

**Αλγόριθμοι και Στοιχεία Πολυπλοκότητας.**

**Ενότητα 1:** Ορισμοί και Ορολογία

Διδάσκων: Ηλίας Κ Σάββας, Αναπληρωτής Καθηγητής.

Τμήμα Μηχανικών Πληροφορικής, Τεχνολογικής Εκπαίδευσης.

**Άδειες χρήσης.**

* Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons (C C). **Αναφορά δημιουργού (B Y), Μη εμπορική χρήση (N C), Μη τροποποίηση (N D), 3.0, Μη εισαγόμενο.**
* Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.



**Χρηματοδότηση.**

* Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
* Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.



# Σκοποί ενότητας.

Ο αναγνώστης να μπορεί να:

1) χρησιμοποιεί την αλγοριθμική γλώσσα,

2) επιλύει απλά προβλήματα,

3) αντιληφθεί την έννοια της πολυπλοκότητας.

# ΠΕΡΙΕΧΌΜΕΝΑ ΕΝΌΤΗΤΑΣ

[Σκοποί ενότητας. 2](#_Toc368174501)

[ΠΕΡΙΕΧΌΜΕΝΑ ΕΝΌΤΗΤΑΣ 3](#_Toc368174502)

[ΑΛΓΌΡΙΘΜΟΙ. 3](#_Toc368174503)

[ΠΡΟΓΡΆΜΜΑΤΑ. 3](#_Toc368174504)

[ΠΊΝΑΚΕΣ. 3](#_Toc368174505)

[1. ΟΡΙΣΜΟΊ ΚΑΙ ΟΡΟΛΟΓΊΑ. 4](#_Toc368174506)

[1.1 Εισαγωγή. 4](#_Toc368174507)

[1.2 Έννοιες κλειδιά. 5](#_Toc368174508)

[1.3 Πολυπλοκότητα και απόδοση αλγορίθμων. 6](#_Toc368174509)

[1.4 Αλγοριθμική γλώσσα. 7](#_Toc368174510)

[1.5 Παραδείγματα και λυμένες ασκήσεις. 8](#_Toc368174511)

# ΑΛΓΌΡΙΘΜΟΙ.

|  |  |
| --- | --- |
| ***Αλγόριθμος 1****: Εξίσωση 2 – Επίλυση Εξίσωσης 2ου βαθμού (ax2+bx+c = 0).* | 5 |
| ***Αλγόριθμος 2****: ΜΚΔ1 (Υπολογισμός Μέγιστου Κοινού Διαιρέτη, των αριθμών χ και ψ με τον αλγόριθμο του Ευκλείδη).*  | 9 |
| ***Αλγόριθμος 3****: Μεταφορά αραιού πίνακα.* | 9 |
| ***Αλγόριθμος 4****: Ανάκτηση στοιχείου αραιού πίνακα από τους δείκτες του.* | 10 |
| ***Αλγόριθμος 5****: Μεταφορά στοιχείων κάτω τριγωνικού πίνακα.* | 12 |
| ***Αλγόριθμος 6****: Ανάκτηση στοιχείου τριγωνικού πίνακα.* | 13 |

# ΠΡΟΓΡΆΜΜΑΤΑ.

|  |  |
| --- | --- |
| ***Πρόγραμμα 1****: Ανάκτηση στοιχείου αραιού πίνακα.* | 10 |
| ***Πρόγραμμα 2****: Κάτω τριγωνικός πίνακας.* | 14 |
| ***Πρόγραμμα 3****: Ενοποίηση ταξινομημένων πινάκων.* | 15 |

# ΠΊΝΑΚΕΣ.

|  |  |
| --- | --- |
| *Πίνακας 1.1: Κάτω τριγωνικός.* | 12 |

# 1. ΟΡΙΣΜΟΊ ΚΑΙ ΟΡΟΛΟΓΊΑ.

## 1.1 Εισαγωγή.

Αλγόριθμος είναι ένα σύνολο οδηγιών, οι οποίες επιλύουν ένα συγκεκριμένο πρόβλημα ή μία κλάση προβλημάτων. Επιπλέον, ένας αλγόριθμος πρέπει να πληροί και τα ακόλουθα κριτήρια:

1. ***Είσοδος***: Για να λειτουργήσει πρέπει να εισαχθούν Ν δεδομένα, από κάποια εξωτερική πηγή, (το Ν μπορεί να είναι και μηδέν).
2. ***Έξοδος***: Με το τέλος του αλγόριθμου, πρέπει να παράγεται τουλάχιστο ένα αντικείμενο σαν αποτέλεσμα.
3. ***Καλά ορισμένος***: Η κάθε οδηγία πρέπει να είναι απόλυτα καθορισμένη, κατανοητή, και να μην αφήνει κανένα περιθώριο αμφισβήτησης, (πχ σε κάποια απόφαση).
4. ***Πεπερασμένος***: Να τελειώνει σε πεπερασμένο αριθμό βημάτων.
5. ***Αποτελεσματικός***: Κάθε οδηγία του αλγόριθμου, πρέπει να είναι απόλυτα βασική, και να μπορεί να εκτελεσθεί από μία υπολογιστική μηχανή. Δηλαδή το τρίτο κριτήριο δεν είναι αρκετό: πρέπει η οδηγία επιπλέον να είναι έτσι, ώστε να φέρνει κάποιο αποτέλεσμα.
6. ***Γενικός***: Εάν είναι δυνατόν, ο αλγόριθμος πρέπει να λύνει μία κλάση προβλημάτων και όχι ένα και μοναδικό πρόβλημα. Για παράδειγμα, καλός είναι ο αλγόριθμος που επιλύει την εξίσωση πρώτου βαθμού, *aχ + b = 0* και όχι μία συγκεκριμένη εξίσωση, σαν την *5χ + 6 = 0*.

Τέλος, ένας αλγόριθμος πρέπει να περιγράφεται αναλυτικά και με τέτοιο τρόπο, ώστε να είναι απόλυτα κατανοητός, ακόμη και σε κάποιον που δεν ξέρει το πρόβλημα που επιλύει, *(βλέπε Αλγόριθμος 1).* Για την ιστορία, η λέξη αλγόριθμος προήλθε από το όνομα ενός Πέρση, που είχε γράψει ένα βιβλίο Μαθηματικών (825 μ.Χ.), τον Abu Ja’far Mohammed ibn Musa al Khowarizmi.

Ανάλογα με την τεχνική επίλυσης ενός προβλήματος, οι αλγόριθμοι διακρίνονται σε:

* Αναδρομικοί, (recursive): αλγόριθμοι που χρησιμοποιούν αναδρομικές λύσεις προβλημάτων, πχ πολυώνυμα Hermite, υπολογισμός παραγοντικού, και άλλα.
* Διαίρει και βασίλευε, (divide and conquer): επιλύουν το πρόβλημα αναγάγοντάς το σε μικρότερα ανάλογα προβλήματα, πχ quick sort, merge sort, και άλλα.
* Άπληστοι, (greedy): επιλύουν προβλήματα επιλέγοντας κάθε φορά την τοπικά βέλτιστη λύση, προσδοκώντας την συνολικά βέλτιστη, πχ πρόβλημα επιστροφής ρέστων, χρονικός προγραμματισμός, και άλλα.
* Δυναμικού προγραμματισμού, (dynamic programming): συνήθως αναδρομικοί αλγόριθμοι οι οποίοι χρησιμοποιούν τις ενδιάμεσα παραγόμενες λύσεις, πχ πολλαπλασιασμός αλυσίδας πινάκων, και άλλα.
* Παράλληλοι, (parallel): εύρεση λύσης όπου το πρόβλημα δεν λύνεται σειριακά, αλλά πολλές σχέσεις του εκτελούνται παράλληλα, πχ πολλά προβλήματα πινάκων, τεχνικές τύπου «διαίρει και βασίλευε», και άλλες.

Επίσης, ανάλογα με την λύση που επιτυγχάνουν μπορούν να διακριθούν σε:

* Βέλτιστοι ή Άριστοι, (optimal): εύρεση της βέλτιστης λύσης του προβλήματος, πχ επίλυσης μίας εξίσωσης δευτέρου βαθμού.
* Προσεγγιστικοί ή ευρεστικοί, (approximation – heuristics): εύρεση «καλών» λύσεων σε άλυτα ή πολύ δύσκολα προβλήματα, πχ χρονοπρογραμματισμός εργασιών, χρωματισμός χάρτη και άλλα.

**Αλγόριθμος 1**: Εξίσωση 2 – Επίλυση Εξίσωσης 2ου βαθμού (ax2+bx+c = 0).

1: Εισαγωγή των a, b, και c.

2: Εάν (a=0 ΚΑΙ b=0 ΚΑΙ c=0). Τότε,

3: Αόριστη εξίσωση,

4: Τέλος αλγόριθμου «Εξίσωση 2»,

5: Τέλος Εάν.

6: Εάν (a=0 ΚΑΙ b=0 ΚΑΙ c≠0). Τότε,

7: Αδύνατη εξίσωση,

8: Τέλος αλγόριθμου «Εξίσωση 2»,

9: Τέλος Εάν.

10: Υπολόγισε *d* 🡨 .

11: Υπολόγισε *x*1 🡨 .

12: Υπολόγισε *x*2 🡨 .

13: Εκτύπωσε τα *x*1 και *x*2.

14: Τέλος Αλγόριθμου «Εξίσωση 2».

## 1.2 Έννοιες κλειδιά.

Για να λειτουργήσει ένας αλγόριθμος χρειάζεται πληροφορίες ή αλλιώς δεδομένα, τις οποίες και επεξεργάζεται και δίνει τις πιθανές λύσεις. Τα δεδομένα σε έναν αλγόριθμο κατηγοριοποιούνται σε τρεις γενικές κατηγορίες:

* ***Δεδομένα*** (Πληροφορίες).
	+ Αρχικά: είναι τα δεδομένα εισόδου που πρέπει να εφοδιαστεί ο αλγόριθμος, ώστε να μπορέσει να επεξεργαστεί το πρόβλημα.
	+ Ενδιάμεσα: αυτά είναι τα δεδομένα που παράγει ο αλγόριθμος, για να μπορέσει να φτάσει στην λύση, και τέλος,
	+ αποτελέσματα: είναι οι πληροφορίες που αφορούν την λύση του προβλήματος.

Στον *Αλγόριθμο 1*, αρχικά δεδομένα είναι τα a, b, και c, ενδιάμεσα το d, και αποτελέσματα τα x1 και x2.

Τα δεδομένα σε έναν αλγόριθμο περιγράφονται σαν ***μεταβλητές***, δηλαδή υπολογιστικές οντότητες για την αναπαράσταση των δεδομένων σε ένα πρόγραμμα ή αλγόριθμο. Επίσης, τα δεδομένα μπορεί να είναι διαφόρων τύπων. Οι ***τύποι δεδομένων,*** καθορίζουν τις τιμές που επιτρέπεται να πάρει μία μεταβλητή, πχ ακέραιος αριθμός, πραγματικός, και ούτω καθεξής, αλλά και τους τρόπους διαχείρισης ή επεξεργασίας των τιμών αυτών, δηλαδή τις επιτρεπόμενες πράξεις. Οι τύποι των δεδομένων μπορούν να διαχωριστούν στους απλούς και σύνθετους τύπους. Οι κατηγορίες των απλών τύπων είναι οι:

1. Ακέραιος αριθμός (integer),
2. Πραγματικός αριθμός (real – float),
3. Πραγματικός αριθμός μεγάλης ακρίβειας (double),
4. Χαρακτήρας (character), και
5. Boolean (με μοναδικές τιμές: Αληθής ⁄ Ψευδής ή 0 ⁄ 1).

Οι σύνθετοι τύποι δεδομένων ονομάζονται δομές δεδομένων. ***Δομή δεδομένων*** είναι ένας τύπος δεδομένων που αποτελείται από σύνθετες τιμές, δηλαδή τιμές που συντίθενται από άλλες απλούστερες επιμέρους τιμές (κόμβοι), και μεταξύ των οποίων υπάρχει ένα οργανωτικό σχήμα. Μερικές από τις πλέον συνηθισμένες δομές δεδομένων είναι οι:

1. Πίνακας (array),
2. Εγγραφή (record – structure),
3. Λίστα (list),
4. Ουρά (queue),
5. Στοίβα ή σωρός (stack),
6. Γράφος (graph),
7. Δένδρα (tree), και τα λοιπά,

και με επιτρεπτές πράξεις:

1. Διαπέραση,
2. Αναζήτηση,
3. Εισαγωγή,
4. Διαγραφή,
5. Διάταξη, και τα λοιπά.

## 1.3 Πολυπλοκότητα και απόδοση αλγορίθμων.

Σαν πολυπλοκότητα ή απόδοση αλγορίθμου, ορίζεται το κόστος χρήσης του αλγόριθμου για την επίλυση ενός προβλήματος. Μονάδες μέτρησης κόστους μπορεί να θεωρηθούν:

* Ο υπολογιστικός χρόνος εκτέλεσης του αλγόριθμου, δηλαδή χρήση της CPU μίας υπολογιστικής μηχανής.
* Η χρήση αποθηκευτικών χώρων.
* Η χρήση πόρων δικτύου, ή και
* σε οποιαδήποτε άλλη μονάδα που μπορεί να εκφράζει χρήση οποιονδήποτε πόρων υπολογιστικών και δικτύων.

Βέβαια, ο χρόνος εκτέλεσης ενός αλγόριθμου θεωρείται σαν το πιο σημαντικό μέτρο χαρακτηρισμού του αλγόριθμου. Μάλιστα, επειδή σε πάρα πολλές περιπτώσεις, είναι εντελώς αδύνατο να υπολογισθεί αυτός ο χρόνος με ακρίβεια, υπολογίζονται συναρτήσεις που αποδίδουν τον καλύτερο, μέσο, και χειρότερο χρόνο του.

Για παράδειγμα στον *Αλγόριθμο 1,* ο καλύτερος χρόνος είναι όταν η εξίσωση είναι αόριστη, και ο χειρότερος όταν θα υπάρχουν οι δύο λύσεις.

## 1.4 Αλγοριθμική γλώσσα.

Για την περιγραφή των αλγορίθμων, έχει προταθεί και χρησιμοποιείται μία «γλώσσα», η λεγόμενη αλγοριθμική γλώσσα, η οποία είναι αποδεσμευμένη από τις λεπτομέρειες μιας κανονικής γλώσσας προγραμματισμού, σαν την C ή την Pascal, Java, και άλλες. Ο στόχος είναι να περιγράφεται η λύση ενός προβλήματος, με τέτοιο τρόπο ώστε να μπορεί να μεταφερθεί αργότερα σε οποιαδήποτε γλώσσα προγραμματισμού. Επίσης, στην αλγοριθμική γλώσσα πρέπει να αποφεύγονται εκφράσεις που υπάρχουν σε κάποια γλώσσα προγραμματισμού, ενώ δεν υπάρχουν σε άλλες, (για παράδειγμα οι τελεστές ++ , --, και άλλοι, της C).

Σε γενικές γραμμές, τα δομικά στοιχεία της αλγοριθμικής γλώσσας είναι τα ακόλουθα:

1. *Δεδομένα* τα οποία μπορεί να είναι:
	1. Μεταβλητές ποσότητες (πχ x, a, mikos, και τα λοιπά).
	2. Σταθερές ποσότητες (πχ ένας αριθμός, ένα όνομα, και άλλα).
2. *Τελεστές*:
	1. Πρόσθεση,
	2. Αφαίρεση,
	3. Πολλαπλασιασμός,
	4. Διαίρεση,
	5. Υπόλοιπο,
	6. Ισότητα,
	7. Όλες οι πιθανές ανισότητες, και τα λοιπά.
3. *Παραστάσεις*:
	1. Εκχώρηση τιμής, πχ a 🡨 3x+1 και τα λοιπά,
	2. Κριτήρια ή συνθήκες, πχ α > 7, onoma = ”Nikos”, και τα λοιπά.
4. *Διατάξεις Ελέγχου Ροής* του Αλγόριθμου:
	1. Η πρόταση:
		1. Εάν (συνθήκη) Τότε {
			1. προτάσεις}.
		2. Αλλιώς,
			1. {προτάσεις},
		3. Τέλος Εάν (ή If … Then … Else … End If).
	2. Η πρόταση πολλαπλής επιλογής:
		1. Επέλεξε με το …
			1. Περίπτωση 1: {προτάσεις},
			2. Περίπτωση 2: {προτάσεις},
			3. …
			4. Περίπτωση ν: {προτάσεις},
		2. Τέλος Επιλογής (Switch ή Case).
5. *Επαναληπτικές Δομές*:
	1. Για … Τέλος Για (For … End For),
		1. Για (προτάσεις ελέγχου επανάληψης),
			1. {προτάσεις},
		2. Τέλος Για.
	2. Εφόσον (While … End While),
		1. Εφόσον (προτάσεις ελέγχου επανάληψης),
			1. {προτάσεις},
		2. Τέλος Εφόσον.
	3. Επανέλαβε (Do … While ή Repeat … Until),
		1. Επανέλαβε,
			1. {προτάσεις},
		2. Μέχρι (προτάσεις ελέγχου επανάληψης).
6. *Διαδικασίες* (procedures – routines):
	1. Διαδικασία «όνομα διαδικασίας με πιθανές παραμέτρους»,
		1. {προτάσεις διαδικασίας},
	2. Τέλος Διαδικασίας «όνομα διαδικασίας».
7. *Συναρτήσεις* (functions):
	1. Τύπος επιστρεφόμενου δεδομένου «όνομα συνάρτησης με πιθανές παραμέτρους»,
		1. {προτάσεις συνάρτησης,
		2. Επέστρεψε …},
	2. Τέλος Συνάρτησης «όνομα συνάρτησης».

## 1.5 Παραδείγματα και λυμένες ασκήσεις.

1. Υπολογισμός Μέγιστου Κοινού Διαιρέτη με χρήση του αναδρομικού αλγόριθμου του Ευκλείδη.

**Λύση.**

***Αλγόριθμος 2****: ΜΚΔ1 (Υπολογισμός Μέγιστου Κοινού Διαιρέτη, των αριθμών χ και ψ με τον αλγόριθμο του Ευκλείδη).*

1: Δεδομένα ⁄ Είσοδος: ακέραιοι χ, ψ.

2: Ακέραιος ΜΚΔ1 (ακέραιος χ, ακέραιος ψ).

3: Αρχή,

4: Εάν ψ > 0. Τότε,

5: Επέστρεψε ΜΚΔ (ψ, χ υπόλοιπο ψ).

6: Αλλιώς,

7: Επέστρεψε ψ,

8: Τέλος Εάν.

9: Τέλος Συνάρτησης «ΜΚΔ1».

1. *Αραιός ονομάζεται ένας πίνακας, του οποίου τα περισσότερα στοιχεία είναι μηδενικά, σε ποσοστά που υπερβαίνουν το 80%. Για να μην γίνεται αυτή η σπατάλη χώρου, ας υποτεθεί ότι μεταφέρονται τα μη μηδενικά στοιχεία του πίνακα, σε ένα νέο πίνακα και διαγράφεται ο παλιός. Το πρόβλημα είναι ότι η θέση των στοιχείων στον αρχικό πίνακα είναι σημαντική, οπότε για να διατηρηθεί αυτή η πληροφορία, ο νέος πίνακας αντί για μονοδιάστατος, θα μπορούσε να είναι δισδιάστατος με την εξής οργάνωση: σε κάθε γραμμή η πρώτη στήλη να περιέχει τα μη μηδενικά στοιχεία του αραιού πίνακα, ενώ στην δεύτερη και τρίτη στήλη, να τοποθετούνται οι συντεταγμένες αυτών των στοιχείων που είχαν στον αρχικό πίνακα, (δηλαδή τον αριθμό της γραμμής και της στήλης τους). Με βάση αυτή τη διαδικασία, να δημιουργηθούν οι αλγόριθμοι μεταφοράς των στοιχείων και αναζήτησης στοιχείου με βάση την θέση του.*

***Αλγόριθμος 3****: Μεταφορά αραιού πίνακα.*

1: Δεδομένα ⁄ Είσοδος: Αραιός πίνακας ΠNxΜ, Πίνακας στόχος ΣΤx3.

2: Διαδικασία «Μεταφορά».

3: Αρχή,

4: Ακέραιοι: ι, κ, λ.

5: λ 🡨 0.

6: Για ι = 1 Μέχρι Ν,

7: Για κ = 1 Μέχρι Μ,

8: Εάν (Πι,κ ≠ 0) Τότε,

9: λ 🡨 λ + 1,

10: Σλ,1 🡨 Πι,κ,

11: Σλ,2 🡨 ι,

12: Σλ,3 🡨 κ,

13: Τέλος Εάν.

14: Τέλος Για (κ),

15: Τέλος Για (ι).

16: Τέλος Διαδικασίας «Μεταφορά».

***Αλγόριθμος 4****: Ανάκτηση στοιχείου αραιού πίνακα από τους δείκτες του.*

1: Δεδομένα ⁄ Είσοδος: Πίνακας ΣΤx3.

2: Τύπος Στοιχείου Αραιού Πίνακα «Ανάκτηση (ακέραιος χ, ακέραιος ψ)».

3: Αρχή,

4: Ακέραιος: ι 🡨 0,

5: Boolean: flag 🡨 «Ψευδής»,

6: Τύπος δεδομένων των στοιχείων του αραιού πίνακα: β.

7: Επανέλαβε,

8: ι 🡨 ι + 1,

9: Εάν ( Σι,2 = χ ΚΑΙ Σι,3 = υ). Τότε,

10: Β 🡨 Σι,1,

11: Flag 🡨 «Αληθής»,

12: Τέλος Εάν.

13: Μέχρι (flag = «Αληθής» Η ι = Τ).

14: Εάν (flag = «Αληθής»). Τότε,

15: Επέστρεψε β.

16: Αλλιώς,

17: Επέστρεψε 0,

18: Τέλος Εάν.

19: Τέλος Συνάρτησης «Ανάκτηση».

Στο *Πρόγραμμα 1* που ακολουθεί, περιγράφεται ένας διαφορετικός αλγόριθμος ανάκτησης στοιχείου.

***Πρόγραμμα 1****: Ανάκτηση στοιχείου αραιού πίνακα.*

#include <stdio.h>

#define M 7 /\* Μέγεθος Πίνακα Στόχου \*/,

int S[M][3]={ -3, 3, 9,

-12, 7, 2,

-5, 12, 4,

-7, 12, 8,

-31, 12, 9,

-5, 15, 17,

-9, 20, 1};

/\* S είναι ο πίνακας στον οποίο μεταφέρθηκε ο αραιός πίνακας \*/,

int anaktisi(int, int);

main()

{

 int i;

 int ii,jj; /\* Δείκτες του υπό ανάκτηση στοιχείου \*/,

 int s; /\* Το αναζητούμενο στοιχείο \*/,

 /\* Εκτύπωση πίνακα στόχου \*/,

 for (i=0; i<M; i++)

 printf("\n %5d % 5d %5d", S[i][0], S[i][1], S[i][2]);

/\* Είσοδος παραμέτρων \*/,

 printf("\n\n Εισαγωγή γραμμής και στήλης του ζητούμενου στοιχείου:");

 scanf("%d %d", &ii, &jj);

 /\* Ανάκτηση στοιχείου \*/,

 s = anaktisi (ii, jj);

 /\* Εκτύπωση στοιχείου \*/,

 printf("\n\n Stoixeio = %5d \n\n", s);

}

/\* ΣΥΝΆΡΤΗΣΗ ΑΝΆΚΤΗΣΗΣ \*/,

int anaktisi(int i, int j)

{

 int k;

 k = 0;

 while (S[k][1]<i && k<M-1)

 k++;

if (S[k][1] == i)

 while (S[k][2]<j && k<M-1)

k++;

 if (S[k][1] == i && S[k][2] == j)

 return S[k][0];

 else

 return 0;

}

1. Τριγωνικοί πίνακες. Ένας τετραγωνικός πίνακας ονομάζεται κάτω (επάνω) τριγωνικός εάν μόνο τα στοιχεία κάτω (επάνω) της κύριας διαγωνίου του, είναι μη μηδενικά, (η κύρια διαγώνιος συμπεριλαμβάνεται στα μη μηδενικά στοιχεία). Επειδή πάλι υπάρχει σπατάλη χώρου (πολλά μηδενικά στοιχεία), ζητείται μία μέθοδος απεικόνισης του πίνακα, ώστε να αποφευχθεί αυτή η σπατάλη χώρου.

**Λύση.**

Εάν ο τριγωνικός πίνακας μεταφερθεί σε ένα νέο μονοδιάστατο πίνακα, τότε το πρώτο ζητούμενο είναι το μέγεθος του πίνακα στόχου. Η γενική μορφή ενός κάτω τριγωνικού πίνακα είναι η ακόλουθη:

*Πίνακας 1.1: Κάτω τριγωνικός.*

Δηλαδή τα μη μηδενικά στοιχεία ανά γραμμή είναι:

1η γραμμή: 1,

2η γραμμή: 2,

3η γραμμή: 3,

…

nη γραμμή: n.

Επομένως το σύνολο των μη μηδενικών στοιχείων είναι:

 ,

επομένως και το μέγεθος του γραμμών του μονοδιάστατου πίνακα στόχου πρέπει να είναι , όπου n το πλήθος των γραμμών ⁄ στηλών του τριγωνικού πίνακα.

Ο αλγόριθμος μεταφοράς δίνεται στον  *Αλγόριθμο 5*.

***Αλγόριθμος 5****: Μεταφορά στοιχείων κάτω τριγωνικού πίνακα.*

1: Δεδομένα ⁄ Είσοδος: Κάτω Τριγωνικός Πίνακας Τnxn, Πίνακας Στόχος ΣΝ.

2: Διαδικασία «Μεταφορά Τριγωνικού».

3: Αρχή,

4: Ακέραιοι: ι, κ, λ = 0.

5: Για ι = 1 Μέχρι n,

6: Για κ = 1 Μέχρι ι,

7: λ 🡨 λ+1,

8: Σλ 🡨 Τι,κ,

9: Τέλος Για (κ),

10: Τέλος Για (λ),

11: Τέλος διαδικασίας «Μεταφορά Τριγωνικού».

Το επόμενο πρόβλημα, είναι πώς θα γίνεται ανάκτηση ενός στοιχείου του τριγωνικού πίνακα που έχει πλέον μεταφερθεί, από τους δείκτες του, έστω i, j. Εάν i < j τότε το ζητούμενο στοιχείο είναι μηδενικό. Εάν όμως i ≥ j τότε, το στοιχείο που βρίσκεται στη i γραμμή, έπεται όλων των στοιχείων των προηγουμένων γραμμών i-1, και το πλήθος των στοιχείων αυτών είναι:

1η γραμμή: 1,

2η γραμμή: 2,

3η γραμμή: 3,

…

(i-1)η γραμμή: i-1.

Επομένως το σύνολο των μη στοιχείων είναι:

 ,

και επιπλέον προηγούνται και τα j-1 στοιχεία της ι γραμμής. Δηλαδή, το σύνολο των στοιχείων που προηγούνται του i, j είναι:

,

και αυτό δείχνει ότι το στοιχείο i, j έχει καταχωρηθεί στον πίνακα στόχο στη θέση:

.

Στον *Αλγόριθμο 6* περιγράφεται η διαδικασία ανάκτησης.

***Αλγόριθμος 6****: Ανάκτηση στοιχείου τριγωνικού πίνακα.*

1: Δεδομένα ⁄ Είσοδος: Πίνακας Στόχος ΣΝ.

2: Τύπος Στοιχείου Τριγωνικού Πίνακα «Ανάκτηση ΚΤΠ (ακέραιος i, ακέραιος j)».

3: Αρχή,

4: Ακέραιος k.

5: Εάν (i < j) Τότε,

6: Επέστρεψε 0,

7: Τέλος Εάν.

8: k 🡨 .

9: Επέστρεψε Σk.

10: Τέλος συνάρτησης «Ανάκτηση ΚΤΠ».

Στο *Πρόγραμμα 2* δίνεται η μεταφορά και ανάκτηση στοιχείου κάτω τριγωνικού πίνακα. Η διαφορά της συνάρτησης ανάκτησης του *Προγράμματος 2*, με αυτής του  *Αλγόριθμου 6*,οφείλεται στο ότι στην C, οι δείκτες των πινάκων αρχίζουν από το μηδέν.

***Πρόγραμμα 2****: Κάτω τριγωνικός πίνακας.*

#include <stdio.h>

#define n 4 /\* Μέγεθος Κάτω Τριγωνικού Πίνακα \*/,

#define N n\*(n+1) / 2 /\* Μέγεθος Πίνακα Στόχου \*/,

int T[n][n]={ -3, 0, 0, 0,

 -12, 7, 0, 0,

 -5, 12, 4, 0,

 -7, 12, 8, 9};

int S[N];

void metafora(void);

int anaktisi(int, int);

main ()

{

 int i, j, l;

 /\* Αρχικοποίηση πίνακα στόχου \*/,

 for (i=0; i<N; i++)

S[i] = 0;

 metafora ();

 printf("\n\n ΚΆΤΩ ΤΡΙΓΩΝΙΚΌΣ ΠΊΝΑΚΑΣ \n\n");

 for (i=0; i<n; i++) {

for (j=0; j<n; j++)

printf ("%5d", T[i][j]);

printf ("\n");

 }

 printf ("\n\n ΠΊΝΑΚΑΣ ΣΤΌΧΟΣ \n\n");

 for (i=0; i<N; i++)

printf("%5d", S[i]);

printf("\n\n");

printf("\n Είσοδος δεικτών γραμμής και στήλης, του υπό αναζήτηση στοιχείου:");

scanf("%d %d", &i, &j);

l = anaktisi (i, j);

if (l == -1)

printf("\n\n Το στοιχείο είναι το: %d \n\n", 0);

else

printf("\n\n Το στοιχείο είναι το: %d \n\n", S[l]);

 }

/\* ΣΥΝΑΡΤΗΣΕΙΣ \*/,

void metafora(void)

{

 int i, k, l = 0;

 for (i=0; i<n; i++)

for (k=0; k<=i; k++)

S[l++] = T[i][k];

}

int anaktisi(int x, int y)

{

 if (x < y)

return -1;

return x \* (x + 1) / 2 + y;

}

1. Ένα ενδιαφέρον πρόβλημα, είναι η ενοποίηση δύο ταξινομημένων πινάκων ΑΝ και ΒΜ, σε ένα νέο αλλά πάλι ταξινομημένο πίνακα CN+M. Μία απλή λύση είναι η μεταφορά του πρώτου στον C, μετά η μεταφορά του δεύτερου στον C, και τέλος η ταξινόμηση του C. Αυτή είναι μία χρονοβόρα λύση, γιατί χρειάζεται να ταξινομηθεί πάλι ο C, και δεν εκμεταλλεύεται το γεγονός ότι οι A και Β είναι ήδη ταξινομημένοι. Προφανώς, η καλύτερη και πιο γρήγορη λύση είναι να μεταφερθούν «ταυτόχρονα» οι Α και Β στον C ταξινομημένα. Στο *Πρόγραμμα 3*, δίνεται μία λύση αυτού του προβλήματος.

**Λύση.**

***Πρόγραμμα 3****: Ενοποίηση ταξινομημένων πινάκων.*

#include <stdio.h>

#define N 5 /\* Μέγεθος Α Πίνακα \*/,

#define M 9 /\* Μέγεθος Β Πίνακα \*/,

#define F N+M /\* Μέγεθος Πίνακα Στόχου \*/,

int A[N] = {2, 6, 7, 9, 9};

int B[M] = {-4, 0, 1, 6, 12, 12, 15, 17, 19};

int C[F];

void enopoiisi(void);

main()

{

 int i;

 printf("\n\n Α ΠΊΝΑΚΑΣ \n\n");

 for (i=0; i<N; i++)

printf("%5d", A[i]);

 printf("\n\n B ΠΊΝΑΚΑΣ \n\n");

 for (i=0; i<M; i++)

printf("%5d", B[i]);

 enopoiisi();

 printf("\n\n Ενοποιημένος Πίνακας \n\n");

 for (i=0; i<F; i++)

printf("%5d", C[i]);

printf("\n\n");

}

/\* ΣΥΝΑΡΤΗΣΕΙΣ \*/,

void enopoiisi(void)

{

 int i, j, k;

 for (i=0, j=0, k=0; k<F; k++) {

if (i == N) { C[k] = B[j++]; continue; }

if (j == M) { C[k] = A[i++]; continue; }

C[k] = (A[i]<B[j]) ? A[i++] : B[j++];

 }

}

Τέλος πρώτης ενότητας.

 