Αλγόριθμοι για Υπολογιστές
Ένας μικρός οδηγός

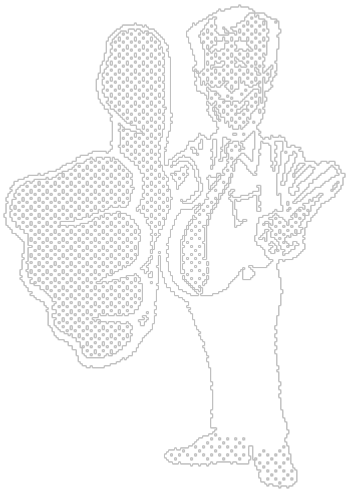
Μάνος Καρβούνης

Για το μάθημα
Εισαγωγή στον Προγραμματισμό

Περιεχόμενα:

Έτοιμοι;

Φύγαμε!



Πρόλογος

Ο τίτλος του παρόντος εγχειριδίου είναι «Αλγόριθμοι για Υπολογιστές». Τι είναι όμως αυτή η αλγοριθμική σκέψη; Κάποιοι από σας ίσως να πουν τώρα –Εγώ ξέρω τι είναι αλγόριθμος, το κάναμε στο Λύκειο και έδωσα και πανελλήνιες, ή και έτσι – Εγώ έχω γράψει ένα πρόγραμμα σε Pascal, κατέχω ήδη την αλγοριθμική σκέψη.

Δεν θα αρνηθώ ότι πολλοί μπορεί να γεννηθήκατε με μια αλγοριθμική-οργανωτική σκέψη, κι αν ισχύει κάτι τέτοιο μπορείτε να θεωρείτε τον εαυτό σας ιδιαίτερα τυχερό. Επίσης, πολλοί μπορεί να έχετε ήδη εμπειρία με τον προγραμματισμό και αυτό είναι μια σαφής ένδειξη ήδη δομημένης αλγοριθμικής σκέψης. Καλό θα ήταν βέβαια, όλοι να ρίξετε μια ματιά σε αυτόν τον οδηγό και να αποφασίσετε αν είναι για σας ή όχι. Πάντως, ο συγκεκριμένος οδηγός το μόνο που έχει σαν προαπαιτούμενο είναι όρεξη για δουλειά, τα υπόλοιπα θα τα μάθουμε μαζί :)

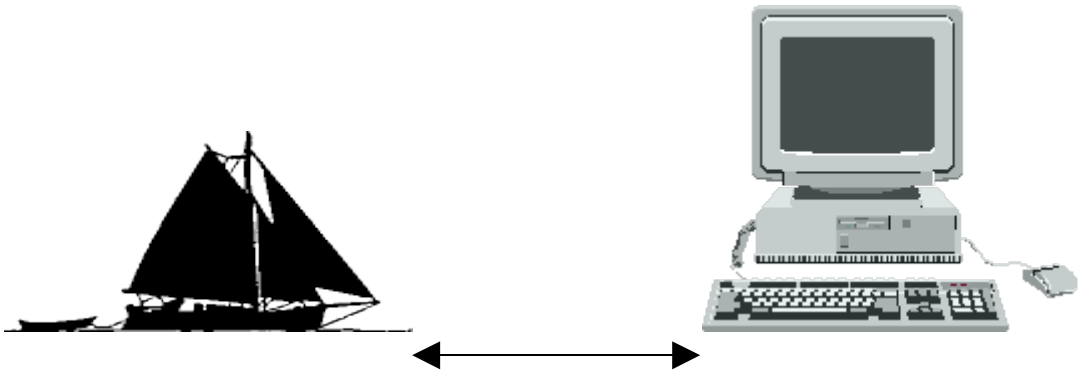
Όταν λοιπόν λέμε αυτή την φράση εννοούμε, με πολύ λίγα λόγια, να μπορούμε να πάρουμε ένα πρόβλημα (οποιασδήποτε φύσεως, από το πώς θα καθαρίσουμε το πληκτρολόγιό μας που έχει γεμίσει σκόνες μέχρι πώς θα σώσουμε τη γη από τους εξωγήινους) και να το αναλύσουμε σε απλά και ξεκάθαρα βήματα. Είναι με άλλα λόγια μια διαφορετική θέαση των πραγμάτων, μια οργάνωσή τους σε μικρά βήματα που ο καθένας μας μπορεί να ακολουθήσει.

Μην σας τρομάζει αυτός ο τρόπος σκέψης, σας διαβεβαιώ ότι τον έχετε χρησιμοποιήσει ουκ ολίγες φορές. Για παράδειγμα, όταν φτιάχνετε ένα γλυκό, η συνταγή που έχετε μπροστά σας είναι ένας αλγόριθμος. Αντί ο Τσελεμεντές (ή ο Μαμαλάκης ή η θεία Βέφα) να σου λέει «Για να φτιάξεις ένα κέικ σοκολάτα χρειάζεσαι κέικ και σοκολάτα» κάθεται και σου αναλύει όσα έχεις να κάνεις σε βηματάκια. Σου λέει τι υλικά χρειάζεσαι, πώς να τα συνδυάσεις και τι ακριβώς να κάνεις ανά πάσα στιγμή κατά τη διαδικασία της παρασκευής.

Αυτό θα προσπαθήσουμε να κάνουμε κι εμείς, δηλαδή να παίρνουμε ένα πρόβλημα και να το αναλύουμε (όχι να φτιάξουμε κέικ!). Όμως θα εστιάσουμε κυρίως σε προβλήματα που θα βάλουμε τον υπολογιστή να τα λύσει για μας, ενώ εμείς απλά θα του παρέχουμε τον αλγόριθμο.

Γι' αυτό το λόγο θα δούμε αρχικά κάποια σημαντικά ζητήματα που θα μας βοηθήσουν να κατανοήσουμε την λειτουργία του υπολογιστή (μόνο όποια κομμάτια μας αφορούν φυσικά), την φύση των αλγορίθμων, τα εργαλεία που έχουμε στη διάθεσή μας για να τους παράγουμε και τελικά, πώς εργαζόμαστε για να φτάσουμε στη διατύπωση ενός απλού και λειτουργικού αλγορίθμου για υπολογιστές.

Ένας συναρπαστικός κόσμος ανοίγεται μπροστά σας, ο κόσμος της πληροφορικής! Σας εύχομαι καλό ταξίδι!



long long way;

Τι είναι ένας αλγόριθμος; Και τι είναι ένας αλγόριθμος για υπολογιστές;

Ένας αλγόριθμος είναι μια σειρά βημάτων που αν ακολουθήσει κανείς θα καταλήξει στην λύση ενός προβλήματος. Είναι δηλαδή, ένα κείμενο οργανωμένο σε βήματα το οποίο το ακολουθείς και κάνεις μια δουλειά. Βέβαια, εδώ υπεισέρχεται ένα πάρα πολύ σημαντικό ζήτημα που θα μας δώσει και μια πρώτη ένδειξη του τι να περιμένουμε σε έναν αλγόριθμο για υπολογιστές.

Το κείμενο του αλγορίθμου λοιπόν, καλούνται να το χειριστούν δύο ομάδες. Η πρώτη ομάδα περιλαμβάνει τους δημιουργούς και η δεύτερη τους εκτελεστές. Όπως είναι προφανές, όσοι ανήκουν στην πρώτη ομάδα είναι επιφορτισμένοι με την παραγωγή του αλγορίθμου ενώ όσοι ανήκουν στη δεύτερη με την εκτέλεση αυτού και την παραγωγή της λύσης στο πρόβλημα. Άρα, μεταξύ αυτών των δύο ομάδων υπάρχει μια εξάρτηση. Η πρώτη ομάδα είναι αναγκασμένη να δημιουργήσει έναν αλγόριθμο που θα είναι κατανοητός από τη δεύτερη ομάδα ώστε αυτή να μπορεί να τον κατανοήσει, να τον εκτελέσει και να παράγει έτσι, αποτελέσματα.

Όταν λοιπόν, στην πρώτη ομάδα ανήκει ένας άνθρωπος και στην δεύτερη ένας άνθρωπος τότε η δουλειά και των δύο είναι εύκολη. Ο δημιουργός δεν έχει παρά να γράψει έναν αλγόριθμο όπως θα τον έγραφε για τον εαυτό του, προσέχοντας φυσικά να μην βάζει τον εκτελεστή να κάνει πράγματα που αυτός δεν γνωρίζει εξαιτίας ελλιπούς γνώσης. Σε κάθε περίπτωση πάντως, δεν υπάρχουν κάποιοι ιδιαίτεροι συντακτικοί ή γραμματικοί κανόνες ή συμβάσεις που πρέπει να ακολουθήσει ο δημιουργός ώστε να γίνει κατανοητός στον εκτελεστή.

Σε έναν αλγόριθμο για υπολογιστές όμως, κάτι τέτοιο δεν ισχύει. Όταν ασχολούμαστε με τη δημιουργία ενός αλγορίθμου για υπολογιστές πρέπει να ακολουθούμε ένα πολύ αυστηρό συντακτικό και λεξιλόγιο. Εμείς θα είμαστε η πρώτη ομάδα (δημιουργοί) και ο υπολογιστής θα είναι η δεύτερη (εκτελεστής), οπότε πρέπει να είμαστε σίγουροι ότι ο αλγόριθμός μας θα είναι κατανοητός από τον υπολογιστή.

Σε αυτόν τον οδηγό λοιπόν, θα ασχοληθούμε με τον τρόπο με τον οποίο πρέπει να κινούμαστε για να δημιουργήσουμε έναν αλγόριθμο κατανοητό από τον υπολογιστή. Θα μάθουμε πως να αξιοποιούμε όλες τις εκπληκτικές δυνατότητες και τα εργαλεία που μας προσφέρει, αλλά και πως να μένουμε εντός των ορίων των περιορισμών που μας υπαγορεύει η επικοινωνία με αυτόν τον καινούργιο μας συνεργάτη.

Ας αρχίσουμε λοιπόν, βλέποντας τα γενικά χαρακτηριστικά που πρέπει να έχει ένας αλγόριθμος για να τον κατανοήσει ένας υπολογιστής. Γενικά, μέσα στον οδηγό θα αναφερόμαστε σε έναν αλγόριθμο για υπολογιστή με το όνομα “απλός”. Αυτό βέβαια, είναι απλά μια σύμβαση μεταξύ μας για να θυμόμαστε ότι πρέπει να είμαστε ιδιαίτερα προσεχτικοί με το πως διατυπώνουμε έναν αλγόριθμο για υπολογιστές.

Ο απλός αλγόριθμος λοιπόν, έχει τις εξής ιδιότητες:

1. Κάθε βήμα του είναι απλό και επιδέχεται ακριβώς μία ερμηνεία
2. Κάθε βήμα του μπορεί να εκτελεστεί αυτοτελώς, χωρίς ο εκτελεστής να πρέπει να ανατρέξει σε άλλα βήματα για να καταλάβει τι πρέπει να κάνει (εκτός κι αν ορίζεται ρητά μέσω παραπομπής)
3. Ο εκτελεστής του αλγορίθμου δεν πρέπει να είναι αναγκασμένος να πάρει σύνθετες αποφάσεις ή οποιασδήποτε μορφής πρωτοβουλία

Για να εμπεδώσουμε τη διαφορά μεταξύ ενός αλγορίθμου για υπολογιστές και ενός για ανθρώπους ας δούμε ένα παράδειγμα απλού και μη απλού αλγόριθμου για ένα προβλήματα.

Πρόβλημα: Πώς θα φτάσω μέχρι την πόρτα του δωματίου μου

Αλγόριθμος 1	Αλγόριθμος 2
1. Πήγαινε ευθεία	1. Πήγαινε ευθεία μέχρι λίγο πριν τον καναπέ
2. Στρίψε δεξιά	2. Στρίψε δεξιά και σταμάτα λίγο πριν την πόρτα

Για να δείτε και μόνοι σας ποιος είναι κατάλληλος για υπολογιστή και ποιος όχι σας παροτρύνω να κάνετε το εξής: Ξεχάστε για λίγο όλα όσα ξέρετε για τη ζωή και σκεφτείτε ότι το μόνο που ξέρετε να κάνετε είναι να κινείστε στις 3 διαστάσεις. Μόνο αυτό όμως, κατά τα άλλα δεν ξέρετε απολύτως τίποτα. Εκτελέστε τους δύο παραπάνω αλγορίθμους (υποθέτουμε ότι το δωμάτιό σας είναι κατάλληλα δομημένο για να μην πέσετε από το μπαλκόνι άμα κάνετε όσα γράφονται!).

Είδατε ότι υπήρξε ένα πρόβλημα στον πρώτο, για να μην πέσετε πάνω στον καναπέ έπρεπε να πάρετε την απόφαση να σταματήσετε, ενώ όταν στρίψατε δεξιά έπρεπε πάλι να σταματήσετε κάποια στιγμή για να μην κουτουλάτε στους τοίχους. Αυτό είναι κακό γιατί υποτίθεται ότι δεν ξέρετε τέτοια πράγματα, εσείς μόνο να κινείστε ξέρετε, όχι να σκέφτεστε!

Ο δεύτερος αλγόριθμος δεν απαιτεί από σας να πάρετε κάποια απόφαση, αν και απαιτεί να πάρετε την πρωτοβουλία να ορίσετε το «λίγο πριν». Αυτό ναι μεν είναι κακό, αλλά μπορούμε να το διορθώσουμε εύκολα ορίζοντας τι εννοούμε.

Με βάση τα προηγούμενα καταλήγουμε στον εξής απλό αλγόριθμο:

1. Κινήσου ευθεία μέχρι λίγο πριν τον καναπέ
2. Στρίψε δεξιά
3. Όσο δεν έχεις φτάσει λίγο πριν την πόρτα εκτέλεσε το βήμα4
4. Κινήσου ευθεία

Και παράλληλα ορίζουμε το λίγο πριν:

Λίγο_πριν(αντικείμενο) { Κάθετη απόσταση 20cm από το αντικείμενο }

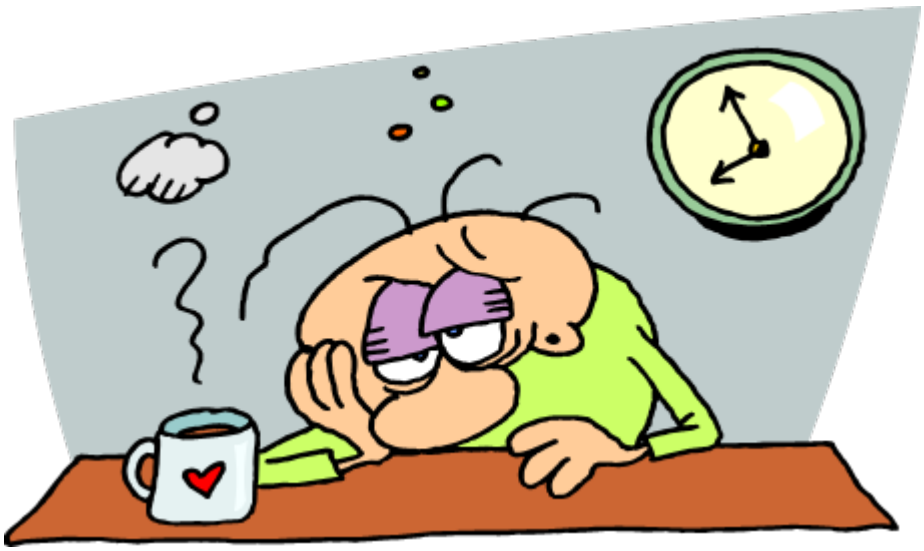
Μην σας φαίνεται παράξενος ο προηγούμενος ορισμός, θα τον καταλάβετε κάποια στιγμή απόλυτα. Προς το παρόν να ξέρετε ότι το αντικείμενο που περικλειόταν στις παρενθέσεις μπορεί να πάρει όποια τιμή του δώσετε εσείς (καναπές, πόρτα, παράθυρο, γκρεμός) και έτσι κάνει τον ορισμό γενικό και επαναχρησιμοποιήσιμο.

Για το παράδειγμά μας δηλαδή:

Λίγο_πριν(καναπές) { Κάθετη απόσταση 20cm από τον καναπέ }
Λίγο_πριν(πόρτα) { Κάθετη απόσταση 20cm από την πόρτα }

Βέβαια, με έναν υπολογιστή δεν θα κληθείτε (στα πλαίσια του μαθήματος της Εισαγωγής στον Προγραμματισμό) να κάνετε ένα πρόγραμμα που σηκώνει τον υπολογιστή και τον πάει μέχρι την πόρτα :D Αλλά το σημαντικό είναι να καταλάβετε ότι δεν πρέπει να προσεγγίζουμε έναν υπολογιστή όπως έναν άνθρωπο ή να περιμένουμε να αντιλαμβάνεται τα πράγματα και να ενεργεί όπως εμείς. Γι' αυτό το λόγο πρέπει να μάθουμε επικοινωνούμε μαζί του στη γλώσσα που καταλαβαίνει. Και αυτό θα ξεκινήσουμε σιγά-σιγά από το επόμενο κεφάλαιο.

Ζεσταθήκαμε;



Ωραία, μια χαρά σας βλέπω, συνεχίζουμε!

Ο υπολογιστής, δυνατότητες και περιορισμοί

Όπως αναφέρθηκε πολλές φορές ως τώρα, θα μας απασχολήσει η δημιουργία αλγορίθμων κατάλληλα κατασκευασμένων ώστε να μπορούν να εκτελεστούν από έναν ηλεκτρονικό υπολογιστή. Για να καταλάβουμε αρχικά, τι διαφορές πρέπει να περιμένουμε σε έναν αλγόριθμο για ανθρώπους σε σχέση με έναν για υπολογιστές ας δούμε πάλι ένα από τα προηγούμενα προβλήματα

Πρόβλημα: Πώς θα φτάσω μέχρι την πόρτα του δωματίου μου

Αλγόριθμος που σκοπό του έχει να εκτελεστεί από άνθρωπο:

1. Κοίταξε το χώρο γύρω σου
2. Εντόπισε την πόρτα
3. Κινήσου προς αυτή

Εδώ πρέπει να καταλάβουμε ότι ο παραπάνω αλγόριθμος είναι απολύτως συμβατός με έναν άνθρωπο, δηλαδή θα μπορούσαμε να τον εκτελέσουμε χωρίς κανένα πρόβλημα. Όμως αυτό συμβαίνει γιατί εντολές της μορφής «Κοίταξε», «Εντόπισε», «Αποφάσισε», «Αισθάνσου» είναι μέρος των φυσικών μας ικανοτήτων.

Ένας υπολογιστής όμως, δεν μπορεί να καταλάβει τέτοιες εντολές, δεν μπορούμε να του δώσουμε έναν τέτοιο αλγόριθμο και να περιμένουμε να κάνει κάτι (πέρα απ' το να μας ανάψει απορημένος τα λαμπάκια του σκληρού του δίσκου). Ο υπολογιστής έχει ορισμένα χαρακτηριστικά που του δίνουν κάποιες εκπληκτικές ικανότητες, αλλά παράλληλα έχει ένα σύνολο περιορισμών ως προς τον τρόπο που μπορεί να εργαστεί και να αντιληφθεί τα πράγματα.

Ας αρχίσουμε πρώτα με τις ικανότητές του:

1. Μπορεί να εκτελέσει αριθμητικές πράξεις με ταχύτητες ασύλληπτες για τον ανθρώπινο νου. Μετά μπορεί βασιζόμενος σε ένα σύνολο απλών πράξεων να αντιληφθεί και να εκτελέσει ορισμένες πιο σύνθετες (εδώ είναι που παίζουν το ρόλο τους οι γλώσσες προγραμματισμού, αλλά αυτό θα το δούμε αργότερα)
2. Μπορεί να αποθηκεύσει τεράστιο αριθμό δεδομένων και να τα ανακτά με εκπληκτικές ταχύτητες. Μετά μπορεί να εκτελέσει πάνω τους κάποιες απλές διαδικασίες επεξεργασίας
3. Είναι πολύ υπάκουος, κάνει ακριβώς ό,τι του λέμε, αρκεί να μπορεί να τα καταλάβει!

Το τελευταίο είναι βέβαια δίκοπο μαχαίρι

*«Ο υπολογιστής έχει ένα καλό, κάνει **ακριβώς** ό,τι του λέμε. Ο υπολογιστής όμως έχει κι ένα κακό, κάνει ακριβώς **ό,τι** του λέμε»*
- Στέφανος Σταμάτης -

Σε σχέση με τους περιορισμούς του

1. Είναι χαζός με όλη τη σημασία της λέξης, είναι ένα φυτό. Ακόμα κι όταν φαίνεται «έξυπνος» πάντα κρύβεται από πίσω ένας ικανός προγραμματιστής που ξόδεψε πολύ χρόνο για να σκεφτεί όλα τα πιθανά ενδεχόμενα και να δώσει λεπτομερείς οδηγίες στον χαζούλη συνεργάτη του.
2. Δεν παίρνει καμία πρωτοβουλία, δεν σκέφτεται καν. Κάνει *ακριβώς* ότι του λέμε, τίποτα περισσότερο τίποτα λιγότερο. Δεν έχει αίσθημα αυτοσυντήρησης ή κοινή λογική, πράγματα που σε μας είναι αυτονόητα σε αυτόν πρέπει να περιγραφούν με λεπτομέρεια για να αντιδράσει όπως προσδοκούμε.
3. Δεν έχει γενική εικόνα αυτού που εκτελεί, κάθε στιγμή βλέπει ακριβώς ένα βήμα του αλγορίθμου (ή όπως θα λέμε όταν προγραμματίζουμε, ακριβώς μία γραμμή κώδικα). Δεν μπορούμε να περιμένουμε να πάρει κάποια «στάση» απέναντι στον αλγόριθμο βασιζόμενος σε όσα είδε ως τότε.
4. Καταλαβαίνει μόνο «απλούς αλγορίθμους» οι οποίοι είναι δομημένοι με τρόπο που θα αναλύσουμε εκτενώς στο αμέσως επόμενο κεφάλαιο.

Ο υπολογιστής λοιπόν, είναι ένας πολύ υπάκουος, ακούραστος, ταχύτατος, αλλά πάνχαζος συνεργάτης μας. Κι επειδή είναι δύσκολο να μάθει να επικοινωνεί μαζί μας θα πρέπει εμείς να πέσουμε στο επίπεδό του και να μάθουμε την απλοϊκή γλώσσα του. Αυτό λοιπόν θα κάνουμε συστηματικά στο επόμενα κεφάλαιο!

Ετοιμαστείτε, τα δύσκολα ξεκινούν!!

Ο θείος Πίτζι
Έχει όλα τα χαρακτηριστικά του υπολογιστή,
αν δεν μπορείτε να γράψετε αλγόριθμο για υπολογιστές
φανταστείτε ότι γράφετε για αυτόν :-D

Τι εργαλεία έχω για να φτιάξω έναν απλό αλγόριθμο;

Σε αυτό το κεφάλαιο επιτέλους θα μπούμε στο δια ταύτα. Θα εστιάσουμε στα βήματα που πρέπει να ακολουθούμε για να καταλήξουμε στην παραγωγή ενός αλγορίθμου κατανοητού από υπολογιστή. Θα μάθουμε κάποιες βασικές τεχνικές και τα εργαλεία που έχουμε στη διάθεσή μας για τη δημιουργία των αλγορίθμων.

Σε αυτό το σημείο είναι καλό να τονιστεί ότι θα εστιάσουμε αρκετά, αλλά ανεπαίσθητα, στην δημιουργία αλγορίθμων ακολουθώντας τις συμβάσεις και τις δυνατότητες που μας προσφέρει μια γλώσσα διαδικαστικού προγραμματισμού όπως η C. Αρκετά απ'όσα θα πούμε έχουν γενική εφαρμογή, αλλά σταδιακά θα κληθείτε να εξοικειωθείτε και με άλλους τύπους προγραμματισμού όπως τον αντικειμενοστραφή ή το λογικό. Αλλά για όλα υπάρχει μια αρχή και αυτό ακριβώς θα προσπαθήσουμε να κάνουμε τώρα, όσο πιο περιγραφικά και απλά γίνεται.

Ας δούμε πρώτα εν συντομία τι θα αναλύσουμε σε αυτό το νευραλγικό κεφάλαιο:

1. Συλλογή πληροφοριών σχετικά με το πρόβλημα
2. Τα δεδομένα μας, πως τα συλλέγουμε και τα δομούμε
3. Είσοδος και έξοδος στον αλγόριθμο
4. Παραγωγή της εξόδου με κατάλληλες επεξεργασίες
5. Μορφή επιτρεπτών εντολών σε έναν αλγόριθμο για υπολογιστή (με μια ανεπαίσθητη εστίαση στη γλώσσα προγραμματισμού C)
6. Αλγοριθμικές δομές (επανάληψη, επιλογή, ακολουθία)
7. Γενική σκιαγράφηση αλγορίθμου σε ανώτερο επίπεδο
8. Διαδικασίες για την αποσαφήνιση του αλγορίθμου

Πάμε λοιπόν, να αναλύσουμε σιγά-σιγά αυτά τα ζητήματα. Στο τέλος του οδηγού υπάρχει ένα παράρτημα όπου μπορείτε να βρείτε μια κάρτα με τα ακριβή βήματα που μπορούμε να ακολουθούμε για την παραγωγή ενός αλγορίθμου για υπολογιστή, όπως αυτά θα αναπτυχθούν σιγά-σιγά παρακάτω.

Συλλογή πληροφοριών σχετικά με το πρόβλημα

Όταν έχουμε μπροστά μας ένα πρόβλημα πρέπει να το διαβάσουμε και να το κατανοήσουμε καλά. Αφότου το κάνουμε αυτό δεν θα είναι λίγες οι φορές που θα δούμε ότι οι γνώσεις μας δεν είναι επαρκείς για την επίλυσή του, ότι δεν γνωρίζουμε κάποια απαραίτητη λεπτομέρεια ή απλά ότι για πρώτη φορά συναντάμε μπροστά μας ένα τέτοιο ζήτημα. Μια τέτοια κατάσταση στο επάγγελμα του προγραμματιστή-αναλυτή είναι πολύ συνηθισμένο φαινόμενο και γι' αυτό πρέπει όλοι μας να είμαστε έτοιμοι να αναζητήσουμε πληροφορίες. Και στην πραγματική ζωή έτσι γίνεται και στα πλαίσια του μαθήματος έτσι θα πρέπει να κινείστε.

Πολύ καλές πηγές πληροφοριών είναι οι εξής (αναφέρομαι βέβαια μόνο για όσα ενδέχεται να σας χρειαστούν στα πλαίσια του μαθήματος, στην πραγματική ζωή θα βρείτε σίγουρα μια δική σας μέθοδο ;)) :

1. Ο διδάσκων και οι μεταπτυχιακοί (είτε μέσω του forum είτε από κοντά)
2. Οι συμφοιτητές σας και κυρίως από μεγαλύτερα έτη
3. Η ιστοσελίδα www.google.com που είναι μια μηχανή αναζήτησης. Αν θέλετε να μάθετε πώς να την αξιοποιείτε σωστά μπορείτε να ρωτήσετε κάποιον από τις προηγούμενες δύο κατηγορίες (είδατε; Κιόλας όσα λέμε μπαίνουν σε εφαρμογή!)
4. Εγκυκλοπαίδειες, περιοδικά και βιβλία

Η αναζήτηση πληροφοριών, η έρευνα πάνω στο πρόβλημα, όταν γίνεται σωστά και μεθοδικά είναι ένα από τα πιο ενδιαφέροντα μέρη του προγραμματισμού. Και σίγουρα είναι και ένας σημαντικός τρόπος που το Πανεπιστήμιο διαχωρίζεται από το Λύκειο, εδώ θα σας παρέχονται μόνο οι κατευθύνσεις, το πώς θα κινηθείτε είναι δική σας ευθύνη!

Τα δεδομένα μας, πως τα συλλέγουμε και τα αναπαριστούμε

Όταν αισθανόμαστε ότι γνωρίζουμε τα απαραίτητα για το πρόβλημά μας είναι η ώρα να το δούμε λίγο πιο αναλυτικά. Διερωτόμαστε λοιπόν, ποια είναι τα δεδομένα μας; Ποιο είναι το σύνολο των πραγμάτων που μπορώ να θεωρήσω ότι μου παρέχονται και τα οποία θα χρησιμοποιήσω για να φτάσω σε μία λύση; Πώς θα τα συλλέξω; Πώς θα τα αναπαραστήσω; Πώς θα τα δομήσω;

Ας πάρουμε τα ερωτήματα αυτά ένα προς ένα και ας πάρουμε και ένα παράδειγμα για να μας βοηθήσει.

Πρόβλημα: Ένα εκπαιδευτικό ίδρυμα σας ζητάει να του φτιάξετε ένα πρόγραμμα για τον υπολογισμό του βαθμού πτυχίου των αποφοίτων του.

Αρχικά βλέπουμε ότι μας λείπουν πολύτιμες πληροφορίες για την επίλυση του προβλήματος. Δε γνωρίζουμε πόσα μαθήματα πρέπει να περάσει ο υποψήφιος, ποια είναι η κλίμακα της βαθμολογίας, τη διαδικασία που εξάγεται ο βαθμός πτυχίου κλπ. Γι' αυτό το λόγο ζητάμε διευκρινήσεις (το σημαντικό είναι να αναγνωρίσετε ότι έπρεπε να κινηθούμε κατάλληλα για τη **συλλογή πληροφοριών**).

Ας υποθέσουμε ότι μαθαίνετε ότι κάθε υποψήφιος διαγωνίζεται σε 5 μαθήματα, η βαθμολογική κλίμακα είναι 0 έως 10 και ο βαθμός πτυχίου είναι απλά ο μέσος όρος των επιμέρους βαθμών.

- *Βήμα συλλογής δεδομένων*

Το επόμενο βήμα είναι να δούμε ποια είναι τα δεδομένα μας. Για να τα βρούμε απαντάμε στην ερώτηση «*Ποια είναι τα αντικείμενα που πρέπει να επεξεργαστώ για να λύσω το πρόβλημα;*». Τα δεδομένα μας θα είναι τα μόνα πράγματα που ξέρουμε για το πρόβλημα και ο εντοπισμός τους είναι ζωτικής σημασίας. Εδώ παρατηρούμε ότι είναι τα 5 μαθήματα που διαγωνίζεται ο κάθε υποψήφιος.

- Βήμα αναπαράστασης δεδομένων

Δυστυχώς όμως ο υπολογιστής δεν θα καταλάβει τίποτα αν μέσα στον αλγόριθμο μας αναφέρουμε τη λέξη μάθημα (αν και αργότερα θα δείτε ότι υπάρχουν γλώσσες προγραμματισμού που το επιτρέπουν αυτό). Ας δούμε τι είδους δεδομένα μπορεί να καταλάβει:

1. **Αριθμούς**, ακεραίους ή πραγματικούς (μέχρι κάποιο σημείο επιθυμητής ακρίβειας)
2. **Χαρακτήρες**, δηλαδή σύμβολα όπως το %,*,(,@ και οτιδήποτε άλλο μπορεί να χαρακτηριστεί ως σύμβολο (ακόμα κι ένας αριθμός ή ένα γράμμα της αλφαβήτου μπορεί να θεωρηθεί σύμβολο)
3. **Συμβολοσειρές**, δηλαδή ένα σύνολο από συνεχόμενους χαρακτήρες που δομούν μια φράση (για παράδειγμα το «Προσπαθώ να φτιάξω αλγόριθμο, αχ τα νεύρα μου :@:@ !!!!» είναι μια συμβολοσειρά)

Μόνο με τη χρήση των παραπάνω τύπων μπορεί ο υπολογιστής να καταλάβει τα δεδομένα που του δίνουμε. Γι΄ αυτό πρέπει να σκεφτόμαστε πως θέλουμε να **αναπαραστήσουμε**, με βάση τα προηγούμενα, τα δεδομένα μας.

Στα πλαίσια του διαδικαστικού προγραμματισμού δεν θα ασχοληθούμε με πολύπλοκες αναπαραστάσεις, τα δεδομένα μας θα είναι συνήθως ήδη ενός εκ των προηγούμενων τύπων. Σε συνθήκες όμως, αντικειμενοστραφούς προγραμματισμού θα ορίζετε νέους τύπους δεδομένων για αναπαράσταση ολόκληρων οντοτήτων. Τελικά όμως, ακόμα κι εκεί, η βάση θα είναι οι προηγούμενοι στοιχειώδεις τύποι.

Για να κάνουμε την αναπαράσταση με επιτυχία πρέπει να σκεφτούμε ποια από τα χαρακτηριστικά των δεδομένων μας αφορούν το πρόβλημα. Μετά αυτά, μόνο αυτά, θα τα αναπαραστήσουμε, κάνοντας φυσικά κάποιες συμβάσεις. Μην περιμένετε να δείτε σε κάποιο πρόγραμμα ένα μάθημα, με τον καθηγητή του και το βιβλίο του! Ας συνεχίσουμε όμως με το παράδειγμά μας για να τα δούμε καλύτερα.

Ένα μάθημα ας πούμε έχει ένα σύνολο πολλών χαρακτηριστικών, το βιβλίο του, τον καθηγητή του, τις ώρες διδασκαλίας του. Εμάς όμως δεν μας ενδιαφέρει αν το μάθημα έχει καλογραμμένο βιβλίο ή αν έχει ιδιότροπο καθηγητή, εμείς θέλουμε να βρούμε έναν μέσο όρο. Γι' αυτό το μόνο που μας νοιάζει είναι ο βαθμός του υποψηφίου σε αυτό. Και γιατί μας νοιάζει μόνο αυτό; Εδώ είναι που μπαίνει το ζήτημα του τι θέλουμε **να δώσουμε ως λύση** στο πρόβλημα. Αν το εντοπίσουμε αυτό τότε η αναπαράσταση των δεδομένων γίνεται εύκολη υπόθεση.

- Βήμα καθορισμού των χαρακτηριστικών που μας αφορούν

Με το να καθορίσουμε ποια είναι η μορφή της λύσης και πως αυτή μπορεί να προέλθει από τα δεδομένα μας μπορούμε να απορρίψουμε όλα εκείνα τα χαρακτηριστικά των δεδομένων που είναι περιττά. Για παράδειγμα, στο πρόβλημα του υπολογισμού ενός αθροίσματος η λύση που μοιάζει αποδεκτή είναι ένας αριθμός ως το άθροισμα των αριθμών που μας δίνονται. Οπότε από τα δεδομένα μας, που θα είναι και αυτά αριθμοί, μας ενδιαφέρει η τιμή τους και όχι ας πούμε, το πρόσημό τους. Προσπαθήστε να βρείτε τη μορφή που πρέπει να έχουν οι λύσεις των παρακάτω προβλημάτων και ποια είναι τα δεδομένα σε κάθε περίπτωση. Πώς θα τα αναπαριστούσατε;

1. Να βρεθεί το ν-οστό γράμμα του λατινικού αλφαβήτου
2. Να δίνουμε τον αριθμό μητρώου ενός ατόμου και να μας επιστρέφει το όνομά του
3. Ένα σύνολο αεροπλάνων πρέπει να προσγειωθεί σε ένα αεροδρόμιο. Αν κατά τη διάρκεια της προσγείωσης υπάρχει η πιθανότητα να έρθουν και άλλα αεροπλάνα να βρείτε το μέσο χρόνο αναμονής μέχρι την προσγείωση.

Οπότε όλη η ουσία είναι να απορρίπτουμε τα χαρακτηριστικά των δεδομένων που δεν μας ενδιαφέρουν και να κρατάμε μόνο τα απολύτως απαραίτητα για τη λύση του προβλήματος. Μετά αυτά θα μπορούμε να τα αναπαραστήσουμε με κάποιον από τους τύπους που αναφέραμε.

Στο παράδειγμά μας μια μορφή λύσης που θα δίναμε είναι ένας αριθμός που αναπαριστά το μέσο όρο. Αυτός ο αριθμός προέρχεται από το άθροισμα των βαθμών των μαθημάτων προς το πλήθος τους. Άρα το μόνο που μας ενδιαφέρει για κάθε μάθημα είναι ο βαθμός του υποψηφίου σε αυτό, δηλαδή ένας αριθμός. Οπότε θα αναπαραστήσουμε κάθε μάθημα με έναν ακέραιο αριθμό.

Ας δούμε τώρα ένα ζήτημα που είναι μεγάλης σημασίας, αυτό του τι θέλουμε να είναι η είσοδος και η έξοδος του αλγορίθμου μας.

Είσοδος και έξοδος στον αλγόριθμο

Κάθε αλγόριθμος για να είναι αποτελεσματικός, δηλαδή να μπορεί να λύσει κάτι, πρέπει να δέχεται εισόδους και να παράγει εξόδους.

Οι είσοδοι όπως είναι απολύτως φυσικό είναι τα δεδομένα μας, με τη μορφή φυσικά που κάναμε την σύμβαση να τα αναπαριστούμε. Αυτά θα τα δεχτεί ο αλγόριθμος, θα τα επεξεργαστεί και θα τα χρησιμοποιήσει για να λύσει το πρόβλημα.

Για να δούμε όμως, πώς μπορούμε να δώσουμε την εντολή σε έναν απλό αλγόριθμο για υπολογιστή να πάρει ως είσοδο κάποια δεδομένα.

Αρχικά πρέπει να ορίσουμε όλες τις μεταβλητές που θέλουμε να χρησιμοποιήσουμε για την αναπαράσταση των δεδομένων μας. Αυτό το βήμα είναι πολύ σημαντικό γιατί μόνο έτσι αντιλαμβάνεται ο υπολογιστής την πρόθεσή μας να χρησιμοποιήσουμε τις συγκεκριμένες μεταβλητές, ενώ παράλληλα δεσμεύει και τον κατάλληλο χώρο στη μνήμη του για να αποθηκεύσει τις τιμές τους. Κατά το βήμα της δήλωσης μας ενδιαφέρει μόνο τι τύπου έχουμε αποφασίσει να είναι τα δεδομένα μας, όχι τι τιμές θα πάρουν ή πως σκοπεύουμε να τις χρησιμοποιήσουμε.

Στο παράδειγμα μας επομένως θα κάναμε τους παρακάτω ορισμούς:

1. Όρισε μια μεταβλητή τύπου ακεραίου με όνομα mat0
2. Όρισε μια μεταβλητή τύπου ακεραίου με όνομα mat1
3. Όρισε μια μεταβλητή τύπου ακεραίου με όνομα mat2
4. Όρισε μια μεταβλητή τύπου ακεραίου με όνομα mat3
5. Όρισε μια μεταβλητή τύπου ακεραίου με όνομα mat4

Πιο σωστό είναι να χρησιμοποιήσουμε έναν πίνακα για την ομαδοποίηση αυτών των ομοειδών δεδομένων (δείτε το παράρτημα στο τέλος του οδηγού για τις δομές δεδομένων αν δεν ξέρετε τι είναι ο πίνακας), αντί για ξεχωριστές μεταβλητές. Αυτό και πιο απλή κάνει τη δήλωση και πολύ πιο εύκολη την μετάβασή μας μεταξύ των μαθημάτων.

1. Όρισε έναν μονοδιάστατο πίνακα ακεραίων μεγέθους 5 με όνομα vathmoi

Το απολύτως σωστό, και αυτό που θα χρησιμοποιείτε κατά κόρον, είναι ο ορισμός μόνο μιας μεταβλητής η οποία θα συσσωρεύει σταδιακά το άθροισμα των βαθμών των μαθημάτων. Θα το παραβλέψουμε αυτό προς το παρόν και θα χρησιμοποιήσουμε τον τρόπο με τον πίνακα. Στα προβλήματα στο τέλος του οδηγού υπάρχει αναλυτικά ο τρόπος που αναφέραμε τώρα με την μεταβλητή-συσσωρευτή και είναι καλό να ρίξετε μια καλή ματιά.

Κατά τον ορισμό μιας μεταβλητής αυτό που γίνεται είναι να δημιουργείται μέσα στη μνήμη του υπολογιστή ένα κουτί (καλά, μη το παίρνετε και τοις μετρητοίς!) όπου μπορούμε να τοποθετήσουμε μια τιμή του τύπου που καθορίζουμε. Για να αναγνωρίζουμε κι εμείς και ο υπολογιστής αυτό το κουτί από τα εκατομμύρια άλλα που έχει στη μνήμη του, του δίνουμε ένα όνομα. Ο πίνακας δεν είναι τίποτα περισσότερο από τέτοια κουτιά - μεταβλητές κολλημένα μεταξύ τους.

Σημαντικό: Αν δεν κάναμε τους ορισμούς με ακριβώς αυτόν τον τρόπο ή δεν τους κάναμε καθόλου ο υπολογιστής (ή αλληγορικά ο κύριος Πίτζι) δεν θα καταλάβαινε τίποτα, δεν θα ήξερε τίποτα. Είπαμε, είναι χαζός και δεν μπορεί να δει πέρα από τη “μύτη” του.

Αφού έχουμε κάνει τους ορισμούς των μεταβλητών είναι καιρός να τους δώσουμε κάποια τιμή. Αυτό γίνεται γενικά με την εντολή «Πάρε τιμή για …» όπου … το όνομα μιας μεταβλητής, ή και έτσι «… = τιμή». Στην πρώτη περίπτωση δίνουμε τη δυνατότητα κατά την εκτέλεση του αλγορίθμου να καθορίζουμε ποιες τιμές θέλουμε να έχουν οι μεταβλητές μας ενώ στη δεύτερη περίπτωση δίνουμε κάποιες προκαθορισμένες τιμές που θα είναι ίδιες σε όλες τις εκτελέσεις του αλγορίθμου. Οπότε με τη χρήση της εντολής «Πάρε τιμή» πετυχαίνουμε να προσαρμόζουμε τον αλγόριθμό μας στις εκάστοτε τιμές και να είναι ο χρήστης αυτός που θα τις καθορίζει.

Αν θέλουμε να πάρουμε μια τιμή για κάποιο κουτί ενός πίνακα πρέπει να καθορίσουμε και ποιο είναι αυτό, δηλαδή να πούμε «Πάρε τιμή για pinakas[0]».

Στο παράδειγμά μας θα λέγαμε:

6. Πάρε τιμή για vathmoi[0] κλπ

Μια τεράστια χρησιμότητα του πίνακα είναι ότι μπορούμε να κινούμαστε εύκολα μεταξύ των κελιών του. Θα μπορούσαμε με τη χρήση μιας κατάλληλης επαναληπτικής δομής να πάρουμε τιμές για κάθε κελί του χωρίς να γράφουμε εντολές για καθένα ξεχωριστά (αλλά αυτό θα το δούμε αργότερα).

Το ζήτημα της **εξόδου** είναι εξίσου ζωτικής σημασίας με αυτό της εισόδου κυρίως γιατί μας βάζει να αναλογιστούμε ποια θα είναι η μορφή της λύσης που θέλουμε (όπως αναφέραμε στο προηγούμενο κεφάλαιο).

Η έξοδος αυτή καθ' αυτή είναι συνήθως ένα μήνυμα ή, σε κάποιες περιπτώσεις, κάποιο αντικείμενο (αυτό θα το δούμε πιο αναλυτικά αργότερα όταν θα μιλήσουμε για διαδικασίες). Όμως μέχρι να φτάσουμε στην διατύπωση της εντολής που θα μας δίνει την έξοδο θα έχουν προηγηθεί πολλές επεξεργασίες και συμβάσεις. Αυτό ακριβώς είναι που μας ενδιαφέρει περισσότερο, να προσπαθήσουμε να αναλογιστούμε τι θα δεχτούμε ως λύση στο πρόβλημά μας. Δεν πρέπει να συγχέουμε την έξοδο με τις επεξεργασίες που απαιτούνται για την παραγωγή της. Σε ένα πρόβλημα όπως πχ υπολόγισε 100 δεκαδικά ψηφία του π οι επεξεργασίες είναι διάφορα πράγματα, αλλά η έξοδος είναι απλά ένα μήνυμα με 100 ψηφία του π.
Η εμφάνιση του τελικού αποτελέσματος από τις επεξεργασίες που προηγήθηκαν και είναι η έξοδός μας γίνεται με την εντολή «Εμφάνισε …» όπου … μπορεί να είναι μια μεταβλητή, μια αριθμητική παράσταση μεταξύ μεταβλητών, κάποιο μήνυμα ή και συνδυασμός τους (π.χ. «Εμφάνισε "Η λύση είναι το" x+y»).

Ας επιστρέψουμε στο παράδειγμά μας για να δούμε τι θα δίναμε σε αυτό ως έξοδο. Ξαναβλέποντας την εκφώνηση λέει «ένα πρόγραμμα για τον υπολογισμό του βαθμού πτυχίου των αποφοίτων του». Οπότε η έξοδος που θα δώσουμε θα είναι ένα μήνυμα που θα περιλαμβάνει τον βαθμό πτυχίου τον οποίο πριν έχουμε υπολογίσει με κάποιον τρόπο.

Παραγωγή της εξόδου με κατάλληλες επεξεργασίες

Το να βρούμε τι τελικά θα εμφανίσουμε ως την έξοδο για τον αλγόριθμό μας είναι πολύ πιο εύκολο από το να καθορίσουμε όλες εκείνες τις επεξεργασίες πάνω στα δεδομένα που θα μας δώσουν το αποτέλεσμα που τελικά θα θεωρήσουμε ως την επιθυμητή έξοδο.

Υπάρχουν προβλήματα όπου το πώς θα παράγουμε την έξοδο αξιοποιώντας τις εισόδους είναι κάτι απλό. Πολλές φορές όμως, θα μας ζητείται να κάνουμε κάτι για πολλούς διαφορετικούς τρόπους και το μυαλό μας θα μπερδεύεται με το πλήθος τους. Σε τέτοιες περιπτώσεις χρησιμοποιούμε την τεχνική της **ειδίκευσης**.

Σύμφωνα με αυτή, αντί να δούμε όλες τις περιπτώσεις σαν μια ολότητα για να βρούμε τις επεξεργασίες που πρέπει να εφαρμόσουμε, κοιτάμε μόνο μερικές επιλεγμένες περιπτώσεις. Στο παράδειγμά μας δεν υπάρχει λόγος να σκεφτούμε εξαρχής πως θα βρούμε το βαθμό πτυχίου όλων φοιτητών ή να μπερδευτούμε με γενικές παρατηρήσεις για διάφορους βαθμούς μαθημάτων. Μια απλή περίπτωση αρκεί. Φυσικά, αυτή η απλή περίπτωση ενδέχεται να μην καλύπτει απρόσμενες εισόδους και καταστάσεις. Γι' αυτό είναι πάντα πολύ σημαντικό να αφιερώνουμε χρόνο για να προβλέψουμε τέτοιες καταστάσεις και να τις χειριζόμαστε ικανοποιητικά μέσα στον αλγόριθμό μας. Τέτοια παραδείγματα θα δείτε στα προβλήματα στο τέλος του οδηγού.

Επιστρέφουμε στο παράδειγμά μας και παίρνουμε μια συγκεκριμένη περίπτωση. Έστω λοιπόν, ότι ένα φοιτητής έχει ένα 10, τρία 8 και ένα 5. Ο πίνακας vathmoi θα είναι κάπως έτσι:

syn

10	8	8	8	5

Η έξοδος εδώ θα ήταν να το (10+8+8+8+8+5)/5. Τι κάναμε δηλαδή; Προσθέσαμε τα περιεχόμενα όλων των κελιών του πίνακα και διαιρέσαμε με το πλήθος τους. Γενικεύοντας αυτή την περίπτωση βλέπουμε ότι αυτό που πρέπει να κάνουμε για να πάρουμε την έξοδο είναι να προσθέσουμε τα δεδομένα μας και να τα διαιρέσουμε με το πλήθος τους.

Έχοντας δει τι θέλουμε ως έξοδο και πώς αυτή παράγεται μπορούμε να οδηγηθούμε και στη διατύπωση κάποιων αρχικών, γενικών βημάτων για τη λύση του προβλήματος:

1. Πρόσθεσε τα περιεχόμενα των κελιών του πίνακα vathmoi
2. Εμφάνισε το αποτέλεσμα της διαίρεσης του προηγούμενου αθροίσματος με το 5
3. Επανάλαβε τα προηγούμενα βήματα μέχρι να έχουν επεξεργαστεί οι βαθμοί όλων των υποψηφίων

Το παραπάνω είναι ένας αλγόριθμος κατάλληλος για να εκτελεστεί από άνθρωπο. Παράγεται εύκολα αφού είναι πολύ κοντά στη φυσική μας διαίσθηση για τον τρόπο λύσης του προβλήματος, αλλά δεν αποτελεί απλό αλγόριθμο αφού περιέχει εντολές που δεν είναι αναγνωρίσιμες από υπολογιστή . Όμως, αποτελεί έναν πολύ καλό **αλγόριθμο ανώτερου επιπέδου** που με το σωστό ορισμό διαδικασιών μπορεί να γίνει εκτελέσιμος από υπολογιστή (όλα αυτά τα ζητήματα θα τα δούμε εκτενέστερα σε λίγο).

Μορφή επιτρεπτών εντολών σε έναν αλγόριθμο για υπολογιστή

Όλες οι εντολές που έχουμε στα χέρια μας (αν όχι όλες, η συντριπτική τους πλειοψηφία) είναι για την επεξεργασία των δεδομένων του αλγορίθμου. Άρα αναμένουμε να είναι εντολές που θα μας βοηθούν να κάνουμε κάποιες εργασίες πάνω σε μεταβλητές και πίνακες τύπου ακεραίου, πραγματικού, χαρακτήρα ή συμβολοσειράς. Ας το δούμε λοιπόν αναλυτικότερα.

Γενικά σε μια μεταβλητή οποιουδήποτε τύπου μπορούμε να κάνουμε τα εξής:

1. Να της δώσουμε μια τιμή
2. Να αλλάξουμε την τιμή της
3. Να της αναθέσουμε την τιμή μιας άλλης μεταβλητής του ίδιου τύπου με αυτή

Όλες αυτές οι εντολές πρέπει να γράφονται με πολύ μεγάλη λεπτομέρεια. Για παράδειγμα «Ανάθεσε στην μεταβλητή met1 την τιμή της μεταβλητής met2».

Εντολές ειδικά για τους αριθμούς:

1. Όλες οι γνωστές μαθηματικές πράξεις (πρόσθεση, αφαίρεση, πολλαπλασιασμός, διαίρεση, ύψωση σε δύναμη). Όπως και στα μαθηματικά είναι διμελής πράξεις, δηλαδή χρειαζόμαστε μεταβλητές που παίρνουν μέρος στις πράξεις και μία που λαμβάνει την τιμή της απαρίθμησης της παράστασης. Για παράδειγμα θα λέγαμε «Πρόσθεσε στη μεταβλητή met1 την μεταβλητή met2, αφαίρεσε την met3, υπολόγισε το παραγοντικό και βάλε στο αποτέλεσμα στην met4» ή και με εξίσωση «met4=(met1+met2-met3)!»
2. Εύρεση απόλυτης τιμής, τετραγωνικής ρίζας, τριγωνομετρικών συναρτήσεων, λογαρίθμου, ακεραίου μέρους κλπ

Ειδικά για τους χαρακτήρες:

1. Να ελέγξουμε αν είναι αλφαριθμητικός, ψηφίο, πεζό ή κεφαλαίο γράμμα

Ειδικά για τις συμβολοσειρές:

1. Να αντιγράψουμε μια συμβολοσειρά, ολόκληρη ή ένα μέρος της, σε μια άλλη συμβολοσειρά. Το περιεχόμενο της παλιάς συμβολοσειράς διαγράφεται.
2. Να αντιγράψουμε στο τέλος μιας συμβολοσειράς μια άλλη συμβολοσειρά, ολόκληρη ή ένα μέρος της
3. Να συγκρίνουμε δύο συμβολοσειρές για να δούμε ποια είναι «μεγαλύτερη». Η σύγκριση γίνεται χαρακτήρα-χαρακτήρα και όποιο γράμμα βρίσκεται πιο κοντά στο ω είναι μεγαλύτερο
4. Να ψάξουμε για έναν συγκεκριμένο χαρακτήρα ή μια συμβολοσειρά μέσα σε μια συμβολοσειρά. Μετά μπορούμε να ορίσουμε μια μεταβλητή που θα κρατάει την θέση του χαρακτήρα ή την αρχή της συμβολοσειράς. Μέσω αυτής της μεταβλητής θα μπορούμε να κάνουμε επεξεργασίες σε συγκεκριμένες θέσεις της αρχικής συμβολοσειράς (για την πλήρη κατανόηση αυτού που περιγράφεται καλύτερα να ξαναδιαβάσετε αυτό το κομμάτι μετά από μια εισαγωγή στους δείκτες της C)
5. Να μάθουμε το μήκος μιας συμβολοσειράς. Αν θέλουμε μπορούμε να καταχωρήσουμε αυτή την τιμή σε μια μεταβλητή

Όλα όσα περιγράφτηκαν τώρα είναι διαδικασίες (ή συναρτήσεις όπως θα τις συναρτήσετε στη C). Θα κάνουμε αναφορά σε αυτές παρακάτω.

Σε έναν πίνακα οι επεξεργασίες που μπορούν να γίνουν είναι:

1. Να καταχωρήσουμε μια συγκεκριμένη τιμή ή την τιμή μιας μεταβλητής σε όποια θέση του επιθυμούμε
2. Να ανακτήσουμε το περιεχόμενο οποιασδήποτε θέσης του
3. Να ταξινομήσουμε τα περιεχόμενά του (φθίνουσα ή αύξουσα ταξινόμηση)

Παρόλο που αυτό το ρεπερτόριο εντολών μοιάζει πάρα πολύ περιορισμένο, μας δίνει τη δυνατότητα να λύσουμε έναν τεράστιο αριθμό προβλημάτων διαφόρων μορφών. Όλο το μυστικό είναι η σωστή αναπαράσταση των δεδομένων (βήμα 2). Αν αυτή γίνει σωστά τότε αυτές οι απλές επεξεργασίες μας δίνουν επαρκέστατες δυνατότητες.

Συγκλονιστικό, ε; :-D

Αλγοριθμικές δομές
(ακολουθία, επιλογή, επανάληψη)

Ο αλγόριθμος όπως έχουμε ήδη πει είναι μια σειρά από βήματα. Όμως παρόλο που μπορεί να αισθανόμαστε ότι αφού είναι μια σειρά από βήματα η εκτέλεσή τους πρέπει να γίνει χωρίς επαναλήψεις ή ότι πρέπει να εκτελεστούν όλα, αυτό δεν ισχύει σε καμία περίπτωση.

Πολλές φορές θα χρειαστεί κάποια βήματα να εκτελεστούν υπό συγκεκριμένες συνθήκες ή να εκτελεστούν πάνω από μια φορά. Ενώ κάποιες άλλες θα είναι αρκετό να εκτελεστούν όπως είναι σημειωμένα και να τελειώνουμε με ό,τι λένε. Το πρώτο θα το ονομάσουμε επιλογή, το δεύτερο επανάληψη και το τρίτο ακολουθία.

Για να δούμε κάποια συγκεκριμένα παραδείγματα όσων λέμε:

Έστω ότι έχουμε μπροστά μας τρία κουτιά και μέσα έχουμε τοποθετήσει μια μπάλα διαφορετικού χρώματος στο καθένα (κίτρινη, κόκκινη, μπλε). Θέλουμε να βρούμε, με όσες προσπάθειες χρειαστεί (το πολύ τρεις, αλλιώς κάτι δεν πάει καλά με τις μπάλες!) την κόκκινη μπάλα. Οπότε πρέπει να κάνουμε τα εξής (ο παρακάτω αλγόριθμος δεν είναι έτοιμος για υπολογιστή, είναι όπως θα δούμε ένας αλγόριθμος «ανώτερου επιπέδου»):

1. Ενόσω δεν έχουμε ανοίξει και τα τρία κουτιά
2. Ανοίγουμε το πρώτο κουτί που δεν έχουμε ήδη ανοίξει
3. Αν η μπάλα είναι κόκκινη
4. Σταμάτησε
5. Τέλος της επιλογής
6. Τέλος της επανάληψης

Στο βήμα 2 θέλαμε κάτι να γίνει υπό συγκεκριμένες συνθήκες (αν έχουμε βρει την κόκκινη μπάλα). Στο βήμα 1 θέλαμε κάποια προηγούμενα βήματα να εκτελεστούν πάνω από μια φορά μέχρι μια συνθήκη τερματισμού (ενόσω δεν έχουν ανοιχτεί όλα τα κουτιά).

Τώρα που έχουμε μια επαφή με την πρακτική εφαρμογή των αλγοριθμικών δομών ας τις δούμε από πιο κοντά:

1. Ακολουθία: Η εκτέλεση των βημάτων θα γίνει όπως ακριβώς τα έχουμε γράψει, δεν θα παραληφθεί τίποτα και όλα θα εκτελεστούν ακριβώς μία φορά. Είναι θα λέγαμε, ο standard τρόπος γραφής.

2. Επιλογή: Κάποια βήματα θα εκτελεστούν μόνο αν μια συνθήκη είναι αληθής, αλλιώς είτε θα εκτελεστούν κάποια άλλα είτε θα συνεχίσουμε με τα υπόλοιπα βήματα παραλείποντας όσα είναι εντός της αλγοριθμικής δομής της επιλογής.

   Η επιλογή έχει δύο μορφές (στην πραγματικότητα είναι περισσότερες, αλλά θα τις δείτε στο κυρίως μέρος του μαθήματος) :

   α. Αν συνθήκη τότε
      βήματα
      τέλος της επιλογής

   β. Αν συνθήκη τότε
      βήματα
      αλλιώς
      βήματα
      τέλος της επιλογής

   Η συνθήκη είναι μια λογική πρόταση, δηλαδή μια διατύπωση που μπορεί να είναι είτε αληθής είτε ψευδής. Σε έναν αλγόριθμο για υπολογιστές όλες οι λογικές προτάσεις βασίζονται τελικά, σε συγκρίσεις μεταξύ τιμών μεταβλητών. Μια συνθήκη δηλαδή θα ήταν «η τιμή της μεταβλητής i είναι μεγαλύτερη αυτής της μεταβλητής j» ή και «στη θέση 2 του πίνακα Α υπάρχει η τιμή 5». Με βάση την αναπαράσταση που έχουμε κάνει στα δεδομένα μας (βήμα 2) αυτές οι συγκρίσεις θα έχουν κάποιο πρακτικό νόημα.

Στην πρώτη μορφή, αν η συνθήκη είναι αληθής εκτελούνται τα βήματα μέχρι το τέλος της επιλογής και μετά κανονικά τα παρακάτω. Αν είναι ψευδής τότε αυτόματα η εκτέλεση των βημάτων συνεχίζεται με το αμέσως επόμενο μετά το τέλος της επιλογής.

Στη δεύτερη μορφή, αν η συνθήκη είναι αληθής εκτελούνται τα βήματα μέχρι το αλλιώς και η εκτέλεση συνεχίζεται με το αμέσως επόμενο βήμα μετά το τέλος της επιλογής. Αν είναι ψευδής τότε εκτελούνται τα βήματα μετά το αλλιώς και μέχρι το τέλος της επιλογής, μετά η εκτέλεση συνεχίζεται κανονικά.

3. Επανάληψη: Κάποια βήματα θα εκτελεστούν μέχρι μια συνθήκη να γίνει ψευδής. Η μορφή επανάληψης που μας ενδιαφέρει είναι αυτή

    Όσο συνθήκη
    βήματα
    τέλος της επανάληψης

    Όσο η συνθήκη είναι αληθής θα εκτελούνται τα βήματα. Μόλις γίνει ψευδής θα συνεχιστεί η εκτέλεση με το αμέσως επόμενο βήμα μετά το τέλος της επανάληψης.

Φυσικά, είναι απολύτως εφικτό και επιτρεπτό μια αλγοριθμική δομή να περιέχει άλλες αλγοριθμικές δομές (π.χ. μια επανάληψη να περιέχει μία επιλογή και αυτή δύο επαναλήψεις).

Η ανάγκη να χρησιμοποιήσουμε στον αλγόριθμό μας επιλογές ή επαναλήψεις είναι σχετικά προφανής. Κατά την επεξεργασία του προβλήματος θα αισθανθούμε ότι θέλουμε κάποιες ενέργειες να γίνουν μόνο υπό συγκεκριμένες συνθήκες, αυτό θα είναι σημάδι για επιλογή. Ενώ αν δούμε ότι κάποιες ενέργειες πρέπει να γίνονται συνέχεια μέχρι να συμβεί κάτι αυτό θα είναι σημάδι επανάληψης.

Για τη σχηματοποίηση των αλγοριθμικών δομών της επιλογής και της επανάληψης μπορούμε να σκεφτόμαστε ότι είναι κουτιά με μια ταμπέλα απέξω. Η ταμπέλα είναι η συνθήκη που πρέπει να είναι αληθής για να τα ανοίξουμε και να δούμε ποια βήματα περιέχουν μέσα. Αν είναι ψευδής τότε αφήνουμε το κουτί και τα βήματα που περιέχει στην ησυχία του και συνεχίζουμε με το πρώτο βήμα εκτός του «κουτιού».

Γενική σκιαγράφηση αλγορίθμου σε ανώτερο επίπεδο

Αυτό το βήμα, παρόλο που τοποθετήθηκε προ-τελευταίο, είναι ουσιαστικά ένα από τα πρώτα βήματα που πρέπει να ακολουθήσουμε κατά την επεξεργασία ενός προβλήματος. Είναι βασικά, το βήμα που μας δίνει την πρώτη και πιο ανθρώπινα κατανοητή μορφή αλγορίθμου.

Ως τώρα επιμείναμε πολύ στο να ακολουθούμε κάποιες αυστηρές αρχές ώστε να δημιουργούμε αλγορίθμους κατάλληλους για υπολογιστή. Θα έλεγα ότι αυτό μπορούμε εν μέρει να το ξεχάσουμε (καλά, τότε τι μας έλεγες επί 28 σελίδες;;; καλά έκανα λοιπόν και μίλαγα με το διπλανό μου :-D) !

Μπορούμε να δημιουργήσουμε έναν αλγόριθμο «ανώτερου επιπέδου» με τη χρήση διαδικασιών (θα τις δούμε εκτενώς αμέσως μετά) ώστε να είναι πολύ πιο κοντά στην πραγματική φύση του προβλήματος που επιλύουμε. Έπειτα βέβαια, πρέπει αυτές τις διαδικασίες να τις ορίσουμε με τέτοια βήματα ώστε να είναι κατάλληλες για εκτέλεση από υπολογιστή (οπότε δεν κάνατε καλά που μιλάγατε με το διπλανό σας!).

Επιστρέφουμε στο πρόβλημα υπολογισμού του βαθμού πτυχίου. Για να φτιάξουμε έναν αλγόριθμο ανώτερου επιπέδου δεν χρειάζεται να ζοριστούμε πολύ, απλά σκεφτόμαστε πώς θα λύναμε εμείς το πρόβλημα. Όπως είδαμε και πριν ένας αλγόριθμος ανώτερου επιπέδου είναι:

1. Ενόσω δεν έχουμε επεξεργαστεί όλους τους φοιτητές
2. Πρόσθεσε τα περιεχόμενα των κελιών του πίνακα vathmoi και τοποθέτησε το αποτέλεσμα σε μεταβλητή athrisma
3. Εμφάνισε athrisma/5
4. Τέλος επανάληψης

Αυτό είναι ένας πολύ ωραίος αλγόριθμος, αρκεί να βρούμε ποιες διαδικασίες χρησιμοποιήσαμε και να τις ορίσουμε με βήματα κατάλληλα για εκτέλεση από υπολογιστή.

Με μια ματιά βλέπουμε ότι υπάρχουν κάποιες εντολές που δεν είναι στο ρεπερτόριο που περιγράψαμε πριν. Αυτές είναι η «πρόσθεσε τα περιεχόμενα τον κελιών ενός πίνακα και βάλε το αποτέλεσμα σε μια μεταβλητή» και η «έλεγχος αν έχουν επεξεργαστεί όλοι οι φοιτητές». Αυτές τις διαδικασίες θα πρέπει μετά με τον τρόπο που θα περιγράψουμε παρακάτω, να τις ορίσουμε εκτενέστερα. Τον ακριβή ορισμό τους θα τον συζητήσουμε στο μάθημα ως μια άμεση εξάσκηση.

Γενικά, κάθε πρόβλημα αρχικά το προσεγγίζουμε από ένα **ανώτερο επίπεδο**, βλέπουμε πως θα το λύναμε εμείς και όσες εντολές δεν είναι κατάλληλες για υπολογιστή τις κάνουμε διαδικασίες και τις ορίζουμε κατάλληλα. Με αυτό τον τρόπο δημιουργούμε μια εύκολα κατανοητή μορφή του αλγορίθμου μας που έπειτα θα την αποσαφηνίσουμε κατάλληλα ώστε να είναι εκτελέσιμη από υπολογιστή.

Ας δούμε ένα ακόμα παράδειγμα για την καλύτερη κατανόηση του τρόπου παραγωγής ενός αλγορίθμου ανώτερου επιπέδου

Πρόβλημα: Υπολογίστε τους πρώτους αριθμούς από το 1 μέχρι και έναν δεδομένο αριθμό m. Μετά εκτυπώστε όσους απ’ αυτούς είναι μικρότεροι του m div 2

1. Δώσε ένα δεδομένο του τύπου ακέραιος, αντιστοίχισε το με το όνομα m
2. Υπολόγισε τους πρώτους από το 1 μέχρι το m και τοποθέτησε τους σε πίνακα ακεραίων protoi μεγέθους m
3. Εκτύπωσε όσους ακεραίους από τον πίνακα protoi είναι μικρότεροι του m div 2

Οι διαδικασίες που ορίσαμε εδώ είναι «Υπολόγισε τους πρώτους μεταξύ δύο δεδομένων τιμών» και «Εκτύπωσε το περιεχόμενο των κελιών ενός πίνακα που πληρούν τη συνθήκη να είναι μικρότεροι του m div 2». Με τη χρήση τους ο αρχικός αλγόριθμος γίνεται πολύ εύκολο να παραχθεί και να κατανοηθεί. Απλά πρέπει να σκεφτούμε πολύ γενικά και σαν … άνθρωποι :D

Θα δείτε ότι αυτό τον γενικό αλγόριθμο θα τον τοποθετείτε μέσα σε μια συνάρτηση που τη λένε main, αλλά αυτό δε χρειάζεται να σας προβληματίζει καθόλου επί της παρούσης.

Διαδικασίες για την αποσαφήνιση του αλγορίθμου

Αρχικά θα ορίσουμε μια πολύ χρήσιμη τεχνική, αυτή της στοιχειοποίησης. Σύμφωνα με αυτή ένα πρόβλημα διασπάται σε μικρότερα υπο-προβλήματα που είναι πιο εύκολο να επιλυθούν. Μετά συνθέτουμε τις επιμέρους λύσεις για να λύσουμε το αρχικό πρόβλημα. Βασικό κομμάτι της στοιχειοποίησης είναι τα στοιχεία, δηλαδή μέρη του αλγορίθμου τα οποία κάνουν μια συγκεκριμένη εργασία ανεξάρτητα από ποιου προβλήματος μέρος είναι.

Για παράδειγμα ένα στοιχείο είναι ο υπολογισμός των τέλειων αριθμών σε ένα δεδομένο διάστημα. Ο αλγόριθμός μας μπορεί να αξιοποιεί αυτό το στοιχείο όπως θέλει, απλά το καλεί και χειρίζεται τις εξόδους που παράγει. Κάποιος άλλος αλγόριθμος μπορεί να το αξιοποιεί διαφορετικά. Εν τέλει, το μόνο που μας ενδιαφέρει για το στοιχείο είναι η εργασία που κάνει ενώ μας δίνεται και η δυνατότητα επαναχρησιμοποίησης του.

Αυτά τα στοιχεία θα τα ονομάσουμε διαδικασίες (στη C θα τα ονομάσετε συναρτήσεις και θα δείτε ότι έχουν κάποιες διαφορές από αυτό που θα περιγράψουμε παρακάτω). Μια διαδικασία λοιπόν, είναι μια εργασία που θέλουμε να κάνει ο αλγόριθμός μας. Μπορεί να είναι πολύπλοκη, αλλά καλό είναι αντί για μια πολύπλοκη εργασία να ορίζουμε περισσότερες και απλούστερες. Στην ιδανική περίπτωση κάθε διαδικασία μας είναι στοιχειώδης, δηλαδή κάνει ακριβώς μία δουλειά.

Στο προηγούμενο κεφάλαιο μιλήσαμε για αλγορίθμους ανώτερου επιπέδου. Αυτοί οι αλγόριθμοι περιέχουν διαδικασίες, δηλαδή είναι μια πρώτη εικόνα των εργασιών που πρέπει να γίνουν και του τρόπου που αυτές συνδέονται μεταξύ τους ώστε να επιλυθεί το πρόβλημα. Άρα η παραγωγή ενός τέτοιου αλγορίθμου όπως ήδη είπαμε, απαιτεί από μας να σκεφτούμε γενικά και χωρίς ιδιαίτερες λεπτομέρειες ποιες είναι οι εργασίες που απαιτεί το πρόβλημά μας. Μετά κάθε μία την κάνουμε διαδικασία όπως θα δούμε στη συνέχεια.

Μια διαδικασία είναι ένας υπο-αλγόριθμος, ένα πακετάρισμα εντολών θα λέγαμε, γι' αυτό έχει τα εξής χαρακτηριστικά:

1. Ένα όνομα, που πρέπει να είναι επεξηγηματικό της εργασίας που εκτελεί
2. Εισόδους
3. Εντολές
4. Εξόδους

Ο αλγόριθμος ανώτερου επιπέδου το μόνο που χρειάζεται να ξέρει είναι το όνομα της διαδικασίας και τι κάνει. Μετά όπου πρέπει την καλεί, της δίνει εισόδους και αξιοποιεί τις εξόδους που αυτή παράγει. Φυσικά, στο τέλος πρέπει να οριστούν τα βήματα της διαδικασίας ώστε ο αλγόριθμος που παράγουν να είναι κατάλληλος για υπολογιστή (δεν ξεφεύγουμε απ' αυτό :P).

Ας δούμε τώρα πως μπορούμε να καλούμε μια διαδικασία (ο τρόπος που θα περιγράψουμε είναι πιο απλοϊκός απ' αυτόν των συναρτήσεων της C, αλλά η ουσία είναι η ίδια). Η κλήση γίνεται σε ένα βήμα του αλγορίθμου και έχει την εξής μορφή:

5. Όνομα: Υπολογισμός περιττών στο διάστημα [1,m]
   Είσοδοι: 1, m (όπου m μια μεταβλητή που έχουμε ορίσει πριν)
   Έξοδοι: perittoi (μονοδιάστατος πίνακας που έχουμε ορίσει πριν)

Το παραπάνω καλεί μια διαδικασία με όνομα «Υπολογισμός περιττών». Δίνει εισόδους το 1 και το m, ενώ παίρνει ως έξοδο έναν πίνακα perittoi. Μην μπερδεύεστε με το που θα ξέρουμε ποιες εισόδους παίρνει η διαδικασία και τι εξόδους αποδίδει, αυτά τα ζητήματα τα καθορίζουμε εμείς όπως μας βολεύει ☺ Φυσικά, μετά την κλήση μιας διαδικασίας τα επόμενα βήματα του αλγορίθμου μπορούν να αξιοποιούν τις εξόδους που επιστρέφει υποθέτοντας ότι περιέχουν αυτό το οποίο υπόσχεται η διαδικαδία να κάνει. Στο παράδειγμά μας δηλαδή, θα γεμίσει τα κελιά του πίνακα perittoi με τους περιττούς αριθμούς από το 1 μέχρι το m. Στα επόμενα βήματα του αλγορίθμου μπορούμε να αξιοποιήσουμε τον πίνακα perittoi όπως θέλουμε ξέροντας ότι μετά την κλήση της διαδικασίας θα περιέχει τους περιττούς από το 1 μέχρι το m.

Ας δούμε τώρα πώς θα ορίζαμε την προηγούμενη διαδικασία. Πρώτα απ' όλα ο ορισμός της γίνεται οπουδήποτε, δεν μας ενδιαφέρει πού θα γράψουμε τα βήματά της, ο υπολογιστής θα τα βρει αρκεί να είναι σε έναν συνεχόμενο και οριοθετημένο χώρο ;-) Συνήθως τον τοποθετούμε μετά το τέλος του αλγορίθμου ανώτερου επιπέδου που έχουμε γράψει.

Υπολογισμός περιττών(ακέραιος a, ακέραιος b)

1. Όρισε μια μεταβλητή τύπου ακεραίου με όνομα i
2. Όρισε μια μεταβλητή τύπου ακεραίου με το όνομα j και αρχική τιμή το 0
3. Όρισε ένα μονοδιάστατου πίνακα ακεραίων με όνομα pinakas και μέγεθος b-a
4. i=a
5. Ενόσω το i δεν είναι ίσο με το b
6. Αν i mod 2 = 1
7. Τοποθέτησε στη θέση j του πίνακα pinakas το i
8. j=j+1
9. Τέλος επιλογής
10. i=i+1
11. Τέλος επανάληψης
12. Επίστρεψε pinakas

Οι μεταβλητές που υπάρχουν δίπλα στο όνομα της διαδικασίας ονομάζονται ορίσματα. Οι τιμές που παίρνουν προέρχονται από μία 1-προς-1 αντιστοίχηση με τις εισόδους που καλείται. Στο παράδειγμά μας a=1 και b=m. Τα ορίσματα είναι κανονικά δηλωμένες μεταβλητές που μπορεί να χειριστεί η διαδικασία. Είναι ουσιαστικά ο τρόπος με τον οποία επικοινωνεί με τον έξω κόσμο και παραμετροποιεί τις εξόδους που παρέχει.

Στο τέλος υπάρχει ένα βήμα που λέει «Επίστρεψε». Όσες μεταβλητές τοποθετούνται μετά από αυτή την εντολή επιστρέφουν τις τιμές τους μέσω μιας 1-προς-1 αντιστοίχησης στις μεταβλητές που ορίστηκαν ως έξοδοι κατά την κλήση. Στο παράδειγμά μας ο πίνακας perittoi παίρνει τα περιεχόμενα του pinakas.

Άρα μια διαδικασία δέχεται εισόδους, κάνει επεξεργασίες και παράγει εξόδους που αξιοποιούνται κατάλληλα απ' τον υπόλοιπο αλγόριθμο.

Σημειώνεται ότι η αντιστοίχηση που γίνεται αφορά μόνο τιμές, δηλαδή η μεταβλητή b και η m είναι δύο ανεξάρτητες μεταβλητές που απλά, έχουν αρχικά την ίδια τιμή. Αν θέλουμε η διαδικασία να αλλάζει το περιεχόμενο της m τότε πρέπει να την τοποθετήσουμε και ως έξοδο (στη C αυτό θα γίνεται μέσω των λεγόμενων δεικτών).

Η τεχνική της στοιχειοποίησης είναι πραγματικά, μια πολύ χρήσιμη μέθοδος για τη δημιουργία κατανοητών προγραμμάτων. Παράλληλα έχει μια σειρά άλλων πλεονεκτημάτων που θα εκτιμήσετε ιδιαίτερα τα επόμενα εξάμηνα (απόκρυψη πληροφορίας). Πάντως, σας συμβουλεύω πάντα να σκέφτεστε με αυτή τη μέθοδο γιατί κυριολεκτικά μας λύνει τα χέρια κατά την φάση της συγγραφής του κώδικα.

Και ναι, φτάσαμε στο τέλος αυτού του μαραθώνιου κεφαλαίου! Μάθαμε πολλά για τη δημιουργία απλών αλγορίθμων και πολλές χρήσιμες τεχνικές. Αυτές οι γενικές αρχές που διατυπώσαμε ελπίζω να σας βοηθήσουν σε όλη σας την προγραμματιστική ζωή ☺

Ακολουθεί μια μικρή σύνδεση όσων είπαμε με το μάθημα της Εισαγωγής στον Προγραμματισμό. Μετά παρουσιάζονται υπολογιστικά προβλήματα για να δούμε στην πράξη πως εφαρμόζονται όσα είπαμε για είδη προβλημάτων που θα αντιμετωπίσετε στα πλαίσια του διαδικαστικού προγραμματισμού.

Σας αξίζει ένα μεγάλο μπράβο που φτάσατε ως εδώ!!

Προγραμματισμός και αλγόριθμοι

Σε όλο αυτόν τον οδηγό μιλήσαμε, και θα συνεχίσουμε να μιλάμε, για αλγορίθμους. Όμως, τόσο στα πλαίσια του μαθήματος όσο και στα πλαίσια του επαγγέλματός σας συχνά θα κληθείτε να παράγετε προγράμματα σε κάποια συγκεκριμένη γλώσσα προγραμματισμού. Οπότε εύλογα δημιουργείται το ερώτημα “Ποια η σχέση των αλγορίθμων που διατυπώσαμε στον οδηγό με ένα πρόγραμμα που θα έλυνε το ίδιο πρόβλημα;”.

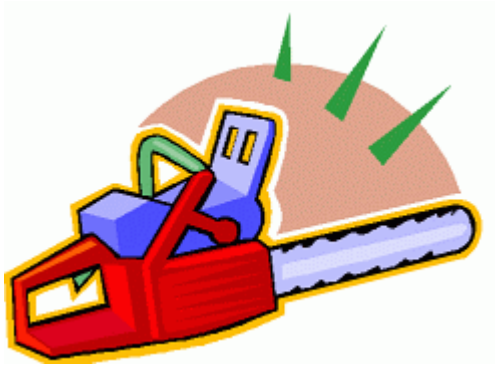
Στον οδηγό είδαμε κάποιες τεχνικές χρήσιμες για τη διατύπωση οποιοδήποτε αλγορίθμου (όπως η τεχνική της ειδίκευσης). Επειδή όμως θέλαμε να εστιάσουμε στους αλγορίθμους για υπολογιστές θεωρήσαμε ως δεδομένο ότι ο εκτελεστής του αλγορίθμου θα είναι ένας υπολογιστής. Επίσης, δεχθήκαμε ότι θα επικοινωνήσουμε μαζί του χρησιμοποιώντας ένα σύνολο από εντολές-εργαλεία και κάποιες παραδοχές για το τι μπορεί και τι δεν μπορεί να κατανοήσει. Η πραγματικότητα είναι ότι ακολουθήσαμε αυτόν τον τρόπο αντιμετώπισης γιατί υπήρξε μια ανεπαίσθητη μεν, σαφής δε, εστίαση στις δυνατότητες που μας παρέχει η γλώσσα C και των προγραμμάτων που μπορούμε να αναπτύξουμε με τη βοήθειά της.

Αυτό σημαίνει ότι οι αλγόριθμοι που δημιουργήσαμε και όσοι κανόνες διατυπώσαμε ήταν κομμένοι και ραμμένοι για τη γλώσσα C. Αρκετά απ' όσα είπαμε, με μερικές ή καθόλου μετατροπές, ισχύουν και πέρα από τα όρια της C. Σε κάθε περίπτωση, οι αλγόριθμοι που μάθαμε να δημιουργούμε έχουν μια σαφή και άμεση σχέση με ένα πρόγραμμα C. Θα δείτε ότι πολύ σύντομα θα βρείτε και μόνοι σας αυτή την αντιστοίχηση και πλέον, δεν θα υπάρχει κάποιος λόγος να περνάτε από το ενδιάμεσο στάδιο της διατύπωσης των αλγορίθμων που περιγράψαμε για να δημιουργήσετε ένα πρόγραμμα σε C.

Με λίγα λόγια, αυτό που τελικά ευελπιστεί να σας προσφέρει αυτός ο οδηγός είναι μια εκτενής και όσο το δυνατόν πιο ανώδυνη εισαγωγή στη δημιουργία προγραμμάτων με C. Παράλληλα, σας προσφέρει και μερικά χρήσιμα tips και βήματα που μπορείτε να ακολουθήσετε για να αντεπεξέλθετε στη λύση προβλημάτων με τη βοήθεια υπολογιστή, προσπαθώντας να μην εστιάσει σε κάποιο συγκεκριμένο είδος προβλήματος.

Είναι αλήθεια ότι η έννοια του αλγορίθμου χρησιμοποιήθηκε αρκετά πέρα από τα όρια της, ίσως και καταχρηστικά, μέσα στον οδηγό. Όπως και να'χει, ακόμα κι αν απέτυχε στο να σας δείξει πως να διατυπώνετε σωστούς αλγορίθμους (άλλωστε θα το μάθετε και αυτό αργότερα στη σχολή), τουλάχιστον επιχείρησε να κάνει μια εισαγωγή στη διατύπωση “αλγορίθμων” με στόχο την κατανόηση των βασικών αρχών της γλώσσας C.

Στο κεφάλαιο που ακολουθεί θα δούμε την εφαρμογή των βημάτων και των τεχνικών που αναφέραμε ως τώρα για τη λύση κάποιων κλασικών διαδικαστικών προβλημάτων. Σας ενθαρρύνω πριν δείτε τη λύση που παρατίθεται να προσπαθήσετε μόνοι σας να διατυπώσετε έναν αλγόριθμο.



Μ'αυτό θα με κυνηγάει ο κ. Ζησιμόπουλος
άμα μάθει ότι σας είπα ότι στους αλγορίθμους
δηλώνουμε μεταβλητές :P

Διαδικαστικό πρόβλημα 1

Πρόβλημα: ***Υπολογίστε το άθροισμα 5 ακέραιων αριθμών που κάθε φορά τους καθορίζει ο χρήστης.***

Προσέξτε πολύ αυτό το πρόβλημα, περιέχει κάποιες επεξεργασίες που θα κάνετε πολύ συχνά στα προγράμματά σας.

Αρχικά διαβάζουμε καλά το πρόβλημα για να **βρούμε τα δεδομένα** μας. Ρωτάμε «Ποια αντικείμενα θα επεξεργαστώ ώστε να λύσω το πρόβλημα;». Παρατηρούμε λοιπόν, ότι τα δεδομένα μας είναι οι αριθμοί – όροι του αθροίσματος.

Συνεχίζουμε προσπαθώντας να εντοπίσουμε **τη μορφή της λύσης** που θέλουμε να δώσουμε. Παρατηρούμε ότι η μορφή της λύσης είναι ένας ακέραιος αριθμός ως το άθροισμα των ακεραίων αριθμών που έδωσε ο χρήστης.

Για να δούμε τώρα πως θα **αναπαραστήσουμε τα δεδομένα** μας με βάση τους τύπους που έχουμε αναφέρει. Οι αριθμοί προφανώς θα αναπαρασταθούν από έναν ακέραιο.

Αφού είδαμε όλα τα προηγούμενα βήματα είμαστε σε θέση να διατυπώσουμε τα πρώτα βήματα του αλγορίθμου μας που αφορούν την **είσοδο των δεδομένων**. Επειδή τα δεδομένα μας είναι ομοειδή και έχουν κάποια σχέση μεταξύ τους θα τα βάλουμε σε πίνακα 5 θέσεων *. Άρα:

1. Όρισε έναν μονοδιάστατο πίνακα ακεραίων 5 θέσεων με όνομα Arithmoi
2. Όρισε μια μεταβλητή τύπου ακεραίου με όνομα i
3. i=0
4. Όσο i<=4
5. Πάρε τιμή για Arithmoi[i]
6. i=i+1
7. Τέλος επανάληψης

* Τονίζω ξανά, ότι δεν είναι απόλυτα σωστό να ορίσετε έναν πίνακα για τον υπολογισμό αυτού του αθροίσματος. Αυτό το κάνουμε για να γίνει μια συζήτηση για τον τρόπο που χειριζόμαστε έναν πίνακα. Ο σωστός τρόπος είναι με μεταβλητή-συσσωρευτή και θα τον δούμε στο αμέσως μετά..

Στα βήματα 2 έως 7 βλέπετε μια πάρα πολύ χρήσιμη και συχνά χρησιμοποιούμενη τακτική για να επεξεργαζόμαστε τις θέσεις ενός πίνακα. Αντί να γράφουμε «Πάρε τιμή» για κάθε θέση του πίνακα (φανταστείτε να το κάνετε αυτό για πίνακα 1000 θέσεων :D) χρησιμοποιούμε μια επαναληπτική δομή. Η μεταβλητή i είναι ένας μετρητής, χρησιμοποιείται για να κρατάει κάθε στιγμή την τιμή της θέσης του πίνακα που θα επεξεργαστούμε. Γι'αυτό ξεκινάει από 0 (βήμα 3), όσο είναι μέσα στα πλαίσια του πίνακα (δεν έχει υπερβεί το 4 δηλαδή – βήμα 4), κάνουμε τη δουλειά που θέλουμε (βήμα 5) και τελικά αυξάνουμε το i κατά 1 ώστε μετά να προσπελάσουμε την επόμενη θέση του πίνακα (βήμα 6).

Ας σκεφτούμε τώρα τον **αλγόριθμο ανώτερου επιπέδου**. Όπως είπαμε αυτός βγαίνει πολύ εύκολα, αρκεί να σκεφτούμε πολύ γενικά τι θέλουμε να κάνει ο αλγόριθμός μας:

8. Πρόσθεσε τα περιεχόμενα των θέσεων του πίνακα Arithmoi μεταξύ τους και τοποθέτησε το αποτέλεσμα σε μεταβλητή athrisma
9. Εμφάνισε athrisma

Πριν ορίσουμε τη διαδικασία του βήματος 3 θα αφήσουμε κατά μέρος την ολότητα του προβλήματος και θα **ειδικευτούμε** σε μια απλή περίπτωση. Έστω ότι ο πίνακας Arithmoi έχει την παρακάτω μορφή:

5	2	8	5	10

Για να σκεφτούμε λίγο όμως, πώς θα κάνουμε την πρόσθεση αυτών των αριθμών. Πάρτε ένα χαρτί και κάντε πολύ αναλυτικά και με τον πιο απλό δυνατό τρόπο (αναλυτικά και απλά, οι λέξεις κλειδιά όταν σκεφτόμαστε αλγοριθμικά) αυτή την πρόσθεση. Θα κάνατε κάτι σαν το εξής:

0+5=5 5+2=7 7+8=15 15+5=20 20+10=30

Δηλαδή υπολογίζατε κάθε φορά το άθροισμα του τρέχοντος αριθμού με το ως τότε άθροισμα των προηγούμενων αριθμών. Αυτή την απλή τεχνική θα χρησιμοποιήσουμε και στον αλγόριθμό μας. Το άθροισμα των προηγούμενων όρων θα είναι μια μεταβλητή S και κάθε φορά θα προσθέτουμε σε αυτή τον επόμενο όρο. Η αρχική τιμή της θα είναι το 0 αφού αρχικά δεν έχουμε προηγούμενους όρους να βρούμε το άθροισμά τους.

Θα λέγαμε δηλαδή ότι όποτε θέλουμε να υπολογίσουμε άθροισμα στον υπολογιστή χρησιμοποιούμε μια μεταβλητή – συσσωρευτή που σιγά σιγά γεμίζει με τους αριθμούς του αθροίσματος. Μάλιστα, σε κάθε άθροιση δεν χρειαζόμαστε ούτε τους προηγούμενους αριθμούς ούτε τους επόμενους, μόνο τον τρέχοντα. Αυτός είναι ο κυριότερος λόγος για τον οποίο δεν χρειάζεται να χρησιμοποιήσουμε έναν πίνακα για την αποθήκευση των τιμών. Προς το παρόν το αγνοούμε αυτό αφού θα το δούμε εκτενέστερα σε λίγο. Ας δούμε τώρα την τυποποιημένη διαδικασία της άθροισης των στοιχείων του πίνακά μας και παράλληλα να την ορίσουμε κιόλας:

Πρόσθεσε αριθμούς από πίνακα (πίνακας ακεραίων Α) {
1. Όρισε μεταβλητή τύπου ακεραίου με όνομα S
2. Όρισε μεταβλητή τύπου ακεραίου με όνομα i
3. i=0
4. S=0
5. Όσο i<=4
6. S=S+A[i]
7. i=i+1
8. Τέλος επανάληψης
9. Επίστρεψε S

}

Είδατε ότι χρησιμοποιήσαμε τα βήματα που αναφέραμε προηγουμένως για την προσπέλαση όλων των θέσεων ενός πίνακα. Αυτό που παρουσιάζει ιδιαίτερο ενδιαφέρον είναι τα βήματα 4 και 6. Στο βήμα 4 δίνουμε αρχική τιμή στη μεταβλητή – συσσωρευτή το 0, αφού αρχικά δεν έχουμε προσθέσει κανέναν όρο στο άθροισμα. Στο βήμα 6 κάνουμε αυτό που περιγράψαμε, δηλαδή προσθέτουμε στην ως τότε τιμή του S το νέο όρο του αθροίσματος που μόλις προσπελάσαμε.

Σημείωση: Στο βήμα 5 αναφέρετε «Όσο i<=4». Είχαμε πει ότι οι διαδικασίες που ορίζουμε καλό είναι να μη δεσμεύονται από παραδοχές ενός συγκεκριμένου προβλήματος. Εδώ πέρα θέλουμε να δημιουργήσουμε μια συνάρτηση που παίρνει έναν πίνακα ακεραίων και επιστρέφει το άθροισμα των περιεχομένων των κελιών του. Γι’ αυτό δεν πρέπει να περιοριστούμε στη διάσταση που μας δίνει ως δεδομένη το πρόβλημα. Το απόλυτα σωστό λοιπόν θα ήταν να δεχόμαστε ως όρισμα τη διάσταση του πίνακα και να το χρησιμοποιούμε αντί του 4.

Οπότε ολοκληρωμένος ο αλγόριθμος που επιλύει το πρόβλημα:

1. Όρισε έναν μονοδιάστατο πίνακα ακεραίων 5 θέσεων με όνομα Arithmoi
2. Όρισε μια μεταβλητή τύπου ακεραίου με όνομα i
3. Όρισε μια μεταβλητή τύπου ακεραίου με όνομα athrisma
4. i=0
5. Όσο i<=4
6. Πάρε τιμή για Arithmoi[i]
7. i=i+1
8. Τέλος επανάληψης
9. Όνομα: Πρόσθεσε αριθμούς από πίνακα
   Είσοδοι: Arithmoi
   Έξοδοι: athrisma
10. Εμφάνισε athrisma

Πρόσθεσε αριθμούς από πίνακα (πίνακας ακεραίων A) {

1. Όρισε μεταβλητή τύπου ακεραίου με όνομα S
2. Όρισε μεταβλητή τύπου ακεραίου με όνομα i
3. i=0
4. S=0
5. Όσο i<=4
6. S=S+A[i]
7. i=i+1
8. Τέλος επανάληψης
9. Επίστρεψε S

}

Όπως είπαμε και πριν, σε κάθε άθροιση χρειαζόμαστε μόνο τον τρέχοντα αριθμό. Σε κανένα σημείο του προγράμματος δεν χρειάζεται να χρησιμοποιήσουμε ξανά κάποιον αριθμό που μας δόθηκε πριν. Οπότε δεν υπάρχει και κάποιος λόγος να τον αποθηκεύουμε. Άρα, αρκεί να δεχόμαστε κάθε αριθμό τη στιγμή που τον χρειαζόμαστε, να κάνουμε την άθροιση και μετά να τον ξεχνάμε για πάντα. Με βάση αυτές τις παρατηρήσεις έχουμε τον παρακάτω αλγόριθμο για τη λύση του προβλήματος

1. Όρισε μια μεταβλητή τύπου ακεραίου με όνομα trexon
2. Όρισε μια μεταβλητή τύπου ακεραίου με όνομα i
3. Όρισε μια μεταβλητή τύπου ακεραίου με όνομα athrisma
4. i=0
5. athrisma=0
6. Όσο i<=4
7. Πάρε τιμή για trexon
8. athrisma = athrisma + trexon
9. i=i+1
10. Τέλος επανάληψης
11. Εμφάνισε άθροισμα

Είδατε λοιπόν, ότι με αυτή την παρατήρηση που μας επέτρεψε να μην χρησιμοποιήσουμε πίνακα, ο αλγόριθμός μας έγινε τόσο τετριμμένος που δεν υπήρχε λόγος να ορίσουμε κάποια διαδικασία.

Διαδικαστικό πρόβλημα 2

Πρόβλημα: ***Έστω ένας πίνακας που περιέχει θετικούς ακεραίους αριθμούς. Δημιουργήστε μια διαδικασία που δέχεται ως είσοδο έναν τέτοιο πίνακα και το πλήθος των θέσεών του. Εντός της διαδικασίας να διαβάζετε επαναληπτικά έναν αριθμό και να αποφαίνεστε αν υπάρχει ή όχι στον πίνακα. Η διαδικασία να τερματίζεται αν διαβαστεί ως αριθμός το -1***

Αυτό το πρόβλημα έχει ως σκοπό του να σας εισάγει στην αναζήτηση σε πίνακα και στη δημιουργία διαδικασιών ανεξάρτητων από κάποιο συγκεκριμένο πρόγραμμα.

Αρχικά διαβάζουμε καλά το πρόβλημα για να **βρούμε τα δεδομένα** μας. Ρωτάμε «Ποια αντικείμενα θα επεξεργαστώ ώστε να λύσω το πρόβλημα;». Βλέπουμε λοιπόν, ότι τα δεδομένα μας είναι ο πίνακας ακεραίων, το πλήθος των θέσεών του και ο αριθμός που διαβάζουμε για να προσδιορίσουμε αν υπάρχει στον πίνακα.

Συνεχίζουμε προσπαθώντας να εντοπίσουμε **τη μορφή της λύσης** που θέλουμε να δώσουμε. Η λύση θα είναι ουσιαστικά ένα μήνυμα που θα μας λέει αν βρήκαμε ή όχι τον αριθμό μέσα στον πίνακα.

Για να δούμε τώρα πως θα **αναπαραστήσουμε τα δεδομένα** μας με βάση τους τύπους που έχουμε αναφέρει. Δεν χρειάζονται κάποιες ιδιαίτερες αναπαραστάσεις, τα δεδομένα είναι ήδη γνωστών και συμβατών τύπων.

Σχετικά με την **είσοδο των δεδομένων**, επειδή θέλουμε να δημιουργήσουμε μια διαδικασία και όχι ένα ολόκληρο πρόγραμμα, πρέπει να δούμε ποια δεδομένα τα δεχόμαστε από έναν «εξωτερικό χώρο» και ποια τα διαβάζουμε εντός της διαδικασίας. Σύμφωνα με την εκφώνηση τον πίνακα και το μέγεθός του τα δεχόμαστε έτοιμα ως εισόδους, ενώ τον αριθμό κάθε φορά τον διαβάζουμε εντός της διαδικασίας. Άρα ο πίνακας και το μέγεθός του πρέπει να δοθούν ως ορίσματα. Οπότε:

Αναζήτηση σε πίνακα (πίνακας ακεραίων Pinakas, ακέραιος meg) {

1. Όρισε μια μεταβλητή τύπου ακεραίου με όνομα arithmos
2. arithmos=0
3. Όσο arithmos!=-1
4. Πάρε τιμή για arithmos

Στο βήμα 3 έχουμε την αρχή μιας επαναληπτικής δομής επειδή η εκφώνηση μας έλεγε να κάνουμε μια εργασία επαναληπτικά. Φυσικά, κάθε επαναληπτική δομή πρέπει να έχει μια συνθήκη τερματισμού και εδώ αυτή ήταν ο αριθμός που διαβάζουμε να είναι -1. Στο βήμα 4 διαβάζουμε τον αριθμό όσες φορές θέλει ο χρήστης ώστε να κάνουμε αυτή την εργασία επαναληπτικά για διαφορετικούς αριθμούς. Το βήμα 2 είναι μια συνηθισμένη τακτική όταν θέλουμε μια επαναληπτική δομή να σταματάει με βάση την τιμή μιας μεταβλητής που διαβάζουμε. Της δίνουμε αρχικά μια τέτοια τιμή ώστε να μπούμε στην επαναληπτική δομή και μετά διαβάζουμε την κανονική τιμή.

Γενικά τα βήματα 1 έως 4 θα τα χρησιμοποιείτε ως έχουν για να διαβάζετε την τιμή μιας μεταβλητής με επαναληπτικό τρόπο και να σταματάτε όταν αυτή πάρει μια συγκεκριμένη τιμή.

Τώρα είναι η ώρα να σκεφτούμε τον **αλγόριθμο ανώτερου επιπέδου**. Όπως είπαμε αυτός βγαίνει πολύ εύκολα, αρκεί να σκεφτούμε πολύ γενικά τι θέλουμε να κάνει ο αλγόριθμός μας:

5. Αναζήτησε στον πίνακα pinakas τον αριθμό arithmos και εμφάνισε σχετικό μήνυμα
6. Τέλος επανάληψης

Το βήμα 5 προφανώς είναι μια διαδικασία, άρα πρέπει να την ορίσουμε. Για να βρούμε ποια είναι τα βήματα της ας **ειδικευτούμε** σε ένα συγκεκριμένο παράδειγμα και ας σκεφτούμε απλά και αναλυτικά. Έστω λοιπόν ότι ο πίνακας είναι όπως φαίνεται παρακάτω και ο αριθμός που αναζητούμε είναι το 56:

12	9	48	13	89	56	1

Ξεκινάμε από την πρώτη θέση του πίνακα και συγκρίνουμε το περιεχόμενό του με το 56. Αφού δεν είναι το ίδιο προχωράμε στην επόμενη θέση. Επαναλαμβάνουμε τα ίδια μέχρι που φτάνουμε στο 56, τα περιεχόμενα τους είναι ίδια οπότε σταματάμε και έχουμε βρει αυτό που ψάχνουμε.

Η παραπάνω περίπτωση καλύπτει τα βασικά βήματα που πρέπει να ακολουθήσουμε αν η αναζήτηση είναι επιτυχής. Αν αντί για το 56 ψάχναμε το 78 θα φτάναμε και στην τελευταία θέση του πίνακα χωρίς να βρούμε ισότητα. Άρα η αναζήτηση θα αποτύγχανε.

Παραπάνω είδαμε μια περίπτωση όπου ακολουθήσαμε την αρχή της ειδίκευσης, αλλά ένα μόνο παράδειγμα δεν έφτανε. Έπρεπε να σκεφτούμε και όλες τις **ειδικές ή/και ακραίες περιπτώσεις**. Η αναγνώριση τέτοιων καταστάσεων είναι πάρα πολύ σημαντική ώστε το πρόγραμμά σας να δουλεύει απολύτως σωστά. Συνήθως για να τις βρείτε θα σκέφτεστε εξτρεμιστικά, δηλαδή τι θα γίνει αν:

1. Πάρω κάποια αναπάντεχη είσοδο
2. Φύγω έξω από τα όρια ενός πίνακα
3. Ο αλγόριθμός μου αναγκαστεί να ακολουθήσει μονοπάτια που δημιουργούν προβλήματα σε επόμενες εντολές (πχ κάποια μεταβλητή ορίζεται μέσα σε δομή επιλογής στην οποία ενδέχεται να μην μπούμε)

Γενικά περιπτώσεις που μια πρώτη ματιά δεν τις αποκαλύπτει τόσο εύκολα γιατί δεν συμβαίνουν συχνά.

Σημείωση: Όπως έχουμε πει ήδη, μια διαδικασία είναι κι αυτή ένας αλγόριθμος. Αν δεν μπορείτε να παράγετε αμέσως τα βήματα της αναζήτησης μέσα από τις προηγούμενες σκέψεις, μπορείτε να αντιμετωπίσετε αυτή τη διαδικασία σαν ένα ανεξάρτητο πρόβλημα και να εργαστείτε όπως περιγράψαμε στον οδηγό.

Έχοντας αυτά τα βήματα στο μυαλό μας ας ορίσουμε τη διαδικασία:

Αναζήτηση (πίνακας ακεραίων Α, ακέραιος n,ακέραιος arithmos)

1. Όρισε μια μεταβλητή τύπου ακεραίου με όνομα i
2. i=0
3. Όσο i<n
4. Αν A[i]=arithmos
5. Εμφάνισε “Η αναζήτηση ήταν επιτυχής”
6. Διακοπή της επανάληψης
7. αλλιώς
8. i=i+1
9. Τέλος επιλογής
10. Τέλος επανάληψης

Τα βήματα 1,2,3,8 και 9 είναι αυτά που περιγράψαμε στο προηγούμενο πρόβλημα για να επιτυγχάνουμε την προσπέλαση όλων των θέσεων ενός πίνακα.

Στο βήμα 4 ελέγχουμε αν το τρέχον κελί του πίνακα έχει περιεχόμενο ίδιο με τον αριθμό που αναζητούμε. Αν ναι, τότε εμφανίζουμε σχετικό μήνυμα (βήμα 5). Το βήμα 6 είναι μια ειδική εντολή που δεν έχουμε αναφέρει και μας δίνει τη δυνατότητα να σταματάμε αμέσως μια επαναληπτική δομή, αν σας μπερδεύει αντικαταστήστε αυτό το βήμα με το i=n. Απλά θέλουμε έναν τρόπο να διασφαλίσουμε ότι δε θα γίνουν άλλες επαναλήψεις αφού βρήκαμε αυτό που ψάχνουμε.

Οπότε συνολικά ο αλγόριθμος έχει αυτή τη μορφή:

Αναζήτηση σε πίνακα (πίνακας ακεραίων Pinakas, ακέραιος meg) {

1. Όρισε μια μεταβλητή τύπου ακεραίου με όνομα arithmos
2. arithmos=0
3. Όσο arithmos!=-1
4. Πάρε τιμή για arithmos
5. Όνομα: Αναζήτηση
   Είσοδοι: Pinakas, meg, arithmos
   Έξοδοι:
6. Τέλος επανάληψης

}

Αναζήτηση (πίνακας ακεραίων A, ακέραιος n,ακέραιος arithmos) {

1. Όρισε μια μεταβλητή τύπου ακεραίου με όνομα i
2. i=0
3. Όσο i<n
4. Αν A[i]=arithmos
5. Εμφάνισε “Η αναζήτηση ήταν επιτυχής”
6. Διακοπή της επανάληψης
7. αλλιώς
8. i=i+1
9. Τέλος επιλογής
10. Τέλος επανάληψης

}

Διαδικαστικό πρόβλημα 3

Πρόβλημα: ***Μετατρέψτε ένα πεζό γράμμα του λατινικού αλφαβήτου σε κεφαλαίο και εμφανίστε το***

Το συγκεκριμένο πρόβλημα έχει ως σκοπό του να σας εισάγει στην αναζήτηση πληροφοριών και την αντιμετώπιση καταστάσεων όπου δεν γνωρίζετε εξαρχής όλες τις λεπτομέρειες.

Αρχικά διαβάζουμε καλά το πρόβλημα για να **βρούμε τα δεδομένα** μας. Παρατηρούμε λοιπόν, ότι τα δεδομένα μας είναι ένα πεζό γράμμα του λατινικού αλφαβήτου.

Συνεχίζουμε προσπαθώντας να εντοπίσουμε **τη μορφή της λύσης** που θέλουμε να δώσουμε. Παρατηρούμε ότι η μορφή της λύσης θα είναι ένα κεφαλαίο γράμμα του λατινικού αλφαβήτου.

Για να δούμε τώρα πως θα **αναπαραστήσουμε τα δεδομένα** μας με βάση τους τύπους που έχουμε αναφέρει. Ο πλέον κατάλληλος είναι αυτός του χαρακτήρα, μιας και μιλάμε για ένα γράμμα του λατινικού αλφαβήτου.

Αφού είδαμε όλα τα προηγούμενα βήματα είμαστε σε θέση να διατυπώσουμε τα πρώτα βήματα του αλγορίθμου μας που αφορούν την **είσοδο των δεδομένων**:

1. Όρισε μια μεταβλητή τύπου χαρακτήρας με όνομα pezo
2. Πάρε τιμή για pezo

Με τα παραπάνω δύο βήματα έχουμε ορίσει τη μεταβλητή που θέλουμε να περιέχει το πεζό γράμμα και έχουμε δώσει σε αυτή την τιμή που επιθυμεί ο χρήστης.

Τώρα είναι η ώρα να σκεφτούμε τον **αλγόριθμο ανώτερου επιπέδου**. Όπως είπαμε αυτός βγαίνει πολύ εύκολα, αρκεί να σκεφτούμε πολύ γενικά τι θέλουμε να κάνει ο αλγόριθμός μας:

3. Μετάτρεψε το περιεχόμενο της pezo στο αντίστοιχο κεφαλαίο γράμμα και καταχώρησε την τιμή του σε μεταβλητή kefalaio
4. Εμφάνισε kefalaio

Τώρα ελέγχουμε ποια βήματα του παραπάνω αλγορίθμου δεν συμβαδίζουν με τις αρχές του απλού αλγορίθμου για υπολογιστή, αυτά θα τα **ορίσουμε ως διαδικασίες**. Το βήμα 3 λοιπόν, πρέπει να το ορίσουμε ως διαδικασία οπότε:

3. Όνομα: Μετατροπή σε κεφαλαίο
   Είσοδος: pezo
   Έξοδος: kefalaio

Τώρα φτάνουμε κοντά στη λύση του προβλήματος, αρκεί να ορίσουμε τα βήματα της παραπάνω διαδικασίας. Όμως, είμαστε αναγκασμένοι ο ορισμός της να γίνει με τέτοιον τρόπο ώστε να μπορεί να εκτελεστεί από υπολογιστή. Το ρεπερτόριο των εντολών μας δεν περιέχει κάποια εντολή μετατροπής σε κεφαλαία ή κάτι τέτοιο, άρα πρέπει **να αναζητήσουμε πληροφορίες** για τον τρόπο με τον οποίο αναπαριστάται ένα γράμμα στον υπολογιστή. Αν το μάθουμε αυτό τότε θα βρούμε τον τρόπο να ορίσουμε κάποια βήματα για τη δημιουργία ενός απλού αλγορίθμου.

Ρωτάμε λοιπόν, και μαθαίνουμε ότι ένας χαρακτήρας στον υπολογιστή αναπαριστάτε με έναν αριθμό, δηλαδή για παράδειγμα το a όταν φυλάσσεται στη μνήμη του υπολογιστή είναι ένας αριθμός, συγκεκριμένα το 97. Επίσης, η αρίθμηση γίνεται με συνεχόμενο τρόπο δηλαδή το b είναι το 98 κοκ Πριν τα πεζά έρχονται τα κεφαλαία και μεταξύ τους υπάρχουν κάποιοι ειδικοί χαρακτήρες. Σχηματικά δηλαδή η κατάσταση έχει κάπως έτσι:

A	B				a	b	c
65	66	…	…	…	97	98	99

Οπότε ο υπολογιστής βλέπει ακόμα και τους χαρακτήρες ως ακέραιους αριθμούς. Απλά όταν του λέμε να τους εμφανίσει ως χαρακτήρες κάνει την αντιστοίχηση του αριθμού με τον χαρακτήρα που αντιπροσωπεύει με βάση τον πίνακα Ascii (ένα κομμάτι του μελετάμε τώρα).

Τώρα που μάθαμε αυτή την πολύτιμη πληροφορία (που σε καμία περίπτωση δεν ήμασταν αναγκασμένοι να την ξέρουμε από πριν, αλλά κάναμε τον κόπο να τη μάθουμε) μπορούμε να δούμε πιο καθαρά. Αλλά ας μην παρασυρθούμε ακόμα από την ολότητα του προβλήματος, αλλά ας **ειδικευτούμε** σε μερικές μόνο περιπτώσεις.

Αν πάρουμε ως είσοδο το a τι θα κάναμε; Αφού το a είναι το 97 και το A είναι το 65 θα αφαιρούσαμε από το a το 32 (μην παραξενεύεστε που κάνουμε αφαίρεση αριθμού από χαρακτήρα, είπαμε ότι κάθε χαρακτήρας είναι ένας αριθμός για τον υπολογιστή). Για το b που είναι το 98 θα αφαιρέσουμε 32 και θα βρίσκαμε το B που είναι το 66. Γενικεύοντας λοιπόν, το κεφαλαίο γράμμα προέρχεται από το πεζό αρκεί να προσθέσουμε το 32.

Μετατροπή σε κεφαλαίο (χαρακτήρας p)
1. Όρισε μεταβλητή τύπου χαρακτήρα με το όνομα k
2. k=p-32
3. Επίστρεψε k

Αυτό ήταν, το πρόβλημά μας λύθηκε! Φτιάξαμε έναν πολύ καλό αλγόριθμο χωρίς να παραβιάσουμε καμία αρχή, χρησιμοποιώντας μόνο εντολές για υπολογιστή και διαδικασίες ☺

Συνολικά ο αλγόριθμος είναι έτσι:

1. Όρισε μια μεταβλητή τύπου χαρακτήρας με όνομα pezo
2. Όρισε μια μεταβλητή τύπου χαρακτήρας με όνομα kefalaio
3. Πάρε τιμή για pezo
4. Όνομα: Μετατροπή σε κεφαλαίο
   Είσοδος: pezo
   Έξοδος: kefalaio
5. Εμφάνισε kefalaio

Μετατροπή σε κεφαλαίο (χαρακτήρας p) {
1. Όρισε μεταβλητή τύπου χαρακτήρα με το όνομα k
2. k=p-32
3. Επίστρεψε k

}

Διαδικαστικό πρόβλημα 4

Πρόβλημα: ***Εικάζεται, αλλά χωρίς να έχει αποδειχθεί μαθηματικά, ότι αν ξεκινήσουμε από ένα φυσικό αριθμό και πάρουμε σαν επόμενο τον n/2 αν n άρτιος, ή τον 3n+1 αν n περιττός, συνεχίζοντας με αυτό τον τρόπο, κάποια στιγμή θα καταλήξουμε στο 1. Για παράδειγμα για τον αριθμό n=22 έχουμε***

22,11,34,17,52,26,13,40,20,10,5,16,8,4,2,1

Ως μήκος κύκλου ορίζεται το πλήθος των παραπάνω αριθμών, δηλαδή το 22 έχει μήκος κύκλου 16.

Δημιουργήστε ένα πρόγραμμα που δέχεται δύο θετικούς ακεραίους αριθμούς m και k και βρίσκει ποιος αριθμός στο διάστημα [m,k] έχει το μικρότερο μήκος κύκλου. Να εμφανίζει αυτόν τον αριθμό, το μήκος κύκλου του και την ακολουθία των αριθμών που δημιουργείται.

Σας παροτρύνω να προσπαθήσετε να διατυπώσετε αλγόριθμο για το παραπάνω πρόβλημα μόνοι σας. Είναι σίγουρα πολύ αυξημένης δυσκολίας γι' αυτό μην απογοητευτείτε καθόλου αν δεν τα καταφέρετε. Αν πιστεύετε ότι έχετε κάποια καλή λύση μπορείτε να τη φέρετε στο μάθημα όπου θα γίνει και μια συζήτηση πάνω στο πρόβλημα και γενικότερα σε όλα όσα αναφέρθηκαν.

Εδώ φτάσαμε στο τέλος! Αισθάνεστε μια ρίγη σε όλο το σώμα είμαι σίγουρος (χειμώνας είναι, θα την έχετε αρπάξει :P). Ελπίζω αυτός ο οδηγός να σας άρεσε και να σας φανεί χρήσιμος ☺ Για απορίες ή προβλήματα που σχετίζονται με αυτόν μπορείτε να μου στείλετε mail στο std05059@di.uoa.gr με τίτλο "Odigos".

Σας εύχομαι καλή σταδιοδρομία, may the Source be with you :D:D

Μάνος Καρβούνης
Αθήνα 2007

Για το μάθημα
Εισαγωγή στον Προγραμματισμό

Παράρτημα σχετικά με τις Δομές Δεδομένων

Τις δομές δεδομένων μπορούμε να τις δούμε σαν αποθηκευτικούς χώρους για τα δεδομένα μας. Τα τοποθετούμε εντός τους όπως τοποθετούνται τα περιοδικά σε μια βιβλιοθήκη ή οι πελάτες σε μια ουρά έξω από ένα εκδοτήριο εισιτηρίων. Κάθε τέτοιος χώρος προσδίδει στα αντικείμενα που περιέχει μια σχέση διάταξης, π.χ. στη βιβλιοθήκη το ένα βιβλίο είναι δίπλα στο άλλο και στο εκδοτήριο ο ένας πίσω απ' τον άλλον. Περισσότερα βέβαια, θα δείτε στα επόμενα εξάμηνα σε σχετικά μαθήματα.

Φυσικά μέσα τις δομές δεδομένων έχουμε πρόσβαση στα δεδομένα. Ο τρόπος που συμβαίνει αυτό ποικίλει ανάλογα με τη δομή (εξάγεται το παλιότερο, το πρώτο, το μικρότερο κλπ).

Δεν είναι ώρα να καλύψουμε σε βάθος το συγκεκριμένο θέμα γι' αυτό θα δούμε ονομαστικά μερικές δομές και θα αναλύσουμε μόνο την πιο απλή, αυτή του πίνακα.

1. Πίνακας
2. Στοίβα
3. Ουρά
4. Λίστα
5. Δέντρο
6. Γράφος

Τον πίνακα μπορούμε να τον φανταστούμε σαν πολλά, μικρά αριθμημένα κουτιά το ένα μετά το άλλο. Κάθε κουτί δέχεται ακριβώς ένα αντικείμενο και όλα τα κουτιά είναι αναγκασμένα να δέχονται το ίδιο είδος αντικειμένων. Στο πρώτο κουτί δίνουμε τον αριθμό 0, στο δεύτερο τον αριθμό 1 κοκ

Αφού είπαμε ότι σε κάθε κουτί βάζουμε ακριβώς ένα αντικείμενο και πρέπει όλα τα κουτιά να παίρνουν τον ίδιο τύπο αντικειμένων έχουμε τους εξής δυνατούς τύπους πινάκων:

1. Αριθμών ακεραίων
2. Αριθμών πραγματικών
3. Συμβόλων (που τελικά δημιουργούν την συμβολοσειρά που αναφέραμε πριν)

Ένας πίνακας βέβαια μπορεί να είναι 2,3,4 κοκ διαστάσεων. Ας δούμε λίγο έναν πίνακα 2 διαστάσεων που μπορεί να μας φανεί χρήσιμος

Κάθε κουτί του προσδιορίζεται πλέον από δύο νούμερα, τον αριθμό της γραμμής και της στήλης του.

Η αναφορά σε ένα στοιχείο του πίνακα γίνεται λέγοντας τον αριθμό του μέσα στον πίνακα. Για παράδειγμα για τον παρακάτω πίνακα που τον ονομάζουμε «PinakasA»

8	9	100	13
κουτί 0	κουτί 1	κουτί 2	κουτί 3

Αν πούμε μέσα στον αλγόριθμό μας «Εμφάνισε το στοιχείο στη θέση 1 του πίνακα PinakasA» θα μας εμφανίσει το 9.

Βήματα για την παραγωγή αλγορίθμων

1. Διαβάζουμε πολύ καλά και προσεκτικά το πρόβλημα
2. Αν έχουμε αμφιβολίες για κάποιες απαιτήσεις του ή δεν γνωρίζουμε πως να κάνουμε κάτι που απαιτείται για την επίλυσή του προσπαθούμε να αναζητήσουμε τις κατάλληλες πληροφορίες
3. Εντοπίζουμε και σημειώνουμε τα δεδομένα μας. Για να το πετύχουμε αυτό απαντάμε στην ερώτηση “Ποια αντικείμενα θα επεξεργαστώ ώστε να λύσω το πρόβλημα;”
4. Έπειτα προσπαθούμε να αντιληφθούμε πώς θα αναπαραστήσουμε τα δεδομένα μας χρησιμοποιώντας έναν από τους τύπους ακέραιος, πραγματικός, χαρακτήρας και συμβολοσειρά. Ουσιαστικά, μέσω της αναπαράστασής μας πρέπει να απορρίψουμε όλα εκείνα τα χαρακτηριστικά των δεδομένων που δεν αφορούν τις επεξεργασίες που πρέπει να κάνουμε για να φτάσουμε στη λύση του προβλήματος
5. Αποφαινόμαστε για το ποιες θα είναι οι είσοδοι και οι έξοδοι στον αλγόριθμό μας. Είσοδοι φυσικά, θα είναι τα δεδομένα μας όπως αποφασίσαμε να τα αναπαραστήσουμε και έξοδοι τα τελικά αποτελέσματα που θα είναι μια επαρκής απάντηση στο πρόβλημα
6. Σκεφτόμαστε γενικά ποιες ενέργειες πρέπει να ακολουθήσουμε για να φτάσουμε στη λύση του προβλήματος. Σε αυτό το στάδιο παράγουμε τον αλγόριθμο ανώτερου επιπέδου και δε δεσμευόμαστε από τις περιορισμένες εντολές που μας προσφέρονται. Η τεχνική της ειδίκευσης ενδέχεται να φανεί ιδιαίτερα χρήσιμη, αλλά πάντα πρέπει να μεριμνούμε και για τις ακραίες περιπτώσεις
7. Ορίζουμε όσες ενέργειες σκεφτήκαμε πριν ως διαδικασίες ακολουθώντας τους περιορισμούς και τον τρόπο επικοινωνίας που μας επιβάλει ένας απλός αλγόριθμος για υπολογιστή
8. Ο αλγόριθμός μας είναι έτοιμος!!

Και πάντα να έχετε στο μυαλό σας ότι αυτό που καλείστε να παράγετε τελικά, είναι απλά μια σειρά επεξεργασιών πάνω στα δεδομένα του προβλήματος ώστε μέσω αυτών να παραχθεί μια καινούργια πληροφορία που θα είναι η λύση του προβλήματος.