξικογραφικά αφού το ω έπεται του α άρα είναι μεγαλύτερο.

Οι συγκρίσεις λογικών δεδομένων έχουν νόημα μόνο στις περιπτώσεις του ίσου (=) και του διάφορου (< >) αφού οι τιμές που μπορούν να έχουν είναι μόνο ΑΛΗΘΗΣ και ΨΕΥΔΗΣ.

§8.1.1 Εντολή ΑΝ

67. Τι γνωρίζετε για τις μορφές της δομής επιλογής; (βλ. §2.4.2, §2.4.3, §2.4.4)

68. Δώστε ένα παράδειγμα ελέγχου περιττών συνθηκών σε δομές επιλογής.

Ένα συχνό λάθος που παρατηρείται στα προγράμματα είναι ο έλεγχος περιττών συνθηκών. Οι επιπλέον έλεγχοι αυξάνουν την πολυπλοκότητα του προγράμματος. Στο διπλανό σχήμα φαίνονται σημειωμένοι οι περιττοί έλεγχοι για ένα παράδειγμα πολλαπλής επιλογής. Οι σημειωμένοι έλεγχοι είναι εξασφαλισμένα αληθείς από τη στιγμή που είναι ψευδείς οι έλεγχοι των προηγούμενων συνθηκών της πολλαπλής επιλογής.

```
ΑΝ βαθμος < 10 ΤΟΤΕ
  ΓΡΑΨΕ 'κατω της βασης'
ΑΛΛΙΩΣ_ΑΝ βαθμος >= 10 ΚΑΙ βαθμος < 13 ΤΟΤΕ
  ΓΡΑΨΕ 'μετρια'
ΑΛΛΙΩΣ_ΑΝ βαθμος >= 13 ΚΑΙ βαθμος < 16 ΤΟΤΕ
  ΓΡΑΨΕ 'καλώς'
ΑΛΛΙΩΣ_ΑΝ βαθμος >= 16 ΚΑΙ βαθμος < 18 ΤΟΤΕ
  ΓΡΑΨΕ 'λιαν καλως'
ΑΛΛΙΩΣ
  ΓΡΑΨΕ 'αριστα'
ΤΕΛΟΣ_ΑΝ
```

§8.2 Εντολές επανάληψης

§8.2.1 Εντολή ΟΣΟ...ΕΠΑΝΑΛΑΒΕ

69. Τι γνωρίζετε για την εντολή της γλώσσας ΟΣΟ...ΕΠΑΝΑΛΑΒΕ

(βλ. §2.4.5 ερ. 18-22)

§8.2.2 Εντολή ΑΡΧΗ...ΜΕΧΡΙΣ_ΟΤΟΥ

70. Τι γνωρίζετε για την εντολή της γλώσσας ΑΡΧΗ...ΜΕΧΡΙΣ_ΟΤΟΥ

(βλ. §2.4.5 ερ. 18-22)

§8.2.3 Εντολή ΓΙΑ... ΑΠΟ... ΜΕΧΡΙ... ΜΕ_ΒΗΜΑ...

71. Τι γνωρίζετε για την εντολή της γλώσσας ΓΙΑ... ΑΠΟ... ΜΕΧΡΙ... ΜΕ_ΒΗΜΑ

(βλ. §2.4.5 ερ. 18-22)

72. Ποιοι κανόνες πρέπει να ακολουθούνται αυστηρά για τη σωστή λειτουργία των προγραμμάτων κατά τη χρήση εμφωλευμένων βρόχων;

- ✔ 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.

ΚΕΦΑΛΑΙΟ 9 - ΠΙΝΑΚΕΣ.

§9.1 Μονοδιάστατοι πίνακες

73. Τι είναι πίνακας; (βλ. §3.3 ερ. 28)

74. Τι μπορεί να χρησιμοποιηθεί ως δείκτης για την αναφορά σε ένα στοιχείο πίνακα;

Ως δείκτης για την αναφορά σε ένα στοιχείο πίνακα μπορεί να χρησιμοποιηθεί ακέραια μεταβλητή με οποιοδήποτε αποδεκτό όνομα, ή ακέραια έκφραση γενικότερα. Είναι σύνηθες στον προγραμματισμό ως δείκτες να χρησιμοποιούνται οι μεταβλητές i, j, k ή εκφράσεις με αυτές. Για μονοδιάστατους πίνακες απαιτείται ένας μόνο δείκτης, για δισδιάστατους δύο που χωρίζονται με κόμμα κλπ

75. Πώς γίνεται η σάρωση πινάκων για ανάγνωση, επεξεργασία και εκτύπωση στοιχείων τους;

Η σάρωση πινάκων για ανάγνωση, επεξεργασία και εκτύπωση στοιχείων τους γίνεται πάντοτε από βρόχους, οι οποίοι επαναλαμβάνονται προκαθορισμένο αριθμό φορών, όσα είναι τα στοιχεία του πίνακα και υλοποιούνται καλύτερα στον προγραμματισμό με την εντολή επανάληψης ΓΙΑ. Προκειμένου για πολυδιάστατους πίνακες η σάρωσή τους υλοποιείται με εμφωλευμένες εντολές ΓΙΑ.

§9.2 Πότε πρέπει να χρησιμοποιούνται πίνακες;

76. Πότε πρέπει να χρησιμοποιούνται πίνακες για την επίλυση προβλημάτων;

Γενικά, αν τα δεδομένα που εισάγονται σε ένα πρόγραμμα πρέπει να διατηρούνται στη μνήμη μέχρι το τέλος της εκτέλεσής του για δευτερογενή (σε δεύτερο στάδιο) σάρωση ή επεξεργασία τους, τότε η χρήση πινάκων είναι απαραίτητη για την επίλυση του προβλήματος. Σε άλλη περίπτωση μπορεί να αποφεύγεται η χρήση τους εκτός κι αν απλοποιεί εμφανώς την κατανόηση-διόρθωση-συντήρηση του προγράμματος. Η απόφαση για την χρήση ή όχι πίνακα για την διαχείριση των δεδομένων είναι κυρίως θέμα εμπειρίας στον προγραμματισμό.

77. Ποια είναι τα μειονεκτήματα από 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. Η ύπαρξη ανώτατου ορίου στο πλήθος των στοιχείων των πινάκων, λόγω του ότι είναι στατικές δομές και το μέγεθος τους πρέπει να δηλώνεται στην αρχή του προγράμματος και παραμένει υποχρεωτικά σταθερό κατά την εκτέλεσή του, αποτελεί περιορισμό των δυνατοτήτων του προγράμματος. Σε περίπτωση που απαιτηθούν περισσότερες θέσεις για την αποθήκευση των δεδομένων του προγράμματος, θα πρέπει ο προγραμματιστής να τροποποιήσει το ανώτατο πλήθος των στοιχείων του χρησιμοποιούμενου πίνακα και να επαναδιανείμει το πρόγραμμα στους χρήστες.

§9.3 Πολυδιάστατοι πίνακες

§9.4 Τυπικές επεξεργασίες πινάκων

78. Ποιες είναι οι τυπικές επεξεργασίες σε στοιχεία πίνακα;

- **Υπολογισμός αθροισμάτων στοιχείων πίνακα.** Προκειμένου για δισδιάστατους πίνακες πολύ συχνά ζητείται το άθροισμα στοιχείων με κοινά χαρακτηριστικά π.χ. βρίσκονται στην ίδια γραμμή ή στήλη.
- **Εύρεση του μέγιστου ή ελάχιστου στοιχείου.** Προκειμένου για μη ταξινομημένους πίνακες πρέπει να γίνει σάρωση όλων των στοιχείων τους ενώ για ταξινομημένους τα ζητούμενα στοιχεία βρίσκονται στις ακριανές θέσεις.
- **Ταξινόμηση στοιχείων πίνακα.** Γίνεται με πολλές μεθόδους. Η πιο απλή αλλά και λιγότερο αποδοτική είναι η ταξινόμηση ευθείας ανταλλαγής ή ταξινόμηση φυσσαλίδας. 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 πίνακα**

  Δύο είναι οι πλέον διαδεδομένοι αλγόριθμοι αναζήτησης, η σειριακή αναζήτηση και 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 δυο (ή περισσότερων) ταξινομημένων πινάκων ενός άλλου, που είναι και αυτός ταξινομημένος. Παρακάτω παρατίθεται ένας αλγόριθμος συγχώνευσης ταξινομημένων πινάκων:

```
Αλγόριθμος MergeSorted
Δεδομένα // A, B, n, m //
!A,B ταξινομημένοι σε αύξουσα σειρά πίνακες n και m στοιχείων αντίστοιχα
j ← 1   !δείκτης σάρωσης A
k ← 1   !δείκτης σάρωσης B
Για i από 1 μέχρι n + m
  Αν j ≤ n και k ≤ m τότε
    Αν A[j] < B[k] τότε
      Γ[i] ← A[j]
      j ← j + 1
    αλλιώς_αν B[k] < A[j] τότε
      Γ[i] ← B[k]
      k ← k + 1
    Τέλος_αν
  αλλιώς_αν j < n τότε
    Γ[i] ← A[j]
    j ← j + 1
  αλλιώς
    Γ[i] ← B[k]
    k ← k + 1
  Τέλος_αν
Τέλος_επανάληψης
Αποτελέσματα // Γ // !ταξινομημένος
Τέλος MergeSorted
```

ΚΕΦΑΛΑΙΟ 10 - ΥΠΟΠΡΟΓΡΑΜΜΑΤΑ.

§10.1 Τμηματικός Προγραμματισμός

79. Τι γνωρίζετε για τον τμηματικό προγραμματισμό

(βλ. §6.4.1, ερ. 64)

80. Τι ονομάζουμε υποπρόγραμμα;

Υποπρόγραμμα ονομάζουμε ένα τμήμα προγράμματος το οποίο επιτελεί ένα αυτόνομο έργο και έχει γραφεί χωριστά από το υπόλοιπο πρόγραμμα .

§10.2 Χαρακτηριστικά των υποπρογραμμάτων

81. Ποιες ιδιότητες πρέπει να διακρίνουν τα υποπρογράμματα;

- ✔ ***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 υποπρογράμμ ατα. 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.

§10.3 Πλεονεκτήματα του τμηματικού προγραμματισμού

82. Ποια είναι τα πλεονεκτήματα του τμηματικού προγραμματισμού

- ✔ **Διευκολύνει την ανάπτυξη του αλγορίθμου και του αντίστοιχου προγράμματος**, μιας και επιτρέπει την εξέταση και την επίλυση απλών προβλημάτων και τη δημιουργία αντίστοιχων υποπρογραμμάτων.
- ✔ **Διευκολύνει την κατανόηση και διόρθωση του προγράμματος**, αφού επιτρέπει εύκολα σε ένα προγραμματιστή να κατανοήσει τις γενικές λειτουργίες που επιτελεί το πρόγραμμα και εύκολα έτσι να εντοπίσει σε ποιο τμήμα-υποπρόγραμμα απαιτείται πιθανή διόρθωση-αλλαγή. Επίσης αυτή η τροποποίηση δε θα επηρεάσει τη λειτουργία των υπολοίπων τμημάτων-υποπρογραμμάτων. Αυτά αποτελούν πολύ βασικά χαρακτηριστικά για ένα πρόγραμμα που πρέπει να συντηρηθεί στο χρόνο ζωής του από πολλούς διαφορετικούς προγραμματιστές.

- ✔ **Απαιτεί λιγότερο χρόνο και προσπάθεια στη συγγραφή του προγράμματος,** μιας και επιτρέπει την επαναχρησιμοποίηση της ίδιας λειτουργίας σε διαφορετικά σημεία ενός προγράμματος μέσω της κλήσης του ίδιου υποπρογράμματος πιθανά με διαφορετικές παραμέτρους. Έτσι μειώνεται ο χρόνος ανάπτυξης, η πιθανότητα λαθών και το πρόγραμμα γίνεται πιο εύληπτο και κατανοητό.
- ✔ **Επεκτείνει τις δυνατότητες των γλωσσών προγραμματισμού,** μιας και μπορεί υποπρογράμματα που χρησιμοποιούνται συχνά από πολλά προγράμματα σε ένα πεδίο εφαρμογής να ενσωματωθούν σε μια βιβλιοθήκη. Έτσι εμπλουτίζονται οι ήδη υπάρχουσες ενσωματωμένες συναρτήσεις μιας γλώσσας προγραμματισμού με τα υποπρογράμματα του χρήστη που του είναι απαραίτητα στη δουλειά του.

§10.4 Παράμετροι

83. Τι είναι οι παράμετροι και ποιος ο ρόλος τους στον τμηματικό προγραμματισμό;

Οι παράμετροι είναι μεταβλητές που επιτρέπουν το πέρασμα των τιμών τους από ένα τμήμα προγράμματος σε άλλο. Οι παράμετροι είναι απαραίτητες για να εξασφαλίσουν την επικοινωνία των τμημάτων-υποπρογραμμάτων που έχουν αναπτυχθεί για την επίλυση ενός προβλήματος. Τα υποπρογράμματα δέχονται συνήθως τιμές (παραμέτρους) από το κυρίως πρόγραμμα ή άλλα υποπρογράμματα που τα καλούν και τους επιστρέφουν νέες τιμές αποτελέσματα.

§10.5 Διαδικασίες και συναρτήσεις

84. Ποια είδη υποπρογραμμάτων γνωρίζετε και που χρησιμεύει καθένα;

Υπάρχουν 2 είδη υποπρογραμμάτων: ***οι διαδικασίες και οι συναρτήσεις***, που τοποθετούνται μετά το τέλος του κυρίου προγράμματος. .

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

- Δεν επιτελούν είσοδο-έξοδο δεδομένων (δεν επιτρέπονται εντολές ΔΙΑΒΑΣΕ ΓΡΑΨΕ)
- Δεν μπορούν να αλλάξουν τις τιμές των παραμέτρων με τις οποίες καλούνται και να τις επιστρέψουν τροποποιημένες στο καλούν τμήμα προγράμματος
- Καλούνται ως τελεστέοι μιας έκφρασης στο καλούν τμήμα προγράμματος.

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

- Επιτελούν και είσοδο-έξοδο δεδομένων
- Μπορούν να μεταβάλλουν τις τιμές των παραμέτρων με τις οποίες καλούνται και να τις επιστρέψουν τροποποιημένες στον καλούν τμήμα προγράμματος.
- Επιλέγονται έναντι των συναρτήσεων όταν πρέπει να επιστραφούν παραπάνω από μία τιμές στο καλούν τμήμα προγράμματος.

§10.5.1 Ορισμός και κλήση συναρτήσεων

85. Πώς ορίζεται και πώς καλείται μια συνάρτηση;

Ο ορισμός μιας συνάρτησης φαίνεται στο σχήμα:

```
ΣΥΝΑΡΤΗΣΗ όνομα(λίστα_παραμέτρων): τύπος_συνάρτησης
Τμήμα_Δηλώσεων
ΑΡΧΗ
  ....
  όνομα<—έκφραση
  ....
ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ
```

- **όνομα**, είναι οποιοδήποτε έγκυρο όνομα της γλώσσας πλην των δεσμευμένων λέξεων.
- **λίστα_παραμέτρων**, είναι ένα σύνολο από 1 ή περισσότερες μεταβλητές χωριζόμενες με κόμμα. Αν δεν υπάρχουν παράμετροι δεν χρησιμοποιούνται οι παρενθέσεις.
- **Τμήμα_Δηλώσεων**, ακριβώς όπως το τμήμα δηλώσεων ενός προγράμματος (ΣΤΑΘΕΡΕΣ, ΜΕΤΑΒΛΗΤΕΣ). Οι μεταβλητές της λίστα_παραμέτρων πρέπει να δηλώνονται στο Τμήμα_Δηλώσεων.
- **Τύπος_συνάρτησης,** μπορεί να είναι μια από τις λέξεις ΑΚΕΡΑΙΑ, ΠΡΑΓΜΑΤΙΚΗ, ΧΑΡΑΚΤΗΡΑΣ, ΛΟΓΙΚΗ, ανάλογα με τον τύπο της τιμής που επιστρέφεται.
- Στις εντολές του σώματος της συνάρτησης πρέπει υποχρεωτικά να υπάρχει μία εντολή εκχώρησης τιμής στο όνομα της συνάρτησης. Αυτή θα είναι της μορφής **όνομα←έκφραση**

Η κλήση μιας συνάρτησης γίνεται μέσα σε οποιοδήποτε έκφραση του καλούντος προγράμματος, όπως οι κλήσεις των ενσωματωμένων συναρτήσεων της ΓΛΩΣΣΑΣ.

§10.5.2 Ορισμός και κλήση διαδικασιών

86. Πώς ορίζεται και πώς καλείται μια διαδικασία;

Ο ορισμός μιας διαδικασίας φαίνεται στο σχήμα:

```
ΔΙΑΔΙΚΑΣΙΑ όνομα(λίστα_παραμέτρων)
Τμήμα_Δηλώσεων
ΑΡΧΗ
  ....
ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ
```

- **όνομα**, είναι οποιοδήποτε έγκυρο όνομα της γλώσσας πλην των δεσμευμένων λέξεων.
- **λίστα_παραμέτρων**, είναι ένα σύνολο από 1 ή περισσότερες μεταβλητές χωριζόμενες με κόμμα. Αν δεν υπάρχουν παράμετροι δεν χρησιμοποιούνται οι παρενθέσεις.
- **Τμήμα_Δηλώσεων**, ακριβώς όπως το τμήμα δηλώσεων ενός προγράμματος (ΣΤΑΘΕΡΕΣ, ΜΕΤΑΒΛΗΤΕΣ). Οι μεταβλητές της λίστα_παραμέτρων πρέπει να δηλώνονται στο Τμήμα_Δηλώσεων.

Η κλήση της διαδικασίας του σχήματος μέσα από το καλούν τμήμα προγράμματος γίνεται ως αυτόνομη εντολή ως εξής:

```
ΚΑΛΕΣΕ όνομα(λίστα_παραμέτρων)
```

§10.5.3 Πραγματικές και τυπικές παράμετροι

87. Ποιες ονομάζονται τυπικές και ποιες πραγματικές παράμετροι;

Τυπικές παράμετροι ή ορίσματα ονομάζονται οι παράμετροι (μεταβλητές) που εμφανίζονται εντός των παρενθέσεων στη δήλωση ενός υποπρογράμματος.

Πραγματικές παράμετροι ή απλά **παράμετροι,** ονομάζονται οι παράμετροι (μεταβλητές, σταθερές, εκφράσεις) που εμφανίζονται εντός παρενθέσεων κατά την κλήση ενός υποπρογράμματος.

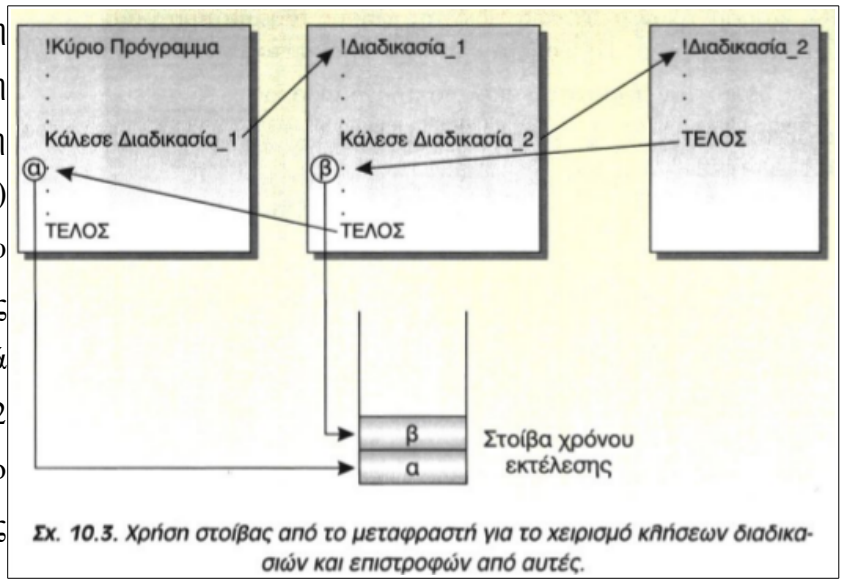
88. Ποιους κανόνες πρέπει να ακολουθούν οι λίστες των παραμέτρων;

- Ο αριθμός των πραγματικών και των τυπικών παραμέτρων πρέπει να είναι ίδιος.
- 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.
- Η τυπική παράμετρος και η αντίστοιχη της πραγματική πρέπει να είναι του ιδίου τύπου.

89. Τι γνωρίζετε για τη στοίβα χρόνου εκτέλεσης (execution time stack);

Όταν μία διαδικασία ή συνάρτηση καλείται από το κύριο πρόγραμμα, τότε η αμέσως επόμενη διεύθυνση του κύριου προγράμματος, που ονομάζεται διεύθυνση επιστροφής (return address), αποθηκεύεται από το μεταφραστή σε μία στοίβα που ονομάζεται στοίβα χρόνου εκτέλεσης (execution time stack). Μετά την ολοκλήρωση εκτέλεσης 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 μία διαδικασία ή συνάρτηση καλεί μία διαδικασία ή συνάρτηση.

Στο διπλανό σχήμα φαίνεται μια περίπτωση όπου από το κυρίως πρόγραμμα καλείται η Διαδικασία_1 που με τη σειρά της καλεί τη Διαδικασία_2. Οι διευθύνσεις επιστροφής (α) και (β) βρίσκονται στη στοίβα χρόνου εκτέλεσης αμέσως μετά την κλήση της Διαδικασία_2. Αυτές γίνονται pop διαδοχικά όταν ολοκληρωθεί εκτέλεση της Διαδικασία_2 και έπειτα της Διαδικασία_1 για να γνωρίζει ο μεταφραστής ποια θα είναι η επόμενη προς εκτέλεση εντολή κάθε φορά.



Σχ. 10.3. Χρήση στοίβας από το μεταφραστή για το χειρισμό κλήσεων διαδικασιών και επιστροφών από αυτές.

§10.6 Εμβέλεια μεταβλητών - σταθερών

90. Τι ονομάζεται εμβέλεια (scope) μεταβλητών ή συμβολικών σταθερών; Ποια είναι η εμβέλεια των μεταβλητών και των συμβολικών σταθερών στη ΓΛΩΣΣΑ;

Εμβέλεια (scope) μεταβλητών ή συμβολικών σταθερών ονομάζεται αυτό που καθορίζει την περιοχή - το τμήμα του προγράμματος όπου ισχύουν οι μεταβλητές ή οι συμβολικές σταθερές σε μια γλώσσα προγραμματισμού.
Στη ΓΛΩΣΣΑ όλες οι μεταβλητές και οι συμβολικές σταθερές είναι τοπικές, δηλαδή είναι γνωστές μόνο στο αντίστοιχο τμήμα προγράμματος (κυρίως πρόγραμμα ή υποπρόγραμμα) όπου δηλώνονται.
Ο μόνος τρόπος για να περάσει τιμή από ένα υποπρόγραμμα σε άλλο (ή από το κυρίως πρόγραμμα σε υποπρόγραμμα) είναι μέσω των παραμέτρων κατά την κλήση του υποπρογράμματος ή μετά το τέλος εκτέλεσής του.

91. Τι είναι η απεριόριστη εμβέλεια (global scope) μεταβλητών και ποια αρχή καταστρατηγεί;

Όσες μεταβλητές ή συμβολικές σταθερές έχουν απεριόριστη εμβέλεια λέγονται καθολικές (global) και είναι γνωστές και μπορούν να χρησιμοποιηθούν σε οποιοδήποτε τμήμα προγράμματος άσχετα πού δηλώθηκαν. Οι καθολικές μεταβλητές καταστρατηγούν την αρχή αυτονομίας των υποπρογραμμάτων, δημιουργούν πολλά προβλήματα και είναι αδύνατο να χρησιμοποιηθούν σε μεγάλα προγράμματα, αφού απαιτούν να γνωρίζει όποιος γράφει ένα υποπρόγραμμα τα ονόματα των καθολικών μεταβλητών που έχουν δηλωθεί από άλλους προγραμματιστές σε άλλα υποπρογράμματα.

92. Τι είναι η περιορισμένη εμβέλεια (local scope) μεταβλητών και συμβολικών σταθερών και ποια τα πλεονεκτήματά της;

Αυτού του είδους η εμβέλεια υποχρεώνει όλες τις μεταβλητές που χρησιμοποιούνται σε ένα τμήμα προγράμματος να είναι δηλωμένες σε αυτό. Οι μεταβλητές ισχύουν μόνο για το υποπρόγραμμα στο οποίο δηλώθηκαν. Η ΓΛΩΣΣΑ παρέχει περιορισμένη εμβέλεια μεταβλητών.
Τα πλεονεκτήματα της περιορισμένης εμβέλειας είναι η απόλυτη αυτονομία των υποπρογραμμάτων και η δυνατότητα να χρησιμοποιείται οποιοδήποτε όνομα χωρίς να ενδιαφέρει αν το ίδιο όνομα χρησιμοποιείται και σε άλλο πρόγραμμα, αφού πρόκειται ουσιαστικά περί ανεξάρτητων μεταβλητών.

93. Τι είναι η μερικώς περιορισμένη εμβέλεια μεταβλητών και συμβολικών σταθερών;

Πρόκειται για τη δυνατότητα που δίνουν ορισμένες γλώσσες προγραμματισμού, άλλες μεταβλητές να ορίζονται ως τοπικές και άλλες ως καθολικές. Αυτή η δυνατότητα προσφέρει μερικά πλεονεκτήματα στον πεπειραμένο προγραμματιστή, αλλά περιπλέκει το πρόγραμμα και την ανάπτυξή του για τον αρχάριο προγραμματιστή.