Τ.Ε.Ι. ΠΑΤΡΑΣ

ΣΧΟΛΗ ΔΙΟΙΚΗΣΗΣ ΚΑΙ ΟΙΚΟΝΟΜΙΑΣ

ΤΜΗΜΑ ΕΠΙΧ/ΚΟΥ ΣΧΕΔΙΑΣΜΟΥ ΚΑΙ ΠΛΗΡΟΦΟΡΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΑΝΑΠΤΥΞΗ ΑΛΓΟΡΙΘΜΩΝ ΓΙΑ ΠΙΝΑΚΕΣ ΕΦΑΡΜΟΓΕΣ ΣΤΗ ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ C++ ΚΑΙ JAVA

ΣΠΟΥΔΑΣΤΕΣ: **ΣΚΟΥΡΑΣ ΛΑΜΠΡΟΣ**
ΓΙΑΝΝΑΚΑΣ ΑΓΓΕΛΟΣ
ΠΑΡΘΕΝΗΣ ΙΩΑΝΝΗΣ

ΕΠΟΠΤΕΥΩΝ ΚΑΘΗΓΗΤΗΣ: ΑΡΗΣ ΜΠΑΚΑΛΗΣ

ΠΑΤΡΑ ΝΟΕΜΒΡΙΟΣ 2008

ΠΕΡΙΕΧΟΜΕΝΑ

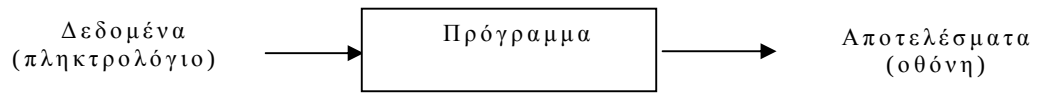
1 Περίληψη

Η πτυχιακή εργασία δομείται σε δύο κύρια μέρη. Στο Α μέρος γίνεται μια σύντομη εισαγωγή στη C++ και περιγράφονται βασικές έννοιες της γλώσσας όπως π.χ. το αλφάβητο της C++ (Κεφάλαιο 4), το Λεξιλόγιο (Κεφάλαιο 5), η δομή και ο τρόπος γραφής ενός προγράμματος σε C++ (Κεφάλαια 6 και 7), βασικές εντολές της γλώσσας (Κεφάλαια 8, 9 και 10) και τέλος παρουσιάζονται οι πίνακες (Κεφάλαιο 11) που είναι και ο βασικός στόχος της πτυχιακής εργασίας. Στη συνέχεια παρουσιάζονται πάρα πολλοί αλγόριθμοι αρχικά για μονοδιάστατους πίνακες και στη συνέχεια για δισδιάστατους πίνακες που καλύπτουν όλες τις λειτουργίες που μπορούν να εφαρμοστούν σε αυτούς.

Στο Β μέρος γίνεται μια σύντομη εισαγωγή στη Java και περιγράφονται αρχικά βασικές έννοιες της γλώσσας και των πινάκων (Κεφάλαια 12-17) όπως και στη C++ ενώ στη συνέχεια περιγράφονται οι ίδιοι αλγόριθμοι με τη C++ προκειμένου να τονιστούν οι διαφορές στην υλοποίηση των αλγορίθμων ανάμεσα στις δύο γλώσσες. Στη Java η υλοποίηση των αλγορίθμων βασίζεται κυρίως στη χρήση συναρτήσεων και επίσης μερικοί από τους αλγόριθμοι της Java παρουσιάζονται με διαφορετικούς τρόπους ώστε να τονιστεί στο χρήστη η ευκολία και η αποτελεσματικότητα της χρήσης συναρτήσεων.

2 Α Μέρος. Εισαγωγή στη C++

Ο προγραμματισμός είναι μια διαδικασία επίλυσης προβλημάτων με χρήση Η/Υ. Ένα πρόγραμμα είναι ένα σύνολο εντολών οι οποίες επιλύουν όπως αναφέραμε ένα συγκεκριμένο πρόβλημα. Η γλώσσα C είναι δεύτερης γενιάς ενώ η γλώσσα C++ είναι πέμπτης γενιάς και ανήκει στις αντικειμενοστραφείς γλώσσες. Κάθε πρόγραμμα έχει 2 βασικά χαρακτηριστικά: λαμβάνει δεδομένα από το χρήστη μέσω του πληκτρολογίου, τα επεξεργάζεται και εμφανίζει τα αντίστοιχα αποτελέσματα στην οθόνη, δηλαδή κάθε πρόγραμμα λαμβάνει είσοδο από το πληκτρολόγιο και δίνει έξοδο στην οθόνη. Βέβαια υπάρχουν και προγράμματα (αρκετά σπάνια) τα οποία δεν χρειάζεται να λάβουν δεδομένα. Σχηματικά ένα πρόγραμμα αναπαριστάνεται ως εξής:



3 Διαδικασία συγγραφής προγραμμάτων στη C+

Για να γράψουμε ένα πρόγραμμα σε C++ ακολουθούμε κατά σειρά τα ακόλουθα βήματα:

- Γράφουμε πρώτα τις εντολές του προγράμματος μέσα στο διορθωτή κειμένου (editor). Ο διορθωτής κειμένου είναι ένας απλός κειμενογράφος με περιορισμένες δυνατότητες μορφοποίησης, εκτύπωσης κ.λ.π. Το πρόγραμμα που γράφουμε ονομάζεται πηγαίο (source program)

- Στη συνέχεια στέλνουμε το πηγαίο πρόγραμμα στο μεταγλωττιστή (compiler) ο οποίος αναλαμβάνει να μεταφράσει το πηγαίο πρόγραμμα σε γλώσσα μηχανής. Αυτό είναι εφικτό μόνο όταν ο μεταγλωττιστής δεν εντοπίσει στο πρόγραμμα μας συντακτικά λάθη. Αν υπάρχουν συντακτικά λάθη ο μεταγλωττιστής μας ενημερώνει για το ποια λάθη είναι αυτά και σε ποια γραμμή έγιναν. Συντακτικά είναι τα λάθη που οφείλονται στον τρόπο γραφής των προγραμμάτων π.χ. έλλειψη κάποιου ερωτηματικού, παρένθεσης κ.λ.π. Το πρόγραμμα που εξάγεται από τον μεταγλωττιστή ονομάζεται αντικείμενο (object program)

- Στη συνέχεια συνδέουμε το αντικείμενο πρόγραμμα με τις βιβλιοθήκες της γλώσσας. Μια βιβλιοθήκη περιέχει εντολές κοινής χρήσης όπως π.χ. εντολές εισόδου-εξόδου, κοινές μαθηματικές συναρτήσεις κ.λ.π. Τη σύνδεση αυτή την κάνει ένα ειδικό πρόγραμμα που ονομάζεται συνδέτης (linker). Το πρόγραμμα που παράγεται από το συνδέτη ονομάζεται εκτελέσιμο (executable program)

- Εκτελούμε το πρόγραμμα και βλέπουμε τις απαντήσεις στην οθόνη

4 Αλφάβητο C++

Όπως κάθε ομιλούμενη γλώσσα χρησιμοποιεί τους χαρακτήρες του αλφαβήτου για να σχηματίσει λέξεις, έτσι και κάθε γλώσσα προγραμματισμού έχει το δικό της αλφάβητο. Με τους χαρακτήρες του αλφαβήτου της η γλώσσα σχηματίζει τις λέξεις ή τα σύμβολα

που αποτελούν το λεξιλόγιο της γλώσσας. Το αλφάβητο της γλώσσας C++ περιλαμβάνει τους ακόλουθους χαρακτήρες:

- οριζόντιος στηλοθέτης (αναπαριστάνεται με το σύμβολο \t)
- κατακόρυφος στηλοθέτης (αναπαριστάνεται με το σύμβολο \v)
- αλλαγής σελίδας
- νέας γραμμής (αναπαριστάνεται με το σύμβολο \n)
- γράμματα αγγλικής αλφαβήτου κεφαλαία και πεζά
- αριθμοί
- σύμβολα πράξεων + - * / %
- σύμβολα στίξης ( ) & { } [ ] , . κ.λ.π.

5 Λεξιλόγιο γλώσσας

Το λεξιλόγιο της C περιλαμβάνει τα ακόλουθα:

- δεσμευμένες λέξεις (reserved words)
- τελεστές (operators)
- αναγνωριστές (identifiers)

5.1 Δεσμευμένες λέξεις

Οι δεσμευμένες λέξεις μιας γλώσσας έχουν αυστηρά καθορισμένη έννοια και τρόπο χρήσης και χρησιμοποιούνται σε συγκεκριμένες θέσεις ενός προγράμματος. Μερικές από τις πιο χαρακτηριστικές δεσμευμένες λέξεις είναι π.χ. ***int***, float, do, ***if***, ***for***, sizeof κ.λ.π.

5.2 Αναγνωριστές

Στην κατηγορία των αναγνωριστών (identifiers) ανήκουν λέξεις που κατασκευάζει ο προγραμματιστής. Τέτοιες λέξεις δίνει ο προγραμματιστής σε μεταβλητές, σταθερές. συναρτήσεις και δικούς του τύπους δεδομένων. Η δημιουργία ονομάτων στη C++ διέπεται από ένα σύνολο κανόνων οι οποίοι είναι ανεξάρτητοι από τον τύπου του αντικειμένου στον οποίο αναφέρεται το όνομα. Οι κανόνες αυτοί αναφέρονται στην επόμενη παράγραφο.

5.3 Μεταβλητές-Σταθερές

Μεταβλητή (variable) είναι μια ποσότητα η οποία μπορεί να μεταβάλλει την τιμή κατά τη διάρκεια εκτέλεσης ενός προγράμματος, ενώ σταθερά (constant) είναι μια ποσότητα η οποία έχει σταθερή τιμή σε όλη τη διάρκεια εκτέλεσης του προγράμματος. Μεταβλητές χρησιμοποιούνται σχεδόν σε κάθε πρόγραμμα (πλην ελαχίστων εξαιρέσεων) και είναι θέσεις μνήμης στις οποίες:

- εισάγουμε δεδομένα από το πληκτρολόγιο
- καταχωρούμε αποτελέσματα που υπολογίζουμε από το πρόγραμμα

Τα ονόματα των μεταβλητών επιλέγονται ελεύθερα από τον προγραμματιστή πρέπει όμως υποχρεωτικά να υπακούουν στους ακόλουθους κανόνες:

- Ο αρχικός χαρακτήρας πρέπει να είναι γράμμα ή underscore (_)
- Οι υπόλοιποι χαρακτήρες μπορεί να είναι γράμματα, αριθμοί ή underscore(_)

- Το μέγιστο μέγεθος μιας μεταβλητής μπορεί να είναι 32 χαρακτήρες
- Παίζει ρόλο ο τρόπος γραφής πεζών-κεφαλαίων. Έτσι τα ονόματα rate, RATE και Rate είναι 3 διαφορετικές μεταβλητές
- Δεν μπορούν να χρησιμοποιούνται δεσμευμένες λέξεις της γλώσσας ως μεταβλητές

Όλες οι μεταβλητές που χρησιμοποιούνται σε ένα πρόγραμμα πρέπει υποχρεωτικά να δηλώνονται πριν τη χρήση τους γιατί αλλιώς θα έχουμε συντακτικό λάθος. Η δήλωση μιας μεταβλητής γίνεται με την ακόλουθη εντολή: τύπος όνομα μεταβλητής;

Παραδείγματα δηλώσεων

int x	η μεταβλητή x δηλώνεται ακέραιου τύπου
float y	η μεταβλητή y δηλώνεται πραγματικού τύπου
char z	η μεταβλητή z δηλώνεται τύπου χαρακτήρα
double k	η μεταβλητή k δηλώνεται τύπου ***double***

Όλες οι μεταβλητές σε ένα πρόγραμμα σε C++ πρέπει να δηλώνονται πριν από τη χρήση τους σε κάποιο τύπο δεδομένων. Ο τύπος δεδομένων καθορίζει το σύνολο τιμών από το οποίο μπορεί να πάρει τιμές μια μεταβλητή. Οι τύποι δεδομένων διακρίνονται σε απλούς ή ενσωματωμένους και σε σύνθετους. Οι απλοί τύποι δεδομένων στη C++ περιγράφονται στο κεφάλαιο 5.3.1.

5.3.1 Τύποι Δεδομένων στη C++

Οι τύποι δεδομένων δείχνουν το σύνολο τιμών από το οποίο μπορεί να πάρει τιμές μια μεταβλητή. Στη γλώσσα C++ υπάρχουν οι ακόλουθοι τύποι:

Τύπος	Μνήμη	Περιγραφή
int	2 bytes	Προσημασμένες ακέραιες τιμές από –32.768 έως +32.767
short int ή ***short***	1 byte	Προσημασμένες ακέραιες τιμές από –128 έως +127
long int ή ***long***	4 bytes	Προσημασμένες ακέραιες τιμές από –2.147.483.648 έως +2.147.483.647
unsigned int ή ***unsigned***	2 bytes	Απρόσημες ακέραιες τιμές από 0 έως 65.535
char	1 byte	Το σύνολο των χαρακτήρων. Ο χαρακτήρας αυτός μπαίνει μέσα σε αποστρόφους π.χ. 'A', '!' κ.λ.π.
signed char	1 byte	Ένας χαρακτήρας με τιμές από –128 έως 127
unsigned char	1 byte	Ένας χαρακτήρας με τιμές από 0 έως 255
float	4 bytes	Δεκαδικές τιμές από 3.4E-38 έως 3.4E+38 και –3.4E-38 έως -3.4E+38
double	8 bytes	Δεκαδικές τιμές από 1.7E-308 έως 1.7E+308 και –1.7E-308 έως -1.7E+308
bool	1 byte	Τιμές ***true***, false (μόνο στη C++)

6 Δομή προγράμματος σε C++

Ένα πρόγραμμα σε C++ έχει την ακόλουθη δομή:

#***include*** <***iostream***.h>	ενσωματώνει στο πρόγραμμα το αρχείο επικεφαλίδας ***iostream***.h
void main()	επικεφαλίδα προγράμματος
{	έναρξη προγράμματος
δήλωση μεταβλητών	
εντολή 1;	
εντολή 2;	
…….	
εντολή n;	
}	τέλος προγράμματος

7 Βασικοί Κανόνες της Γλώσσας

Στη γλώσσα C++ ισχύουν οι ακόλουθοι κανόνες όσον αφορά τη συγγραφή ενός προγράμματος:

- Κάθε πρόγραμμα ξεκινά με τη συνάρτηση main(). Η συνάρτηση αυτή είναι μοναδική και μπορεί να βρίσκεται οπουδήποτε μέσα σε ένα πρόγραμμα.
- Κάθε πρόγραμμα ξεκινά και τελειώνει με τα σύμβολα { και }
- Κάθε εντολή τελειώνει με ένα ελληνικό ερωτηματικό (;)
- Η συνάρτηση main() πρέπει να επιστρέφει μια τιμή που να αντανακλά την κατάσταση του προγράμματος. Όταν επιστρέφεται η τιμή 0 στο λειτουργικό σύστημα τότε αυτό είναι ένδειξη ότι το πρόγραμμα τερματίζεται κανονικά χωρίς λάθη

8 Εντολές Εισόδου – Εξόδου

8.1 Εντολή Εισόδου

Με την εντολή εισόδου μπορούμε να εισάγουμε τιμές σε μεταβλητές. Έχει την ακόλουθη μορφή:

```
cin>>μεταβλητή;
```

για να εισάγουμε μια τιμή σε μια μεταβλητή ή

```
cin>>μεταβλητή1>>μεταβλητή2>>…;
```

για να εισάγουμε πολλές τιμές σε διαφορετικές μεταβλητές

8.2 Εντολή Εξόδου

Με την εντολή εξόδου μπορούμε να εμφανίσουμε στην οθόνη τιμές μεταβλητών και σχόλια. Έχει την ακόλουθη μορφή:

```
cout<<"σχόλιο";
```

για να εκτυπώσουμε ένα σχόλιο ή

```
cout<<μεταβλητή;
```

για να εκτυπώσουμε ένα αποτέλεσμα ή

```
cout<<"σχόλιο"<<μεταβλητή;
```

για να εκτυπώσουμε ένα σχόλιο και ένα αποτέλεσμα μαζί

8.3 Εντολή #*include*

Γράφοντας την εντολή:

- #***include*** <filename> ψάχνει για το αρχείο που καθορίζουμε μόνο στον ειδικό κατάλογο για ***include*** files
- #***include*** "filename" ψάχνει για το αρχείο που καθορίζουμε και στον τρέχοντα κατάλογο αλλά και στο ***include*** directory

8.4 Εντολή #*define* και const

Με την εντολή:

```
#define DAYS_IN_WEEK 7
```

ορίζουμε τη σταθερά DAYS_IN_WEEK να παίρνει την τιμή 7. Αυτές οι σταθερές κληρονομούνται από τη γλώσσα C και ονομάζονται macro-based σταθερές. Οι σταθερές αυτές δηλώνονται πάντα έξω από το main()

Με την εντολή:

```
const int DAYS_IN_WEEK=7;
```

ορίζουμε μια formal constant. Ορίζοντας μια formal constant μέσα στο main() έχει διαφορά από το να δηλωθεί έξω από το main() διότι στην πρώτη περίπτωση η σταθερά αυτή είναι τοπική (local) για τη συνάρτηση main() ενώ στη δεύτερη περίπτωση η σταθερά είναι γενική (global) δηλαδή διαθέσιμη σε οποιαδήποτε άλλη συνάρτηση του προγράμματος.

8.5 Τελεστές

8.5.1 Σχεσιακοί Τελεστές

Οι σχεσιακοί τελεστές χρησιμοποιούνται για να δημιουργήσουν συνθήκες. Στη γλώσσα C++ είναι οι ακόλουθοι:

Σχεσιακοί Τελεστές
< (μικρότερο)
<=(μικρότερο ή ίσο)
> (μεγαλύτερο)
>=(μεγαλύτερο ή ίσο)
== (ισότητα)
!= (διάφορο)

Σημείωση

Πρέπει να τονίσουμε τη διαφορά ανάμεσα στην εντολή καταχώρισης η οποία συμβολίζεται με το = και τον τελεστή της ισότητας ο οποίος συμβολίζεται με ==.

8.5.2 Λογικοί Τελεστές

Οι λογικοί τελεστές χρησιμοποιούνται προκειμένου να συνδυάσουν πολλές απλές συνθήκες μέσα σε μια σύνθετη συνθήκη. Η τιμή μιας συνθήκης απλής ή σύνθετης είναι είτε ***true*** (αληθής) είτε false (ψευδής). Στη γλώσσα C++ οι λογικοί τελεστές είναι οι ακόλουθοι είναι οι ακόλουθοι:

Λογικοί Τελεστές	Προτεραιότητα
! (NOT)	1
&& (AND)	2
|| (OR)	3

9 Εντολές Επιλογής

9.1 Απλή επιλογή

Στην απλή επιλογή ελέγχεται μια συνθήκη και εφόσον είναι αληθής τότε εκτελείται μια ομάδα εντολών 1, ενώ αν είναι ψευδής τότε εκτελείται μια ομάδα εντολών 2. Έχει την ακόλουθη σύνταξη:

```
if (συνθήκη)
{
    ομάδα εντολών 1;
}
else
{
    ομάδα εντολών 2;
}
```

9.2 Περιορισμένη επιλογή

Στην περιορισμένη επιλογή ελέγχεται μια συνθήκη και εφόσον είναι αληθής τότε εκτελείται μια ομάδα εντολών 1 ενώ αν είναι ψευδής τότε δεν εκτελείται τίποτα και το πρόγραμμα συνεχίζει απλώς στην επόμενη εντολή μετά το ***if***. Έχει την ακόλουθη σύνταξη:

```
if (συνθήκη)
{
    ομάδα εντολών 1;
}
```

9.3 Εμφωλευμένη επιλογή

Στην εμφωλευμένη επιλογή ελέγχεται μια συνθήκη 1 και εφόσον είναι αληθής τότε εκτελείται μια ομάδα εντολών 1. Αν είναι ψευδής τότε ελέγχεται μια συνθήκη 2 και εφόσον είναι αληθής τότε εκτελείται μια ομάδα εντολών 2. Αν είναι ψευδής τότε ελέγχεται μια συνθήκη 3 κ.ο.κ. Συνολικά ελέγχονται n συνθήκες και εκτελείται μόνο μια ομάδα εντολών. Αν δεν είναι καμία συνθήκη αληθής τότε εκτελείται μια ομάδα εντολών n+1. Έχει την ακόλουθη σύνταξη:

```
if (συνθήκη 1)
{
   ομάδα εντολών 1;
}
else
   if (συνθήκη 2)
   {
      ομάδα εντολών 2;
   }
    else
        ……..
        if (συνθήκη n)
        {
          ομάδα εντολών n;
        }
        else
        {
          ομάδα εντολών n+1;
        }
```

Παρατηρήσεις

1.Σε κάθε εντολή επιλογής η συνθήκη είναι μια λογική παράσταση η οποία παίρνει είτε την τιμή αλήθεια (***true***) είτε την τιμή ψέμα (false). Το 0 αντιστοιχεί στη λογική τιμή αλήθεια και ένας αριθμός ≠0 αντιστοιχεί στην λογική τιμή ψέμα.

2.Κάθε ομάδα εντολών μπορεί να είναι είτε μια μόνο εντολή ή ένα μπλοκ από εντολές μέσα σε άγκιστρα. Αν είναι μόνο μια εντολή τότε μπορούμε να παραλείψουμε τα άγκιστρα.

9.4 Εντολή πολλαπλής επιλογής switch

Η εντολή πολλαπλής επιλογής χρησιμοποιείται όταν ελέγχουμε την τιμή μιας μεταβλητής (όχι συνθήκης) και ανάλογα με την τιμή αυτή εκτελούμε είτε μια ομάδα εντολών 1 είτε μια ομάδα εντολών 2 είτε μια εντολών n. Αν η μεταβλητή δεν πάρει καμία από τις τιμές 1, 2, … n τότε εκτελείται η ομάδα εντολών που βρίσκεται στο default. Η εντολή switch γενικά έχει την ακόλουθη μορφή:

```
switch (μεταβλητή)
{
        case τιμή 1: ομάδα εντολών 1;
                     break;
        case τιμή 2:       ομάδα εντολών 2;
                     break;
        ……..
        case τιμή n: ομάδα εντολών n;
                     break;
        default:           ομάδα εντολών n+1;
```

```
}
```

Σημείωση

Η εντολή ***break*** είναι προαιρετική. Όμως ουσιαστικά είναι επιβεβλημένη η χρησιμοποίηση της γιατί αν απουσιάζει από το τέλος εντολών μιας ομάδας τότε εκτελούνται οι εντολές και της επόμενης ομάδας γεγονός βέβαια που δεν έχει νόημα. Συνεπώς η ***break*** στην εντολή switch απαγορεύει την εκτέλεση των εντολών των υπολοίπων ομάδων πέρα της ομάδας εντολών που θα εκτελεστεί.

10 Εντολές Επανάληψης

Οι εντολές επανάληψης (ανακύκλωσης) προκαλούν την εκτέλεση μιας εντολής ή μιας ομάδας εντολών μέσα σε άγκιστρα είτε ένα προκαθορισμένο αριθμό επαναλήψεων είτε για όσο χρόνο μια συνθήκη είναι αληθής. Στη C++ χρησιμοποιούμε 3 εντολές επανάληψης.

10.1Εντολή επανάληψης *for*

Η εντολή ***for*** επαναλαμβάνει μια ομάδα εντολών συνήθως ένα συγκεκριμένο αριθμό φορών αν και μπορεί να χρησιμοποιηθεί ακόμα και για την επανάληψη μιας ομάδας εντολών για όσο χρόνο μια συνθήκη είναι αληθής. Έχει την ακόλουθη σύνταξη:

```
for (αρχικές τιμές; συνθήκη; μεταβολές)
{
    ομάδα εντολών;
}
```

Οι αρχικές τιμές μπορεί να είναι μια ή περισσότερες εντολές που διαχωρίζονται με κόμμα και έχουν στόχο να εκτελεστούν κάποιες εντολές πριν αρχίσει η επανάληψη. Η συνθήκη μπορεί να είναι απλή είτε να συγκροτείται από επιμέρους συνθήκες χωριζόμενες με κόμμα (σύνθετη) και για όσο χρόνο είναι αληθής εκτελείται η ομάδα εντολών ανάμεσα στα άγκιστρα. Μετά από κάθε επανάληψη εκτελούνται οι εντολές που βρίσκονται στο τμήμα μεταβολές οι οποίες έχουν σαν σκοπό να μεταβάλλουν την τιμή της συνθήκης. Στη συνέχεια ελέγχεται πάλι η νέα τιμή της συνθήκης και εφόσον εξακολουθεί να είναι αληθής επαναλαμβάνεται η εκτέλεση

της ομάδας εντολών. Όταν η συνθήκη γίνει ψευδής τότε η εντολή ***for*** τερματίζεται και το πρόγραμμα συνεχίζει στην επόμενη εντολή μετά την ***for***.

10.2Εντολή Επανάληψης *while*

Η εντολή ***while*** χρησιμοποιείται όπως και η ***for*** για να επαναλάβουμε την εκτέλεση μιας εντολής ή μιας ομάδας εντολών μέσα σε άγκιστρα είτε ένα συγκεκριμένο αριθμό φορών είτε για όσο χρόνο μια συνθήκη είναι αληθής. Πρώτα ελέγχεται η συνθήκη και εφόσον είναι αληθής τότε εκτελείται η ομάδα εντολών. Για όσο χρόνο η συνθήκη (απλή ή σύνθετη) είναι αληθής (***true***) τότε επαναλαμβάνεται η εκτέλεση της ομάδας εντολών. Όταν η συνθήκη γίνει ψευδής η εντολή ***while*** τερματίζεται και το πρόγραμμα συνεχίζει στην επόμενη εντολή μετά την ***while***. Έχει την ακόλουθη σύνταξη:

```
while (συνθήκη)
{
    ομάδα εντολών;
}
```

10.3Εντολή επανάληψης do..*while*

Η εντολή do ***while*** είναι αντίστοιχη με την εντολή ***while*** με τη διαφορά ότι πρώτα εκτελείται η ομάδα εντολών και μετά ελέγχεται η συνθήκη, δηλαδή η ομάδα εντολών θα εκτελεστεί τουλάχιστον μια φορά ακόμα και αν η συνθήκη είναι ψευδής. Έχει την ακόλουθη σύνταξη:

```
do
{
    ομάδα εντολών;
}
while (συνθήκη);
```

10.4Εντολή *break*

Στις εντολές επανάληψης ***for***, ***while*** και do ***while*** μπορούμε να παρεμβάλουμε μέσα στην ομάδα εντολών την εντολή ***break*** η οποία προκαλεί τη διακοπή της εκτέλεσης των υπολοίπων εντολών της ομάδας και κάνει άμεση έξοδο από την επανάληψη. Έχει την ακόλουθη σύνταξη:

```
Εντολή επανάληψης
{
  break;
}
```

10.5Εντολή continue

Η εντολή continue τοποθετείται επίσης μέσα σε μια επανάληψη και προκαλεί τη διακοπή της εκτέλεσης των υπολοίπων εντολών της ομάδας αλλά δεν κάνει έξοδο από το βρόχο, απλώς επιβάλλει επανέλεγχο της συνθήκης. Έχει την ακόλουθη σύνταξη:

```
Εντολή επανάληψης
{
  continue;
```

```
}
```

11 Πίνακες στη C++

Οι πίνακες είναι διαδοχικές θέσεις μνήμης στις οποίες τοποθετούνται δεδομένα του ιδίου τύπου. Στα δεδομένα ενός πίνακα αναφερόμαστε με ένα δείκτη θέσης (μονοδιάστατος πίνακας) ή με ένα ζεύγος δεικτών θέσης (δισδιάστατος πίνακας).

11.1Μονοδιάστατοι πίνακες

Τα στοιχεία του πίνακα είναι όπως αναφέραμε του ιδίου τύπου και τοποθετούνται σε διαδοχικές θέσεις μνήμης. Λέγεται μονοδιάστατος ή γραμμικός πίνακας γιατί όλα τα στοιχεία του είναι τοποθετημένα σε μια γραμμή όπως φαίνεται και στο ακόλουθο σχήμα:

0	**1**	**2**	**3**	**4**	**5**
12	-7	45	7	4	8

Με τα έντονα γράμματα φαίνονται οι θέσεις του πίνακα. Παρατηρούμε ότι οι θέσεις ενός μονοδιάστατου πίνακα αριθμούνται από τη θέση 0.

11.1.1Δήλωση μονοδιάστατου πίνακα

Η δήλωση ενός μονοδιάστατου πίνακα γίνεται με την ακόλουθη εντολή:

```
τύπος όνομα_πίνακα [πλήθος στοιχείων];
```

Για παράδειγμα γράφοντας σε ένα πρόγραμμα τη δήλωση ***int*** x[5]; ορίζουμε ένα πίνακα με όνομα x που συγκροτείται από 5 ακεραίους σε διαδοχικές θέσεις μνήμης.

0	1	2	3	4
x[0]	x[1]	x[2]	x[3]	x[4]

Γενικά το στοιχείο ενός μονοδιάστατου πίνακα στη θέση i συμβολίζεται σαν Πίνακας[i][1]. Κάθε μία από τις διαδοχικές θέσεις μνήμης που συγκροτούν ένα πίνακα αποτελούν ένα στοιχείο του πίνακα.

Ο πίνακας δεσμεύει στατικά μνήμη δηλαδή όλη η απαιτούμενη μνήμη γιαυτόν δεσμεύεται κατά τη διάρκεια του συντακτικού ελέγχου (compilation) του προγράμματος και δεν είναι δυνατόν να αλλάξει κατά τη διάρκεια εκτέλεσης του.

11.1.2 Δήλωση μεγέθους μονοδιάστατου πίνακα

Η δήλωση του μεγέθους ενός μονοδιάστατου πίνακα γίνεται με την ακόλουθη εντολή:

const int n=τιμή;

Με τη λέξη const δηλώνουμε σταθερές διότι το μέγεθος του πίνακα πρέπει να είναι σταθερό.

11.1.3 Αρχικές τιμές μονοδιάστατου πίνακα

Μαζί με την δήλωση ενός πίνακα μπορούμε να δώσουμε και αρχικές τιμές στα στοιχεία του πίνακα με εντολή της μορφής:

int x[5]= {10, 25, 15, 35,5};

τότε ο πίνακας x θα έχει τα ακόλουθα στοιχεία:

[1] Τα στοιχεία ενός μονοδιάστατου πίνακα αριθμούνται πάντα από το μηδέν

0	1	2	3	4
10	25	15	35	5

Δηλώνοντας κάποιο πίνακα χωρίς αρχικές τιμές τα στοιχεία του έχουν τιμές απροσδιόριστες.

11.1.4 Προσαρμοζόμενη διάσταση μονοδιάστατου πίνακα

Όταν δηλώνουμε ένα πίνακα με αρχικές τιμές μπορούμε να μην προσδιορίζουμε το πλήθος των στοιχείων του αλλά να αφήνουμε τη C++ να προσαρμόζει ανάλογα με το πλήθος των αρχικών τιμών και το πλήθος των στοιχείων του πίνακα. Τότε λέμε ότι ορίζουμε τον πίνακα με προσαρμοζόμενη διάσταση. Για παράδειγμα επιτρέπεται να δηλώσουμε ένα πίνακα με 5 στοιχεία γράφοντας:

```
int x[] = {10,25,15,35,5};
```

11.1.5 Εκχώρηση τιμών στα στοιχεία μονοδιάστατου πίνακα

Από τη στιγμή που έχουμε δηλώσει έναν πίνακα (με αρχικές τιμές ή χωρίς αρχικές τιμές) δεν υπάρχει τρόπος να εκχωρήσουμε, με μια εντολή, τιμές σε όλα τα στοιχεία του. Δηλαδή είναι λάθος να γράψουμε: x ={4, 8 12, 5, 20}; Οι τιμές καταχωρούνται μόνο σε μεμονωμένα στοιχεία του πίνακα κάθε φορά.
Για παράδειγμα δίνοντας τις καταχωρίσεις:

- x[0]=4;
- x[1]=8;
- x[2]=12;
- x[3]=5;

- x[4]=20;

ο πίνακας x θα έχει τα ακόλουθα στοιχεία:

0	**1**	**2**	**3**	**4**
4	8	12	3	20

11.1.6 Μέγεθος μονοδιάστατου πίνακα

Αν έχουμε ορίσει ένα πίνακα τότε το πλήθος όλων των bytes της μνήμης που αφιερώνονται για τα στοιχεία του πίνακα προσδιορίζεται μέσω του τελεστή sizeof. Για παράδειγμα μετά την δήλωση ***int*** x[]={1,2,3,4,5}; η εντολή ***cout***<<"Size of array x in bytes is "<<***sizeof***(x)<<***endl***; θα μας δώσει εκτυπώσει 10 (5 ακέραιοι επί 2 bytes ο καθένας). Το μέγεθος ενός μονοδιάστατου πίνακα δηλαδή το πλήθος των στοιχείων δηλώνεται συνήθως με τη βοήθεια μιας καθολικής σταθεράς όπως φαίνεται στο ακόλουθο παράδειγμα:

```
const n=3; //Πλήθος στοιχείων
```

11.1.7 Πλήθος στοιχείων μονοδιάστατου πίνακα

Για να υπολογίσουμε μέσα στο πρόγραμμα το πλήθος στοιχείων ενός δηλωθέντος πίνακα θα διαιρέσουμε το μέγεθος του πίνακα (σε bytes) δια του πλήθους των bytes που αντιστοιχούν στον τύπο του κάθε στοιχείου του. Για παράδειγμα μετά την δήλωση ***int*** x[]={1,2,3,4,5}; η εντολή:

```
cout<<"# of elements of array x is <<sizeof(x)/sizeof(int)<<endl;
```

θα μας δώσει εκτύπωση 5 (5 στοιχεία στον πίνακα x).

11.1.8 Γενικοί Αλγόριθμοι Μονοδιάστατων Πινάκων στη C++

11.1.8.1 Διάβασμα Μονοδιάστατου Πίνακα

Για να διαβάσουμε (γεμίσουμε) ένα μονοδιάστατο πίνακα με τιμές γράφουμε τον ακόλουθο κώδικα:[2]

```
for (i=0;i<n;i++)
{
    cout<<"Δώσε τιμή στο "<<i<<" στοιχείο";
    cin>>x[i];
}
```

Παρατήρηση: Σε κάθε επανάληψη διαβάζουμε ένα στοιχείο του πίνακα.

11.1.8.2 Εκτύπωση Μονοδιάστατου Πίνακα

Για να εκτυπώσουμε (εμφανίσουμε) τα στοιχεία ενός μονοδιάστατου πίνακα στην οθόνη γράφουμε τον ακόλουθο κώδικα:

```
for (i=0;i<n; i++)
    cout<<x[i]<<endl;
```

Η εκτύπωση των στοιχείων του πίνακα θα γίνει σε μια στήλη. Αν θέλουμε ο πίνακας να εκτυπωθεί σε μια γραμμή με αποστάσεις μεταξύ των στοιχείων του, τότε γράφουμε τον ακόλουθο κώδικα:

[2] Υποθέτουμε ότι σε όλες τις λειτουργίες που αναφέρουμε ότι έχει προηγηθεί η δήλωση ***int*** x[5];

```
for (i=0; i<n; i++)
    cout<<setw(5)<<x[i];
```

Παρατήρηση: Σε κάθε επανάληψη εκτυπώνουμε ένα στοιχείο του πίνακα. Επίσης η συνάρτηση setw που ορίζει το πλάτος εκτύπωσης υπάρχει στο header file iomanip.h

11.1.8.3 Υπολογισμός μέσου όρου Μονοδιάστατου Πίνακα

Για να υπολογίσουμε το μέσο όρο των στοιχείων ενός μονοδιάστατου πίνακα αθροίζουμε πρώτα όλα τα στοιχεία του πίνακα και στη συνέχεια διαιρούμε το άθροισμα με το συνολικό πλήθος των στοιχείων του πίνακα.

Αυτή η λειτουργία υλοποιείται με τον ακόλουθο κώδικα:

```
for (i=0; i<n; i++)
     sum+=x[i];

mo=(float)sum/n;
cout<<"Μέσος όρος "<<mo<<endl;
```

Παρατήρηση: Ο μέσος όρος του πίνακα υπολογίζεται μόνο μια φορά στο τέλος της επανάληψης. Επίσης ο αθροιστής των στοιχείων του πίνακα πρέπει να αρχικοποιηθεί με μηδέν. Αυτό είναι εφικτό είτε στη δήλωση του αθροιστή sum (δηλαδή να κάνουμε ταυτόχρονη δήλωση και αρχικοποίηση όπως π.χ. int sum=0) είτε με εντολή καταχώρισης sum=0 πριν την επανάληψη for

11.1.8.4 Υπολογισμός πλήθους άρτιων-περιττών στοιχείων

Για να υπολογίσουμε το πλήθος άρτιων και περιττών στοιχείων όλου του πίνακα χρησιμοποιούμε 2 μετρητές, τον μετρητή art ο οποίος υπολογίζει το πλήθος των άρτιων στοιχείων και το μετρητή per ο οποίος υπολογίζει αντίστοιχα το πλήθος των περιττών στοιχείων του πίνακα. Οι 2 μετρητές πρέπει να αρχικοποιηθούν με μηδέν πριν την επανάληψη. Η λειτουργία αυτή υλοποιείται με το ακόλουθο τμήμα κώδικα:

```
for (i=0;i<n; ++)
    if (x[i]%2==0)
      art++;
    else
      per++;

cout<<"Πλήθος άρτιων "<<art<<endl;
cout<<"Πλήθος περιττών "<<per<<endl;
```

Παρατήρηση: Για να υπολογίσουμε το πλήθος άρτιων και περιττών στοιχείων του πίνακα πρέπει ο είτε ο πίνακας να περιέχει ακέραια στοιχεία είτε να μετατρέψουμε τα στοιχεία του σε ακέραια (με cast operator) διότι ο αριθμητικός τελεστής % στη C++ εφαρμόζεται μόνο σε ακέραια στοιχεία.

11.1.8.5 Εύρεση μέγιστου-ελάχιστου στοιχείου πίνακα και των θέσεων τους σε μονοδιάστατο πίνακα

Για να βρούμε το μέγιστο και το ελάχιστο στοιχείο του πίνακα και τις θέσεις τους στον πίνακα υποθέτουμε αρχικά ότι ο μέγιστο και το ελάχιστο στοιχείο του πίνακα είναι το πρώτο (αποθηκεύουμε το μέγιστο και το ελάχιστο στοιχείο του πίνακα στις μεταβλητές max και min αντίστοιχα). Επίσης στις μεταβλητές maxp και minp αποθηκεύουμε τη θέση του μέγιστου και του ελάχιστο στοιχείο του πίνακα αντίστοιχα. Η λειτουργία αυτή υλοποιείται με το ακόλουθο τμήμα κώδικα:

```
max= min=x[0];
maxp= minp=0;

for (i=0;i<n; i++)
{
    if (x[i]>max)
    {
        max=x[i];
        maxp=i;
    }

    if (x[i]<min)
    {
        min=x[i];
        minp=i;
     }
}
```

```
cout<<"Μέγιστο = "<<max<<" στη θέση "<<maxp+1<<endl;
cout<<"Ελάχιστο = "<<min<<" στη θέση "<<minp+1<<endl;
```

Παρατήρηση: Η εκτύπωση των αποτελεσμάτων γίνεται μια φορά στο τέλος της επανάληψης

11.1.9 Αλγόριθμοι Αναζήτησης σε Μονοδιάστατους Πίνακες

11.1.9.1 Σειριακή αναζήτηση στοιχείου σε πίνακα και εύρεση όλων των θέσεων του καθώς και του πλήθους εμφανίσεων του

Αρχικά διαβάζουμε το στοιχείο αναζήτησης και μετά το ελέγχουμε (συγκρίνουμε) με κάθε στοιχείο του πίνακα. Κάθε φορά που το βρίσκουμε στον πίνακα τυπώνουμε τη θέση που το συναντήσαμε και αυξάνουμε ένα μετρητή για να υπολογίσουμε ταυτόχρονα και το πλήθος των εμφανίσεων του. Η λειτουργία αυτή υλοποιείται με το ακόλουθο τμήμα κώδικα:

```
cout<<"Δώσε στοιχείο αναζήτησης"<<endl;
cin>>y;
cout<<endl;

for (i=0;i<n; i++)
    if (y==x[i])
    {
      a++;
      cout<<"Το στοιχείο βρέθηκε στη θέση "<<i+1<<endl;
    }
```

```
cout<<"Το στοιχείο βρέθηκε συνολικά "<<a<<" φορές"<<endl;
```

Παρατήρηση: Ο μετρητής των εμφανίσεων του στοιχείου στον πίνακα τυπώνεται μια φορά στο τέλος της επανάληψης. Επίσης αν το στοιχείο αναζήτησης είναι διαφορετικό από ένα στοιχείο του πίνακα, τότε δεν εκτελούμε κάποια ενέργεια, απλά προχωρούμε στον επόμενο έλεγχο.

11.1.9.2 Σειριακή αναζήτηση στοιχείου σε πίνακα και εύρεση μόνο της $1^{ης}$ θέσης του στοιχείου στον πίνακα

Αρχικά διαβάζουμε το στοιχείο αναζήτησης όπως και πριν και μετά το ελέγχουμε (δηλαδή το συγκρίνουμε) με κάθε στοιχείο του πίνακα. Όταν το συναντήσουμε για $1^{η}$ φορά στον πίνακα (αν βέβαια το συναντήσουμε) τυπώνουμε τη θέση που το βρήκαμε και τερματίζουμε την αναζήτηση διότι δεν μας ενδιαφέρουν οι υπόλοιπες θέσεις στις οποίες πιθανώς να υπάρχει το στοιχείο. Η λειτουργία αυτή υλοποιείται με το ακόλουθο τμήμα κώδικα:

```
cout<<"Δώσε στοιχείο αναζήτησης"<<endl;
cin>>y;
found=false;

for (i=0;i<n; i++)
    if (y==x[i])
    {
```

```
        cout<<"Το στοιχείο βρέθηκε για 1η φορά στη θέση
"<<i+1<<endl;
        found=true;
        break;
    }

if (found==false)
    cout<<"Το στοιχείο δεν υπάρχει στον πίνακα "<<endl;
```

Παρατήρηση: Με τη λογική μεταβλητή found ελέγχουμε την ύπαρξη του στοιχείου στον πίνακα. Αρχικά υποθέτουμε ότι το στοιχείο δεν υπάρχει στον πίνακα θέτοντας στη μεταβλητή found την τιμή false. Αν το στοιχείο βρεθεί στον πίνακα τότε στην found θέτουμε true. Στο τέλος ελέγχουμε την τιμή της found για να ελέγξουμε τελικά αν το στοιχείο βρέθηκε ή όχι στον πίνακα. Επίσης η εντολή break τερματίζει την επανάληψη του for ώστε να σταματήσουμε τον έλεγχο στις υπόλοιπες θέσεις του πίνακα αφού το στοιχείο βρέθηκε.

11.1.9.3 Σειριακή αναζήτηση στοιχείου σε πίνακα και εύρεση μόνο της τελευταίας θέσης του στοιχείου στον πίνακα

Αρχικά διαβάζουμε το στοιχείο αναζήτησης όπως και πριν και μετά το ελέγχουμε (δηλαδή το συγκρίνουμε) με κάθε στοιχείο του πίνακα. Όταν τελειώσουμε την αναζήτηση στον πίνακα, τυπώνουμε μόνο την τελευταία θέση στην οποία βρέθηκε το στοιχείο (αν βέβαια το στοιχείο υπάρχει στον πίνακα), οι υπόλοιπες θέσεις στις οποίες (πιθανώς) το συναντήσαμε δεν μας ενδιαφέρουν και γιαυτό

τις αγνοούμε (δηλαδή δεν τις εκτυπώνουμε). Η λειτουργία αυτή υλοποιείται με το ακόλουθο τμήμα κώδικα:

```
cout<<"Δώσε στοιχείο αναζήτησης"<<endl;
cin>>y;
found=false;

for (i=n-1;i>=0; i--)
    if (y==x[i])
    {
      cout<<"Το στοιχείο βρέθηκε για τελευταία φορά στη θέση
"<<i+1<<endl;
      found=true;
    }

if (found==false)
    cout<<"Το στοιχείο δεν υπάρχει στον πίνακα "<<endl;
```

Παρατήρηση: Η αναζήτηση του στοιχείου αρχίζει από τέλος του πίνακα ώστε να εντοπίσουμε μόνο την τελευταία θέση που το στοιχείο υπάρχει στον πίνακα. Όταν το συναντήσουμε η επανάληψη τερματίζεται

11.1.9.4 Δυαδική αναζήτηση στοιχείου σε πίνακα

Αρχικά διαβάζουμε πάλι το στοιχείο αναζήτησης και υποθέτουμε ότι δεν υπάρχει στον πίνακα. Στη μεταβλητή left την αρχική θέση του πίνακα δηλαδή το 0 ενώ στη μεταβλητή right καταχωρούμε την

τελική θέση του πίνακα. Η επανάληψη αυτή εκτελείται είτε μέχρι το left να γίνει μεγαλύτερο από το right (οπότε αυτό σημαίνει ότι το στοιχείο δεν υπάρχει στον πίνακα) είτε μέχρι να βρούμε το στοιχείο στον πίνακα. Σε κάθε επανάληψη εξετάζουμε το μεσαίο στοιχείο του πίνακα το οποίο καταχωρείται στη μεταβλητή mid και αν το στοιχείο αναζήτησης δεν είναι ίσο με το στοιχείο στη θέση mid, τότε εκμεταλλευόμαστε το γεγονός ότι ο πίνακας είναι ταξινομημένος περιορίζοντας την αναζήτηση είτε στο επάνω μισό (αν το στοιχείο αναζήτησης είναι μεγαλύτερο του μεσαίου στοιχείου του πίνακα) είτε στο κάτω μισό (αν το στοιχείο αναζήτησης είναι μικρότερο του μεσαίου στοιχείου του πίνακα). Η λειτουργία αυτή υλοποιείται με το ακόλουθο τμήμα κώδικα:

```
cout<<"Δώσε στοιχείο αναζήτησης"<<endl;
cin>>y;

found= false;
l=0;
r=n-1;

while (found==false && l<=r)
{
        mid=(l+r)/2;

        if (x[mid]==y)
        {
            cout<<"Βρέθηκε στη θέση "<<mid+1<<endl;
            found= true;
        }
```

```
        else
            if (y>x[mid])
                l=mid+1;
            else
                r=mid-1;
}

if (found== false)
    cout<<"Το στοιχείο δεν βρέθηκε"<<endl;
```

Παρατήρηση: Ο αλγόριθμος δυαδικής αναζήτησης εφαρμόζεται μόνο σε ταξινομημένους πίνακας και είναι πιο γρήγορος από τον αλγόριθμο σειριακής αναζήτησης αφού εκτελείται σε χρόνο O(logn) ενώ ο αλγόριθμος σειριακής αναζήτησης σε χρόνο O(n). Επίσης αν το στοιχείο αναζήτησης υπάρχει πολλές φορές στον πίνακα τότε ο αλγόριθμος αυτός εντοπίζει μόνο την $1^{η}$ εμφάνιση του στοιχείου στον πίνακα.

11.1.10 Αλγόριθμοι Ταξινόμησης Μονοδιάστατων Πινάκων

11.1.10.1 Αλγόριθμος φυσαλίδας (bubblesort)

Η 1η επανάληψη που εκτελείται αφορά τα περάσματα στον πίνακα. Σε κάθε πέρασμα συγκρίνουμε τα στοιχεία του πίνακα (με την 2η επανάληψη) και ταξινομούμε το μεγαλύτερο κάθε φορά στοιχείο του πίνακα τοποθετώντας το στο τέλος του πίνακα. Η λειτουργία αυτή υλοποιείται με το ακόλουθο τμήμα κώδικα:

```
for (b=1;b<n; b++)
    for (i=0; i<n-b; i++)
        if (x[i]>x[i+1])
        {
            temp=x[i];
            x[i]=x[i+1];
            x[i+1]=temp;
        }
```

Παρατηρήσεις:

- Ο αλγόριθμος αυτός εκτελεί περιττά βήματα όταν ο πίνακας είναι ήδη ταξινομημένος ή μερικώς ταξινομημένος
- Η ταξινόμηση που εκτελείται είναι σε αύξουσα σειρά. Αν θέλουμε να γίνει σε φθίνουσα σειρά τότε αλλάζει η φορά της ανισότητας σε < δηλαδή if x[i]<x[i+1]

11.1.10.2 Βελτιωμένος αλγόριθμος φυσαλίδας (bubblesort)

Ο αλγόριθμος αυτός βελτιώνει το χρόνο του προηγούμενου εκτελώντας ακριβώς όσα περάσματα απαιτούνται Η λειτουργία αυτή υλοποιείται με το ακόλουθο τμήμα κώδικα:

```
sorted= false;
b=1;

while (b<n && sorted==false)
{
        sorted= true;

        for (i=0; i<n-b; i++)
            if (x[i] > x[i+1])
            {
                temp= x[i];
                x[i]= x[i+1];
                x[i+1]= temp;
                sorted= false;
            }
        b++;
}
```

Παρατήρηση: Ο αλγόριθμος αυτός δεν εκτελεί περιττά βήματα στον πίνακα διότι τερματίζεται στο βήμα που δεν θα χρειαστεί καμία εναλλαγή στοιχείων.

11.1.10.3 Ταξινόμηση Επιλογής (Selectionsort)

Ο αλγόριθμος αυτός εκτελεί N-1 βήματα όπου N είναι το μέγεθος του πίνακα. Σε κάθε βήμα ταξινομεί ένα στοιχείο του πίνακα με τον εξής τρόπο: υποθέτει στο βήμα i ότι το i-οστο στοιχείο του πίνακα είναι το ελάχιστο. Στη συνέχεια βρίσκει το μικρότερο στοιχείο από τα υπόλοιπα στοιχεία του πίνακα και το εναλλάσσει με το στοιχείο που είχε θέσει αρχικά ως μικρότερο σε αυτό το βήμα.

Ο αλγόριθμος αυτός υλοποιείται με το ακόλουθο τμήμα κώδικα:

```
for (i=0; i<n; i++)
{
     min= x[i];
     minp= i;

     for (j=i+1;j<n; j++)
          if (x[j]<min)
          {
             min=x[j];
             minp=j;
          }
          x[minp]=x[i];
          x[i]=min;
}
```

Παρατήρηση: Ο αλγόριθμος επιλογής έχει χρόνο χειρότερης περίπτωσης $O(n^2)$.

11.1.10.4 Ταξινόμηση Εισαγωγής (Insertionsort)

Ο αλγόριθμος αυτός εκτελεί N-1 βήματα όπου N είναι το μέγεθος του πίνακα. Σε κάθε βήμα ταξινομεί ένα στοιχείο του πίνακα με τον εξής τρόπο: αν βρισκόμαστε π.χ. στο i-οστο βήμα εξετάζουμε όλα τα στοιχεία αριστερά από αυτό και όσο υπάρχουν μικρότερα από αυτό, τα μεταθέτουμε μια θέση δεξιά (right shift) παρεμβάλλοντας (εισάγοντας) το i-οστο στοιχείο του βήματος στην κατάλληλη θέση.

Ο αλγόριθμος αυτός υλοποιείται με το ακόλουθο τμήμα κώδικα:

```
for (i=1;i<n;i++)
{
    p=x[i];

    while (i!=0 && p<x[i-1])
    {
            x[i]=x[i-1];
            i--;
    }

    x[i]=p;
}
```

Παρατήρηση: Ο αλγόριθμος εισαγωγής έχει χρόνο χειρότερης περίπτωσης επίσης $O(n^2)$.

11.1.10.5 Αναδρομικός Αλγόριθμος Quicksort

Ο αλγόριθμος αυτός είναι ο πιο γρήγορος αλγόριθμος ταξινόμησης και λειτουργεί ως εξής: εντοπίζουμε το μεσαίο στοιχείο του πίνακα (το ονομάζουμε στοιχείο pivot) και μεταθέτουμε κατάλληλα τα υπόλοιπα στοιχεία του πίνακα έτσι ώστε όλα τα στοιχεία αριστερά από το μεσαίο στοιχείο του πίνακα να είναι μικρότερα από αυτό ενώ όλα τα στοιχεία δεξιά από το μεσαίο στοιχείο του πίνακα να είναι μεγαλύτερα από αυτό. Άρα έτσι έχουμε ταξινομήσει το μεσαίο στοιχείο του πίνακα. Στη συνέχεια ο αλγόριθμος αυτός εκτελείται αναδρομικά για το αριστερό και το δεξιό τμήμα του πίνακα και έτσι ταξινομείται συνολικά όλος ο πίνακας.

Ο αλγόριθμος quicksort υλοποιείται με το ακόλουθο τμήμα κώδικα:

```
void quicksort(int L, int R, int x[])
{
      i=L;
      j=R;
      int pivot=x[(L+R)/2];

      do
      {
            while (x[i]<pivot)
                  i++;

            while (x[j]>pivot)
                  j--;
```

```
        if (i<=j)
        {
            temp=x[i];
            a[i]=x[j];
            x[j]=temp;
            i++;
            j--;
        }
    }
    while (i<j);

    if (L<j)
        quicksort(L, j, x);

    if (i<R)
        quicksort(i, R, x);
```

Παρατηρήσεις:

- Οι μεταβλητές i και j έχουν δηλωθεί καθολικά, ενώ η pivot δηλώνεται τοπικά στη συνάρτηση

- Στην επανάληψη ***while*** (x[j]>pivot) j--; εντοπίζουμε το 1^{ο} στοιχείο από δεξιά του μεσαίου που είναι μικρότερο από αυτό

- Στην επανάληψη ***while*** (x[i]<pivot) i++; εντοπίζουμε το 1^{ο} στοιχείο από αριστερά του μεσαίου που είναι μεγαλύτερο από αυτό

- Με τις εντολές:

- temp=x[i];
- x[i]=x[j];
- x[j]=temp;

- εναλλάσσουμε τα στοιχεία x[j] και x[i] ώστε τα ταξινομηθεί το μεσαίο στοιχείο του πίνακα

- Με την εντολή quicksort(L, j, x);καλούμε αναδρομικά τη συνάρτηση quicksort για τον αριστερό υποπίνακα

- Με την εντολή quicksort(i, R, x);καλούμε αναδρομικά τη συνάρτηση quicksort για το δεξιό υποπίνακα

- Ο αλγόριθμος quicksort έχει χρόνο χειρότερης περίπτωσης επίσης $O(n^2)$ και μέσης περίπτωσης O(nlogn)

11.1.11 Αριθμητικές Πράξεις Μονοδιάστατων Πινάκων

11.1.11.1 Άθροισμα Πινάκων

Για το άθροισμα δύο πινάκων μεταβιβάζουμε στη συνάρτηση athroisma τους πίνακας x και y και υπολογίζουμε το άθροισμα τους στον πίνακα z

Η λειτουργία αυτή υλοποιείται με το ακόλουθο τμήμα κώδικα:

```
void athroisma(int x[n],int y[n],int z[n])
{
      int i;

      for (i=0;i<n;i++)
            z[i]=x[i]+y[i];
}
```

11.1.11.2 Πολλαπλασιασμός Πινάκων

Για τον πολλαπλασιασμό δύο πινάκων μεταβιβάζουμε στη συνάρτηση ginomeno τους πίνακας x και y και υπολογίζουμε το γινόμενο τους στον πίνακα z

Η λειτουργία αυτή υλοποιείται με το ακόλουθο τμήμα κώδικα:

```
void ginomeno(int x[n], int y[n], int z[n])
{
      int i;

      for (i=0;i<n;i++)
            z[i]=x[i]*y[i];
}
```

11.2Δισδιάστατοι πίνακες

Τα στοιχεία του πίνακα είναι και πάλι του ιδίου τύπου και βρίσκονται επίσης σε διαδοχικές θέσεις μνήμης. Λέγεται δισδιάστατος ή ορθογώνιος πίνακας γιατί όλα τα στοιχεία του είναι τοποθετημένα σε γραμμές και στήλες όπως φαίνεται και στο ακόλουθο σχήμα:

	0	**1**	**2**	**3**	**4**
0	12	-7	45	7	4
1	14	8	10	6	60
2	-9	5	20	3	2

Με τα έντονα γράμματα φαίνονται πάλι οι θέσεις του πίνακα. Παρατηρούμε ότι οι δείκτες και των γραμμών και των στηλών αριθμούνται από το 0.

11.2.1Δήλωση δισδιάστατου πίνακα

Η δήλωση ενός μονοδιάστατου πίνακα γίνεται με την ακόλουθη εντολή:

```
τύπος όνομα_πίνακα [πλήθος γραμμών][πλήθος στηλών];
```

Με τη δήλωση αφιερώνεται χώρος μνήμης σε διαδοχικές θέσεις κατάλληλος για να δεχθεί δεδομένα του περιγραφόμενου τύπου και σε πλήθος όσο το γινόμενο των διαφορετικών πληθών των δεικτών. Για παράδειγμα γράφοντας σε ένα πρόγραμμα τη δήλωση ***int*** x[3][5]; ορίζουμε ένα πίνακα με όνομα x που αποτελείται από 3 γραμμές και 5 στήλες και συνολικά από 15 στοιχεία τύπου ***int***:

	0	1	2	3	4
0	x[0,0]	x[0,1]	x[0,2]	x[0,3]	x[0,4]
1	x[1,0]	x[1,1]	x[1,2]	x[1,3]	x[1,4]
2	x[2,0]	x[2,1]	x[2,2]	x[2,3]	x[2,4]

Γενικά το στοιχείο ενός δισδιάστατου πίνακα στη γραμμή i και στη στήλη j συμβολίζεται σαν Πίνακας[i][j].[3]

11.2.2 Αρχικές τιμές δισδιάστατου πίνακα

Μαζί με την δήλωση ενός δισδιάστατου πίνακα μπορούμε να δώσουμε και αρχικές τιμές στα στοιχεία του πίνακα με εντολή της μορφής:

int x[2][5]={100, 90, 80, 70, 60, 50, 40, 30, 20, 10};

Τότε στον πίνακα x οι ακέραιοι μέσα στις αγκύλες τοποθετούνται κατά γραμμές επομένως θα έχουμε τα στοιχεία:

	0	1	2	3	4
0	100	90	80	70	60
1	50	40	30	20	10

Δηλαδή το x[0][0] θα είναι ο ακέραιος 100, το x[1][2] θα είναι ακέραιος 30 κ.λ.π. Η προηγούμενη δήλωση θα μπορούσε να γίνει και ως εξής:

int x[2][5]= {{100,90,80,70,60}, {50,40,30,20,10}};

[3] Οι γραμμές και οι στήλες ενός δισδιάστατου πίνακα αριθμούνται πάντα από το μηδέν

11.2.3 Εκχώρηση τιμών στα στοιχεία δισδιάστατου πίνακα

Από τη στιγμή που θα δηλώσουμε ένα πίνακα δεν υπάρχει τρόπος να δώσουμε με μια εντολή νέες τιμές σε όλα τα στοιχεία του (όπως συμβαίνει κατά την δήλωσή του με αρχικές τιμές). Δηλαδή μετά από τη δήλωση: ***int*** x[2][5]; είναι λάθος να γράψουμε:

x={{100,90,80,70,60,50,40,30,20,10}};

Μπορούμε όμως να δίνουμε τιμές σε μεμονωμένα στοιχεία του πίνακα γράφοντας:

x[1][3]=100;
x[0][2]=500;

11.2.4 Μέγεθος δισδιάστατου πίνακα

Το μέγεθος ενός δισδιάστατου πίνακα δηλαδή το πλήθος γραμμών και στηλών του δηλώνεται συνήθως με τη βοήθεια 2 καθολικών σταθερών όπως φαίνεται στο ακόλουθο παράδειγμα:

```
const n=3; Πλήιος γραμμών
const m=4; //Πλήθος στηλών
```

11.2.5 Γενικοί Αλγόριθμοι Δισδιάστατων Πινάκων

11.2.5.1 Διάβασμα Δισδιάστατου Πίνακα

Για να διαβάσουμε (γεμίσουμε) ένα δισδιάστατο πίνακα κατά γραμμές με τιμές που εισάγουμε από το πληκτρολόγιο γράφουμε τις

ακόλουθες εντολές: (έστω ότι έχει προηγηθεί η δήλωση ***float*** x[3][5])

```
for (i=0;i<3;i++)
    for (j=0;j<5;j++)
    {
        cout<<"Δώσε τιμή στο στοιχείο στη θέση <<i+1<<”,”<<j+1;
        cin<<x[i][j];
    }
```

Για να διαβάσουμε (γεμίσουμε) ένα δισδιάστατο πίνακα **κατά στήλες** με τιμές που εισάγουμε από το πληκτρολόγιο γράφουμε τις ακόλουθες εντολές: (έστω ότι και πάλι έχει προηγηθεί η δήλωση ***float*** x[3][5])

```
for (j=0;j<3;j++)
    for (i=0;i<5;i++)
    {
        cout<<"Δώσε τιμή στο στοιχείο στη θέση <<j+1<<”,”<<i+1;
        cin<<x[i][j];
    }
```

11.2.5.2 Εκτύπωση δισδιάστατου πίνακα

Για να εκτυπώσουμε (εμφανίσουμε) τα στοιχεία ενός δισδιάστατου πίνακα στην οθόνη γράφουμε τις ακόλουθες εντολές:

(έστω ότι έχει προηγηθεί η δήλωση ***float*** x[3][5] και βέβαια ο πίνακας x έχει πάρει τιμές)

```
for (i=0;i<3;i++)
{
    for (j=0;j<5;j++)
          cout<<x[i][j];
    cout<<endl;
}
```

11.2.5.3 Υπολογισμός Μέσου Όρου Δισδιάστατου Πίνακα

Για να υπολογίσουμε το μέσο όρο των στοιχείων ενός δισδιάστατου πίνακα αθροίζουμε πρώτα όλα τα στοιχεία του πίνακα και στη συνέχεια διαιρούμε το άθροισμα με το πλήθος όλων των στοιχείων του πίνακα που είναι το γινόμενο των γραμμών και των στηλών. Αυτή λειτουργία υλοποιείται με το ακόλουθο τμήμα κώδικα:

```
float mesos_oros(int x[n][m])
{
      int i,j,sum=0;

      for (i=0; i<n;i++)
           for (j=0;j<m;j++)
                sum+=x[i][j];

      return (float)sum/(n*m);
}
```

11.2.5.4 Υπολογισμός μέσου όρου κάθε γραμμής δισδιάστατου πίνακα

Για να υπολογίσουμε το μέσο όρο των στοιχείων κάθε γραμμής ενός δισδιάστατου πίνακα αθροίζουμε πρώτα όλα τα στοιχεία της κάθε γραμμής και στη συνέχεια διαιρούμε το άθροισμα με το πλήθος όλων των στοιχείων της γραμμής. Αυτή λειτουργία υλοποιείται με το ακόλουθο τμήμα κώδικα:

```
void mo_grammon(int x[n][m])
{
    int i,j,sum=0;
    float mo=0;

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

        for (j=0;j<m;j++)
            sum+=x[i][j];

        mo=(float)sum/m;

        cout<<"Μέσος όρος "<<i+1<<" γραμμής ="
<<mo<<endl;
    }
}
```

Παρατήρηση: Ο μέσος όρος κάθε γραμμής του πίνακα υπολογίζεται στο τέλος της εσωτερικής επανάληψης όπου έχουμε υπολογίσει το άθροισμα των στοιχείων της γραμμής. Επίσης ο αθροιστής αυτός πρέπει να μηδενίζεται στην κάθε γραμμή του πίνακα (και όχι μόνο μια φορά όπως πριν).

11.2.5.5 Υπολογισμός μέσου όρου κάθε στήλης δισδιάστατου πίνακα

Για να υπολογίσουμε το μέσο όρο των στοιχείων κάθε στήλης ενός δισδιάστατου πίνακα αθροίζουμε πρώτα όλα τα στοιχεία της κάθε στήλης και στη συνέχεια διαιρούμε το άθροισμα με το πλήθος όλων των στοιχείων της στήλης. Αυτή λειτουργία υλοποιείται με το ακόλουθο τμήμα κώδικα:

```
void mo_stilon(int x[n][m])
{
    int i,j,sum=0;
    float mo=0;

    for (j=0;j<m;j++)
    {
        sum=0;

        for (i=0;i<n;i++)
            sum+=x[i][j];

        mo=(float)sum/m;
```

```
        cout<<"Μέσος όρος "<<j+1<<" στήλης =" <<mo<<endl;
    }
}
```

11.2.5.6 Υπολογισμός πλήθους άρτιων-περιττών στοιχείων δισδιάστατου πίνακα

Για να υπολογίσουμε το πλήθος άρτιων και περιττών στοιχείων όλου του πίνακα χρησιμοποιούμε 2 μετρητές, τον μετρητή a ο οποίος υπολογίζει το πλήθος των άρτιων στοιχείων και το μετρητή p ο οποίος υπολογίζει αντίστοιχα το πλήθος των περιττών στοιχείων του πίνακα. Η λειτουργία αυτή υλοποιείται με το ακόλουθο τμήμα κώδικα:

```
void artioi_perittoi(int x[n][m])
{
   int i,j,a=0,p=0;

   for (i=0;i<n;i++)
        for (j=0;j<m;j++)
             if (x[i][j]%2==0)
                  a++;
             else
                  p++;
```

```
    cout<<"Άρτιοι Πίνακα ="<<a<<endl;
    cout<<"Περιττοί Πίνακα ="<<p<<endl;
}
```

Παρατήρηση: Οι 2 μετρητές a και p πρέπει να αρχικοποιηθούν με μηδέν πριν ξεκινήσουν οι 2 επαναλήψεις.

11.2.5.7 Υπολογισμός πλήθους άρτιων-περιττών στοιχείων κάθε γραμμής δισδιάστατου πίνακα

Για να υπολογίσουμε το πλήθος άρτιων και περιττών στοιχείων κάθε γραμμής του πίνακα χρησιμοποιούμε 2 μετρητές, τον μετρητή a ο οποίος υπολογίζει το πλήθος των άρτιων στοιχείων και το μετρητή p ο οποίος υπολογίζει αντίστοιχα το πλήθος των περιττών στοιχείων κάθε γραμμής του πίνακα. Η λειτουργία αυτή υλοποιείται με το ακόλουθο τμήμα κώδικα:

```
void artioi_perittoi_grammon(int x[n][m])
{
    int i,j,a,p;

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

        for (j=0;j<m;j++)
            if (x[i][j]%2==0)
                a++;
```

```
                    else
                        p++;


            cout<<"Άρτιοι   "<<i+1<<" γραμμής = "<<a<<endl;
            cout<<"Περιττοί "<<i+1<<" γραμμής = "<<p<<endl;
        }
}
```

11.2.5.8 Υπολογισμός πλήθους άρτιων-περιττών στοιχείων κάθε στήλης δισδιάστατου πίνακα

Για να υπολογίσουμε το πλήθος άρτιων και περιττών στοιχείων κάθε στήλης του πίνακα χρησιμοποιούμε ομοίως 2 μετρητές, τον μετρητή a ο οποίος υπολογίζει το πλήθος των άρτιων στοιχείων και το μετρητή p ο οποίος υπολογίζει αντίστοιχα το πλήθος των περιττών στοιχείων κάθε στήλης του πίνακα. Η λειτουργία αυτή υλοποιείται με το ακόλουθο τμήμα κώδικα:

```
void artioi_perittoi_stilon(int x[n][m])
{
        int i,j,a,p;


        for (j=0;j<m;j++)
        {
                a=0;
                p=0;
```

```
            for (i=0;i<n;i++)
                if (x[i][j]%2==0)
                    a++;
                else
                    p++;

            cout<<"Άρτιοι  "<<j+1<<" στήλης = "<<a<<endl;
            cout<<"Περιττοί "<<j+1<<" στήλης = "<<p<<endl;
        }
}
```

11.2.5.9 Εύρεση μέγιστου-ελάχιστου στοιχείου πίνακα και των θέσεων τους στο δισδιάστατο πίνακα

Για να βρούμε στο δισδιάστατο πίνακα το μέγιστο και το ελάχιστο στοιχείο του και τις θέσεις τους στον πίνακα, υποθέτουμε αρχικά ότι ο μέγιστο και το ελάχιστο στοιχείο του πίνακα είναι το πρώτο (αποθηκεύουμε το μέγιστο και το ελάχιστο στοιχείο του πίνακα στις μεταβλητές max και min αντίστοιχα). Επίσης στις μεταβλητές maxpi και maxpj καταχωρούμε τη γραμμή και τη στήλη που βρίσκεται το μέγιστο στοιχείο του πίνακα αντίστοιχα, στις μεταβλητές minpi και minpj καταχωρούμε τη γραμμή και τη στήλη που βρίσκεται το ελάχιστο στοιχείο του πίνακα αντίστοιχα. Επίσης ο αλγόριθμος μας εντοπίζει όλα τα στοιχεία που είναι ίσα με το μέγιστο και το ελάχιστο στοιχείο αντίστοιχα και τυπώνει τις θέσεις στις οποίες αυτά εμφανίζονται.

Η λειτουργία αυτή υλοποιείται με το ακόλουθο τμήμα κώδικα:

```
void max_min(int x[n][m])
{
	int i,j,max,min,maxpi,maxpj,minpi,minpj;

	max=min=x[0][0];
	maxpi=maxpj=minpi=minpj=0;

	for (i=0;i<n;i++)
		for (j=0;j<m;j++)
		{
			if (x[i][j]>max)
			{
				max=x[i][j];
				maxpi=i;
				maxpj=j;
			}

			if (x[i][j]<min)
			{
				min=x[i][j];
				minpi=i;
				minpj=j;
			}
		}
	cout<<"Μέγιστο ="<<max<<" στις θέσεις "<<endl;

	for (i=0;i<n;i++)
		for (j=0;j<m;j++)
			if (x[i][j]==max)
```

```
                    cout<<i+1<<","<<j+1<<endl;

    cout<<"Ελάχιστο ="<<min<<" στις θέσεις "<<endl;

    for (i=0;i<n;i++)
        for (j=0;j<m;j++)
            if (x[i][j]==min)
                cout<<i+1<<","<<j+1<<endl;
```

11.2.5.10 Εύρεση μέγιστου-ελάχιστου στοιχείου κάθε γραμμής του πίνακα και των θέσεων τους στη γραμμή

Για να βρούμε το μέγιστο και το ελάχιστο στοιχείο σε κάθε γραμμή του δισδιάστατου πίνακα καθώς και τις θέσεις τους στη γραμμή, υποθέτουμε αρχικά ότι ο μέγιστο και το ελάχιστο στοιχείο κάθε γραμμής είναι το αρχικό στοιχείο της γραμμής. Στη συνέχεια συγκρίνουμε κάθε επόμενο στοιχείο της γραμμής με το μέγιστο και το ελάχιστο στοιχείο αντίστοιχα και στο τέλος εκτυπώνουμε το μέγιστο και το ελάχιστο σε κάθε γραμμή και τις θέσεις τους.

Η λειτουργία αυτή υλοποιείται με το ακόλουθο τμήμα κώδικα:

```
void max_min_grammis(int x[n][m])
{
    int i, max, min, j, maxpj, minpj;

    for (i=0;i<n;i++)
    {
        max=min=x[i][0];
```

```
            maxpj=minpj=0;

            for (j=0;j<m; j++)
            {
                if (x[i][j]>max)
                {
                    max=x[i][j];
                    maxpj=j;
                }

                if (x[i][j]<min)
                {
                    min=x[i][j];
                    minpj=j;
                }
            }

            cout<<"Μέγιστο "<<i+1<<" γραμμής= "<<max<<" στη
θέση "<<maxpj+1<<endl;

            cout<<"Ελάχιστο "<<i+1<<" γραμμής = "<<min<<" στη
θέση "<<minpj+1<<endl;
        }
}
```

Παρατήρηση

Τα αποτελέσματα για την κάθε γραμμή τυπώνονται στο τέλος της εσωτερικής επανάληψης, δηλαδή όταν έχουμε ολοκληρώσει τον έλεγχο της γραμμής.

11.2.5.11 Εύρεση μέγιστου-ελάχιστου στοιχείου κάθε στήλης του πίνακα και των θέσεων τους στη στήλη

Για να βρούμε το μέγιστο και το ελάχιστο στοιχείο σε κάθε στήλη του δισδιάστατου πίνακα καθώς και τις θέσεις τους στη στήλη, υποθέτουμε αρχικά ότι ο μέγιστο και το ελάχιστο στοιχείο κάθε στήλης είναι το αρχικό στοιχείο της στήλης. Στη συνέχεια συγκρίνουμε κάθε επόμενο στοιχείο της στήλης με το μέγιστο και το ελάχιστο στοιχείο αντίστοιχα και στο τέλος εκτυπώνουμε το μέγιστο και το ελάχιστο σε κάθε στήλη και τις θέσεις τους.

Η λειτουργία αυτή υλοποιείται με το ακόλουθο τμήμα κώδικα:

```
void max_min_stilis(int x[n][m])
{
	int i, max, min, j, maxpi, minpi;

	for (j=0; j<m; j++)
	{
		max=min=x[0][j];
		maxpi=minpi=0;

		for (i=0;i<n; i++)
		{
			if (x[i][j]>max)
			{
				max=x[i][j];
				maxpi=i;
			}
```

```
                if (x[i][j]<min)
                {
                    min=x[i][j];
                    minpi=i;
                }
            }

            cout<<"Μέγιστο "<<i+1<<" στήλης= "<<max<<" στη θέση "<<maxpi+1<<endl;

            cout<<"Ελάχιστο "<<i+1<<" στήλης = "<<min<<" στη θέση "<<minpj+1<<endl;
        }
}
```

Παρατήρηση

Τα αποτελέσματα για την κάθε στήλη τυπώνονται στο τέλος της εσωτερικής επανάληψης, δηλαδή όταν έχουμε ολοκληρώσει τον έλεγχο της στήλης.

11.2.5.12 Άθροισμα διαγωνίων

Για να υπολογίσουμε το άθροισμα της κύριας και της δευτερεύουσας διαγωνίου ενός πίνακα χρησιμοποιούμε δύο αθροιστές (ένα για τα στοιχεία κάθε διαγωνίου). Βέβαια για να έχει νόημα αυτή η λειτουργία πρέπει ο πίνακας να είναι τετραγωνικός δηλαδή το άθροισμα των γραμμών και στων στηλών του πίνακα να είναι ίσα.

Η λειτουργία αυτή υλοποιείται με το ακόλουθο τμήμα κώδικα:

```
void athroisma_diagonion(int x[n][m])
{
	int i,j,sum1=0,sum2=0;

	for (i=0;i<n; i++)
		for (j=0;j<m; j++)
		{
			if (i==j)
				sum1+=x[i][j];

			if (i+j==n-1)
				sum2+=x[i][j];
		}

	cout<<"Άθροισμα Κύριας Διαγωνίου = "<<sum1<<endl;
	cout<<"Άθροισμα Δευτερεύουσας Διαγωνίου "<<sum2<<endl;
}
```

11.2.6 Αλγόριθμοι Αναζήτησης σε Δισδιάστατους Πίνακες

11.2.6.1 Σειριακή αναζήτηση στοιχείου σε δισδιάστατο πίνακα

Για να εκτελέσουμε τη σειριακή αναζήτηση στοιχείου σε δισδιάστατο πίνακα και να μετρήσουμε το πλήθος εμφανίσεων του στοιχείου καθώς και τις θέσεις στις οποίες αυτό εμφανίζεται, εκτελούμε με το ακόλουθο τμήμα κώδικα:

```
void search(int x[n][m],int y)
{
      int i,j,t=0;

      for (i=0;i<n;i++)
            for (j=0;j<m;j++)
                  if (y==x[i][j])
                  {
                        cout<<"Θέση "<<i+1<<","<<j+1<<endl;
                        t++;
                  }

      if (t==0)
          cout<<"Δεν βρέθηκε στον πίνακα"<<endl;
      else
          cout<< "Βρέθηκε "<<t<<" φορές "<<endl;
}
```

Παρατήρηση

Το y είναι το στοιχείο αναζήτησης το οποίο έχουμε διαβάσει στο κύριο πρόγραμμα ή σε άλλη συνάρτηση και το μεταβιβάζουμε στη συνάρτηση search_line ως όρισμα

11.2.6.2 Σειριακή αναζήτηση στοιχείου σε κάθε γραμμή του δισδιάστατου πίνακα

Για να εκτελέσουμε τη σειριακή αναζήτηση στοιχείου σε δισδιάστατο πίνακα και να μετρήσουμε το πλήθος εμφανίσεων του στοιχείου σε κάθε γραμμή του πίνακα ξεχωριστά καθώς και τις θέσεις στις οποίες αυτό εμφανίζεται στη γραμμή, εκτελούμε με το ακόλουθο τμήμα κώδικα:

```
void search_line(int x[n][m],int y)
{
	int i,j,t;

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

		for (j=0;j<m;j++)
		{
			if (y==x[i][j])
			{
				cout<<"Θέση "<<i+1<<","<<j+1<<endl;
				t++;
			}

		cout<< "Βρέθηκε "<<t<<" φορές στη γραμμή "<<i+1<<endl;
```

```
            }
        }
}
```

Παρατήρηση

Ο μετρητής t μηδενίζεται σε κάθε γραμμή ώστε να μετρά ξεχωριστά το πλήθος εμφανίσεων σε κάθε γραμμή

11.2.6.3 Σειριακή αναζήτηση στοιχείου σε κάθε στήλη του δισδιάστατου πίνακα

Για να εκτελέσουμε τη σειριακή αναζήτηση στοιχείου σε δισδιάστατο πίνακα και να μετρήσουμε το πλήθος εμφανίσεων του στοιχείου σε κάθε στήλη του πίνακα ξεχωριστά καθώς και τις θέσεις στις οποίες αυτό εμφανίζεται στη στήλη, εκτελούμε με το ακόλουθο τμήμα κώδικα:

```
void search_column(int x[n][m],int y)
{
      int i,j,t;

      for (j=0; j<m; j++)
      {
            t=0;

            for (i=0;i<n;i++)
            {
                  if (y==x[i][j])
```

```
                {
                    cout<<"Θέση "<<i+1<<","<<j+1<<endl;
                    t++;
                }

            cout<< "Βρέθηκε "<<t<<" φορές στη στήλη "<<j+1<<endl;
            }
        }
}
```

Παρατήρηση

Ο μετρητής t μηδενίζεται σε κάθε στήλη ώστε να μετρά ξεχωριστά το πλήθος εμφανίσεων σε κάθε στήλη

11.2.7 Αριθμητικές Πράξεις Δισδιάστατων Πινάκων

11.2.7.1 Άθροισμα Πινάκων

Για το άθροισμα δύο πινάκων μεταβιβάζουμε στη συνάρτηση athroisma τους πίνακας x και y και υπολογίζουμε το άθροισμα τους στον πίνακα z

Η λειτουργία αυτή υλοποιείται με το ακόλουθο τμήμα κώδικα:

```
void athroisma (int x[n][n],int y[n][n],int z[n][n])
{
      int i, j;

      for (i=0;i<n;i++)
            for (j=0;j<n; j++)
                  z[i][j]=x[i][j]+y[i][j];
}
```

11.2.7.2 Πολλαπλασιασμός Πινάκων

Για τον πολλαπλασιασμό δύο πινάκων μεταβιβάζουμε στη συνάρτηση ginomeno τους πίνακας x και y και υπολογίζουμε το γινόμενο τους στον πίνακα z

Η λειτουργία αυτή υλοποιείται με το ακόλουθο τμήμα κώδικα:

```
void ginomeno(int x[n][n],int y[n][n],int z[n][n])
{
      int i, j, k;

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

```
            {
                z[i][j]=0;

                for (k=0;k<n; k++)
                    z[i][j]+=x[i][k]*y[k][j];
            }
}
```

11.2.8 Ειδικές Μορφές Δισδιάστατων Πινάκων

11.2.8.1 Μοναδιαίος Πίνακας

Για την κατασκευή του μοναδιαίου πίνακα ο οποίος περιέχει το 1 σε όλες τις θέσεις της κύριας διαγωνίου και το 0 σε όλες τις υπόλοιπες θέσεις του γράφουμε το ακόλουθο τμήμα κώδικα:

```
void monadiaios(int x[n][n])
{
      int i, j;

      for (i=0;i<n;i++)
            for(j=0;j<n; j++)
                  if (i==j)
                        x[i][j]=1;
                  else
                        x[i][j]=0;
}
```

11.2.8.2 Ανάστροφος Πίνακας

Για την κατασκευή του ανάστροφου πίνακα ενός πίνακα y ο οποίος καταχωρείται σε ένα πίνακα z, γράφουμε το ακόλουθο τμήμα κώδικα:

```
void anastrofos (int y[n][n],int z[n][n])
{
      int i, j;
```

```
        for (i=0;i<n;i++)
                for(j=0;j<n; j++)
                        z[i][j]=y[j][i];
}
```

12 Β Μέρος. Εισαγωγή στη Java

Η γλώσσα προγραμματισμού Java σχεδιάστηκε για να δώσει λύσεις σε μία σειρά πρακτικών προβλημάτων που εμφανίζονταν σε όλες τις ήδη υπάρχουσες γλώσσες. Η γλώσσα στην οποία στηρίχτηκε αρκετά η προσπάθεια δημιουργίας αυτής της νέας γλώσσας ήταν η C++, μία αντικειμενοστραφής (object oriented) γλώσσα η οποία όμως πέρα από τα αρχικά προβλήματα, κυρίως σε πρακτικά ζητήματα, με τους υπάρχοντες μεταφραστές, παρουσίασε ουσιαστικά προβλήματα στην πορεία. Η λύση που προτάθηκε από τους μηχανικούς της Sun ήταν η δημιουργία μιας εντελώς νέας γλώσσας που θα είναι αντικειμενοστραφής και θα βασίζεται στη C++.

Ο επίσημος ορισμός της Java, όπως αυτός δίνεται από τη Sun, έχει ως εξής:

Java: Μία απλή, αντικειμενοστραφής, έχουσα δικτυακή γνώση, μεταφράσιμη, εύρωστη, ασφαλής, ανεξάρτητη αρχιτεκτονικής, μεταφέρσιμη, υψηλής απόδοσης, πολυνηματική, δυναμική γλώσσα.

Όταν ζητάμε από τη Java να μεταγλωττίσει το πρόγραμμα μας αυτή δεν παράγει ένα πρόγραμμα όπως αυτά που έχουμε συνηθίσει να βλέπουμε. Αντί να δημιουργήσει ένα εκτελέσιμο αρχείο γεμάτο από εντολές μηχανής, ο μεταγλωττιστής της Java παράγει στην έξοδο αυτό που είναι γνωστό ως κώδικας byte Java (Java byte codes). Ο κώδικας byte Java είναι εντολές γραμμένες για κάποια εικονική μηχανή Java (virtual machine) που δεν υπάρχει πραγματικά. Για να εκτελέσουμε το πρόγραμμά μας, πρέπει να έχουμε ένα ερμηνευτή κώδικα byte java, πράγμα που σημαίνει ότι

στην πραγματικότητα εξομοιώνουμε την εικονική μηχανή Java (JVM) σε όποιον υπολογιστή και όποιο λειτουργικό σύστημα χρησιμοποιούμε. Κανονικά δεν θα το αντιλαμβανόμαστε αυτό, αφού η Java το κάνει αυτόματα όταν εκτελούμε την εφαρμογή μας μέσα στο Αλληλεπιδραστικό Περιβάλλον Ανάπτυξης (Interactive Development Environment – IDE) της Java.

13 Χαρακτηριστικά Java

Όταν εκτελούμε μια μικροεφαρμογή στον Ιστό (Web), είναι ευθύνη του φυλλομετρητή (browser) να υλοποιήσει τον κώδικα byte της java. (Η έκδοση 3.0 και οι μεταγενέστερες του Microsoft Internet Explorer και η έκδοση 2.0 και οι επόμενες του Netscape Navigator), μεταξύ άλλων υλοποιούν τον ερμηνευτή του κώδικα byte Java στο PC). Η προσέγγιση μέσω της εικονικής μηχανής Java εξυπηρετεί 3 σκοπούς:

- **Ανεξαρτησία από την μηχανή.** Υπάρχει μία μόνο Εικονική Μηχανή Java. (Στην πραγματικότητα, δεν υπάρχει καμία – αυτό είναι που την κάνει εικονική). Υπάρχει ένας διαφορετικός εξομοιωτής της Εικονικής Μηχανής Java για κάθε διαφορετικό τύπο PC. Αυτό είναι που δίνει στην Java την ανεξαρτησία της από τη μηχανή.

- **Ασφάλεια.** Η πείρα έχει δείξει πόσο δύσκολο είναι να επιτευχθεί ένα λογικό επίπεδο ασφαλείας στο λογισμικό. Από τη στιγμή που τα προγράμματα εκτελούνται στην Κεντρική Μονάδα Επεξεργασίας (ΚΜΕ – CPU) σε εγγενή κατάσταση (native mode), οποιαδήποτε ατέλεια στο λειτουργικό σύστημα ή στο φυλλομετρητή μπορεί να αξιοποιηθεί από τους χάκερ που ψάχνουν για μια είσοδο στη μηχανή – πελάτη. Τα προγράμματα

της Java δεν έχουν αυτή την ευελιξία. Ένα πρόγραμμα Java εκτελείται στην εικονική μηχανή Java που δημιουργείται από το φυλλομετρητή. Καθώς το πρόγραμμα εκτελείται, ο φυλλομετρητής είναι πάντοτε παρών, παρακολουθώντας κατά κάποιον τρόπο το πρόγραμμα ώστε να εξασφαλίσει ότι αυτό δεν θα κάνει κάτι που δεν πρέπει.

Μικρότερο μέγεθος μικροεφαρμογών. Τα προγράμματα της Εικονικής μηχανής Java είναι πολύ μικρότερα από τα συμβατικά προγράμματα. Αυτό συμβαίνει γιατί οι ενσωματωμένες συναρτήσεις βιβλιοθήκης της Java, τις οποίες καλεί το πρόγραμμά μας, βρίσκονται μέσα στο φυλλομετρητή αντί να συνδέονται (link) στο πρόγραμμα. Αυτό μειώνει σημαντικά το μέγεθος των μικροεφαρμογών Java Που πρέπει να "κατέβουν" (download), οπότε μειώνει και το χρόνο μεταφοράς. Η Java μειώνει ακόμη περισσότερο το χρόνο μεταφοράς στο σύστημά μας, κατεβάζοντας μόνον ότι χρειάζεται. (Ο κώδικας byte που παράγεται για μια δεδομένη κλάση δεν μεταφέρεται μέχρι να τον χρειαστεί το πρόγραμμα).

Έγκαιρη Μεταγλώττιση. Υπάρχει ένα μειονέκτημα στην ερμηνεία του κώδικα byte της Java από το φυλλομετρητή – μειωμένη απόδοση. Τα προγράμματα κώδικα byte της Java δεν εκτελούνται τόσο γρήγορα όσο τα κανονικά προγράμματα, εγγενούς κώδικα. Για να αντιμετωπίσει αυτό το πρόβλημα, το Microsoft Internet Explorer υποστηρίζει μια ευκολία που ονομάζεται Έγκαιρη Μεταγλώττιση (compilation Just-In-Time ή JIT). Όταν το Internet Explorer δέχεται για πρώτη φορά μια μικροεφαρμογή Java, αν η Έγκαιρη Μεταγλώττιση είναι ενεργοποιημένη, μετατρέπει τον κώδικα byte της Java σε εγγενές

πρόγραμμα, το οποίο και αποθηκεύει στο δίσκο. Στην πράξη, η Έγκαιρη Μεταγλώττιση αντιμετωπίζει τον κώδικα Byte της Java σαν γλώσσα προέλευσης και τον μεταγλωττίζει σε γλώσσα της τοπικής μηχανής. Η Έγκαιρη Μεταγλώττιση διατηρεί πολλά από τα πλεονεκτήματα του κώδικα Byte της Java. Από τη στιγμή που δεν αλλάζει των κώδικα Java που παράγεται από τη Java, το πρόγραμμα που προκύπτει εξακολουθεί να είναι ασφαλές και ανεξάρτητο από τη μηχανή και οι χρόνοι μεταφοράς παραμένουν ελαχιστοποιημένη. Το Έγκαιρα Μεταγλωττισμένο πρόγραμμα εκτελείται πολλές φορές γρηγορότερα, ακόμη και από τους καλύτερους εξομοιωτές κώδικα byte Java.

Η Java προκάλεσε ίσως το μεγαλύτερο ενδιαφέρον σε σύγκριση με οποιαδήποτε άλλη εξέλιξη στον κόσμο του Internet. Όλοι μιλούν γι' αυτήν. Όλοι έχουν ενθουσιαστεί με τη Java για τις δυνατότητες που προσφέρει. Είναι η πρώτη που κατάφερε να συμπεριλάβει ήχο και κίνηση σε μια ιστοσελίδα. Η Java επιπλέον επιτρέπει στους χρήστες να αλληλεπιδρούν (interact) με την ιστοσελίδα. Εκτός από το να διαβάζει απλά και ίσως να συμπληρώνει μία φόρμα, ο χρήστης μπορεί τώρα να παίξει παιχνίδια, να συνομιλήσει, να λαμβάνει συνεχώς τις πιο πρόσφατες πληροφορίες και πολλά άλλα.

Ακολουθούν μερικές από τις πολλές δυνατότητες της Java:

- Ήχος ο οποίος εκτελείται όποτε ο χρήστης φορτώνει μία σελίδα
- Μουσική που παίζει στο φόντο μιας σελίδας
- Δημιουργία κινουμένων σχεδίων
- Βίντεο

Ø Παιχνίδια με πολυμέσα

Η Java δεν είναι απλά μια γλώσσα προγραμματισμού του δικτύου με ειδικά χαρακτηριστικά. Παρόλο που η HotJava ήταν η πρώτη γλώσσα που συμπεριέλαβε ήχο και κίνηση, ο Microsoft Internet Explorer 2.0 και ο Netscape Navigator 2.0 υποστηρίζουν αυτά τα χαρακτηριστικά με πολλούς και διαφορετικούς τρόπους. Τι κάνει τη Java να ξεχωρίζει; Η Java είναι μια γλώσσα προγραμματισμού για ποικίλες εφαρμογές. Δεν προσφέρει απλά τη δυνατότητα να προσθέσει ο χρήστης νέο περιεχόμενο στις σελίδες του (όπως συμβαίνει στο Netscape και στον Internet Explorer) αλλά επιτρέπει να προσθέσουμε και τον κώδικα που είναι απαραίτητος. Δεν χρειάζεται πλέον να περιμένουμε για να κυκλοφορήσει ο browser που θα υποστηρίξει τον συγκεκριμένο τύπο εικόνας ή το ειδικό πρωτόκολλο παιχνιδιού (special game protocol). Με τη Java εμείς στέλνετε στους browsers το περιεχόμενο που χρειάζεται και το πρόγραμμα για να δούμε αυτό το περιεχόμενο την ίδια στιγμή.

Η Java δεν είναι γλώσσα μόνο για τα web sites. Η Java είναι μια γλώσσα προγραμματισμού που μας επιτρέπει να κάνουμε ό,τι και οι παραδοσιακές γλώσσες, όπως η Fortran και η C++. Είναι σαφώς πιο καθαρή και πιο εύκολη όμως στη χρήση από αυτές. Σαν γλώσσα η Java είναι:

- Απλή (Simple)
- Αντικειμενοστραφής, δηλαδή τα πάντα στη Java είναι είτε κλάση, είτε μέθοδος ή αντικείμενο
- Ανεξάρτητη από το σύστημα, δηλαδή τα προγράμματα σε Java μπορούν να διαβαστούν και να τρέξουν από μεταγλωττιστές σε

διάφορες πλατφόρμες όπως Windows 95, Windows NT και Solaris 2.3

- Ασφαλής
- Πολυνηματική, δηλαδή ένα απλό πρόγραμμα σε Java μπορεί να κάνει πολλά, διαφορετικά προγράμματα ανεξάρτητα και αλληλεπιδρώντα.

14 Τρόπος γραφής προγραμμάτων σε Java

Για να γράψουμε ένα πρόγραμμα σε Java ακολουθούμε κατά σειρά τα ακόλουθα βήματα:

1. Γράφουμε πρώτα τις εντολές του προγράμματος μέσα στο διορθωτή κειμένου (editor). Ο διορθωτής κειμένου είναι ένας απλός κειμενογράφος με περιορισμένες δυνατότητες μορφοποίησης, εκτύπωσης κ.λ.π. Το πρόγραμμα που γράφουμε ονομάζεται πηγαίο (source program)
2. Στη συνέχεια στέλνουμε το πηγαίο πρόγραμμα στο μεταγλωττιστή (compiler) ο οποίος αναλαμβάνει να μεταφράσει το πηγαίο πρόγραμμα σε γλώσσα μηχανής. Αυτό είναι εφικτό μόνο όταν ο μεταγλωττιστής δεν εντοπίσει στο πρόγραμμα μας συντακτικά λάθη. Αν υπάρχουν συντακτικά λάθη ο μεταγλωττιστής μας ενημερώνει για το ποια λάθη είναι αυτά και σε ποια γραμμή έγιναν. Συντακτικά είναι τα λάθη που οφείλονται στον τρόπο γραφής των προγραμμάτων π.χ. έλλειψη κάποιου ερωτηματικού, παρένθεσης κ.λ.π. Το πρόγραμμα που εξάγεται από τον μεταγλωττιστή ονομάζεται αντικείμενο (object program).

3. Στη συνέχεια συνδέουμε το αντικείμενο πρόγραμμα με τις βιβλιοθήκες της γλώσσας. Μια βιβλιοθήκη περιέχει εντολές κοινής χρήσης όπως π.χ. εντολές εισόδου-εξόδου, κοινές μαθηματικές συναρτήσεις κ.λ.π. Τη σύνδεση αυτή την κάνει ένα ειδικό πρόγραμμα που ονομάζεται συνδέτης (linker). Το πρόγραμμα που παράγεται από το συνδέτη ονομάζεται εκτελέσιμο (executable program)
4. Εκτελούμε το πρόγραμμα και βλέπουμε τις απαντήσουμε στην οθόνη

15 Δομή Προγράμματος Java

Ένα πρόγραμμα σε java έχει την ακόλουθη μορφή:

```
public class όνομα //το όνομα της κλάσης πρέπει να είναι ίδιο με το όνομα του αρχείου
{
  public static void main(String [] args)
  {
    δήλωση μεταβλητών;

    εντολή 1;
    εντολή 2;
    .....
  εντολή ν;
  } //τέλος κύριου προγράμματος
} //τέλος κλάσης
```

Παράδειγμα

```
public class HelloWorld
{
  public static void main (String [] args)
  {
    System.out.println ("Hello World!");
  }
}
```

Το Hello World είναι ίσως το πιο απλό πρόγραμμα που μπορεί κανείς να φανταστεί. Παρόλα αυτά μας ενδιαφέρει για πολλούς λόγους. Ας το εξετάσουμε γραμμή προς γραμμή.

Η αρχική δήλωση class μπορεί να θεωρείται ότι προσδιορίζει το όνομα του προγράμματος, στη συγκεκριμένη περίπτωση Hello World. Ο compiler στην πραγματικότητα παίρνει το όνομα από τη δήλωση class HelloWorld στο αρχείο πηγαίου κώδικα. Αν υπάρχουν παραπάνω από μία κλάση σε ένα αρχείο, τότε ο compiler της Java θα αποθηκεύσει το καθένα σ' ένα ξεχωριστό .class αρχείο. Το Hello World class περιλαμβάνει μία μέθοδο, τη main. Όπως και στη C, η μέθοδος main μας δείχνει από που ξεκινά να εκτελείται μία εφαρμογή. Η μέθοδος δηλώνεται δημόσια (public) δηλαδή μπορούν να την καλέσουν από παντού. Δηλώνεται στατική (static) δηλαδή όλα τα παραδείγματα της κλάσης μοιράζονται την ίδια μέθοδο. Δηλώνεται κενή (void) που σημαίνει ότι αυτή η μέθοδος δεν επιστρέφει τιμή, όπως και στη C.

Όταν καλείται η μέθοδος main τυπώνει το Hello World! στην οθόνη. Αυτό επιτυγχάνεται με μέθοδο System.out.println . Για να είμαστε πιο ακριβείς, αυτό επιτυγχάνεται καλώντας τη συνάρτηση

println() του πεδίου out που ανήκει στην κλάση System. Αλλά για την ώρα θα την θεωρούμε σαν μία μέθοδο.

Αντίθετα με την printf στη C η μέθοδος System.out.println προσθέτει μια καινούρια γραμμή στο τέλος της εξόδου. Έτσι δεν είναι ανάγκη να συμπεριλάβουμε το \n στο τέλος κάθε αλφαριθμητικού.

16 Χαρακτηριστικά Java

16.1Αλφάβητο γλώσσας

Όπως κάθε ομιλούμενη γλώσσα χρησιμοποιεί τους χαρακτήρες του αλφαβήτου για να σχηματίσει λέξεις, έτσι και κάθε γλώσσα προγραμματισμού έχει το δικό της αλφάβητο. Με τους χαρακτήρες του αλφαβήτου της η γλώσσα σχηματίζει τις λέξεις ή τα σύμβολα που αποτελούν το λεξιλόγιο της γλώσσας. Το αλφάβητο της γλώσσας Java περιλαμβάνει τους ακόλουθους χαρακτήρες:

- οριζόντιος στηλοθέτης (αναπαριστάνεται με το σύμβολο \t)
- κατακόρυφος στηλοθέτης (αναπαριστάνεται με το σύμβολο \v)
- αλλαγής σελίδας
- νέας γραμμής (αναπαριστάνεται με το σύμβολο \n)
- γράμματα αγγλικής αλφαβήτου κεφαλαία και πεζά
- αριθμοί
- σύμβολα πράξεων + - * / %
- σύμβολα στίξης ( ) & { } [ ] , . κ.λ.π.

16.2Λεξιλόγιο γλώσσας

Το λεξιλόγιο της Java περιλαμβάνει τα ακόλουθα:

- δεσμευμένες λέξεις (reserved words)
- τελεστές (operators)
- αναγνωριστές (identifiers)

16.3Αριθμητικοί Τύποι Δεδομένων

Κάθε τύπος δεδομένων έχει μία κύρια (περιοχή) τιμών. Ο Compiler δεσμεύει χώρο μνήμης για να αποθηκεύσει κάθε μεταβλητή ή σταθερά σύμφωνα με τον τύπο των δεδομένων τους. Η Java παρέχει αρκετούς βασικούς (ενσωματωμένους) τύπους δεδομένων για αριθμητικές τιμές, χαρακτήρες, και Boolean τιμές. Σε αυτό το τμήμα, εισάγονται αριθμητικοί τύποι δεδομένων.

Η Java έχει έξι αριθμητικούς τύπους: 4 για ακέραιους και 2 για πραγματικούς αριθμούς. Ο ακόλουθος πίνακας κατηγοριοποιεί τους 6 αριθμητικούς τύπους δεδομένων, τους δεκαδικούς τους, και τα αποθηκευτικά τους μεγέθη.

Όνομα	**Εύρος τιμών**	**Μέγεθος Μνήμης**
byte	-2^7 (- 128) έως $2^7 - 1$ (127)	8-bit signed
short	-2^{15} (- 32768) έως $2^{15} - 1$ (32767)	16-bit signed
int	-2^{31} (-2147483648) έως $2^{31} - 1$ (2147483647)	32-bit signed
long	-2^{63} έως $2^{63} - 1$ (π.χ.. –	64-bit signed

	9223372036854775808 μέχρι 9223372036854775807)	
float	- 3.4E38 έως 3.4E38	32-bit IEEE 754
double	- 1.7E308 έως 1.7E308	64-bit IEEE 754

16.4Δεσμευμένες λέξεις

Οι δεσμευμένες λέξεις μιας γλώσσας έχουν αυστηρά καθορισμένη έννοια και τρόπο χρήσης και χρησιμοποιούνται σε συγκεκριμένες θέσεις ενός προγράμματος. Μερικές από τις πιο χαρακτηριστικές δεσμευμένες λέξεις είναι οι εξής:

- int, float, do, if, for, κ.λ.π.

16.5Αριθμητικές Σταθερές

Μια αριθμητική σταθερά είναι ένας βασικός τύπος τιμής ο οποίος εμφανίζεται κατευθείαν στο πρόγραμμα. Για παράδειγμα 34, 1000000 και 5.0 είναι αριθμητικές σταθερές στις ακόλουθες εντολές:

```
int i = 34
long l = 1000000;
double d = 5.0;
```

Οι αριθμητικές σταθερές πραγματικού τύπου είναι γραμμένα με ένα δεκαδικό σημείο. Από λάθος μια σταθερά πραγματικού τύπου

χρησιμοποιείται ως τύπου διπλής ακρίβειας. Για παράδειγμα 5.0 θεωρείται μια διπλής ακρίβειας τιμή όχι μια πραγματική τιμή. Μπορούμε να φτιάξουμε ένα αριθμητικό ή ένα πραγματικό αριθμό προσθέτοντας το γράμμα f, F, d ή D. Για παράδειγμα μπορούμε να χρησιμοποιήσουμε 100.2f ή 100.2F για ένα αριθμητικό αριθμό και 100.2d ή 100.2D για ένα πραγματικό αριθμό. Μια αριθμητική σταθερά μπορεί επίσης να γραφτεί με επιστημονική παράσταση. Για παράδειγμα 1.23456e + 2 που είναι ισοδύναμο του 1234565x 10^2 = 123.456

16.6Δήλωση Μεταβλητών

Οι μεταβλητές χρησιμοποιούνται για να αναπαραστήσουν πολλούς διαφορετικούς τύπους από δεδομένα. Για να χρησιμοποιήσουμε μια μεταβλητή, τη δηλώνουμε με το όνομα της καθώς και τον τύπο δεδομένων που αναπαριστά. Αυτό ονομάζεται δήλωση μεταβλητής. Δηλώνοντας μια μεταβλητή λέμε στον compiler πόση μνήμη να δεσμεύει για την μεταβλητή σύμφωνα με τον τύπο δεδομένων τους. Η σύνταξη για δήλωση μιας μεταβλητής είναι η ακόλουθη:

τύπος δεδομένων όνομα μεταβλητής;

Παραδείγματα δηλώσεων μεταβλητών

int x; // Δήλωση της x ως ακέραια;

double radius; // Δήλωση radius ως πραγματική μεταβλητή;

char a; // Δήλωση της a ως μεταβλητή χαρακτήρας;

16.7Δήλωση και Αρχικοποίηση μεταβλητών σε ένα βήμα

Οι μεταβλητές συχνά έχουν αρχικές τιμές. Μπορούμε να δηλώσουμε μια μεταβλητή και να την αρχικοποιήσουμε σε ένα βήμα. Για παράδειγμα παρατηρούμε τον ακόλουθο κώδικα: int x = 1; Αυτό είναι ισοδύναμο με τις επόμενες δύο δηλώσεις: int x; x= 1;

16.8Σταθερές

Η τιμή μιας μεταβλητής μπορεί να αλλάξει κατά τη διάρκεια εκτέλεσης του προγράμματος. Μια σταθερά αναπαριστά μόνιμα δεδομένα τα οποία ποτέ δεν αλλάζουν. Π.χ. το π είναι μια σταθερά που αν τη χρησιμοποιούμε συχνά δεν θέλουμε να πληκτρολογούμε συνεχώς την τιμή 3.14159; αντίθετα μπορούμε να καθορίσουμε μια σταθερά για το Π. Η σύνταξη για να καθορίσουμε μια σταθερά είναι η εξής:

***final* τύπος δεδομένων όνομα σταθεράς = τιμή;**

Η λέξη final είναι λέξη κλειδί της Java το οποίο σημαίνει ότι η σταθερά δεν μπορεί να αλλάξει.

Παράδειγμα

double radius, area;
final double pi = 3.14159; // Declare a constant

16.9Τελεστές

16.9.1 Αριθμητικοί Τελεστές

Οι αριθμητικοί τελεστές χρησιμοποιούνται για την εκτέλεση πράξεων. Στη γλώσσα Java είναι οι ακόλουθοι:

Αριθμητικοί Τελεστές
+ (άθροισμα)
- (διαφορά)
* (γινόμενο)
/ (πηλίκο)
% (υπόλοιπο)

π.χ.

int i1 = 34 + 1; //το i1 παίρνει την τιμή 35

double d1 = 34.0 – 0.1; //το d1 παίρνει την τιμή 33.9

long i2 = 300 * 30; //το i2 παίρνει την τιμή 9000

double d2 =1.0 /2.0; //το d2 παίρνει την τιμή 0.5

int i3 = 1 /2; //το i3 παίρνει την τιμή 0. Το αποτέλεσμα είναι το ακέραιο μέρος της διαίρεσης

byte i4 = 20%3; //το i4 παίρνει την τιμή 2. Το αποτέλεσμα είναι αυτό που μένει μετά τη διαίρεση

Το αποτέλεσμα μιας ακέραιας διαίρεσης είναι ένας ακέραιος. Για παράδειγμα 5/2 =2 όχι 2.5. Το κλασματικό (δεκαδικό) μέρος της απάντησης κόβεται. Ο τελεστής %δίνει το υπόλοιπο της διαίρεσης. Επιπλέον 7 %3 κάνει 1 και 20 %3 κάνει 2. Αυτός ο τελεστής είναι συχνά για ακεραίους, αλλά επίσης μπορεί να χρησιμοποιηθεί με τιμές πραγματικές. Οι τελεστές + και – μπορεί να είναι και οι δύο μοναδιαίοι και δυαδικοί. Ένας μοναδιαίος τελεστής έχει μόνο ένα τελεστέο ενώ ένας δυαδικός τελεστής έχει δύο τελεστέους. Για παράδειγμα το -5 μπορεί να θεωρηθεί σαν μοναδιαίος τελεστής για

να κάνει αρνητικό τον αριθμό 5, ενώ ο τελεστής στο 4 –5 είναι δυαδικός τελεστής για την αφαίρεση του 5 για 4.

16.9.2 Προσαρμογή τύπων

Γενικά η Java μπορεί να κάνει αυτόματα μετατροπή από έναν αριθμητικό τύπο σε ένα μεγαλύτερο τύπο όμως, πρέπει να χρησιμοποιήσουμε μια προσαρμογή. Μια προσαρμογή (cast) είναι μια ρητή μετατροπή μιας τιμής, από τον τρέχοντα τύπο της σε έναν άλλον. Η προσαρμογή εμφανίζεται με τον επιθυμητό τύπο μέσα σε παρενθέσεις πριν από την τιμή που πρόκειται να μετατραπεί, όπως εδώ:

float fvalue**;**

int ivalue **= (*int*)** fvalue**;** //μετατροπή τύπου ***float*** σε ***int***

double grades**;**

float grade **= (*float*)** grades**;** //μετατροπή τύπου ***double*** σε ***float***

16.9.3 Σχεσιακοί Τελεστές

Οι σχεσιακοί τελεστές χρησιμοποιούνται για να δημιουργήσουν συνθήκες. Στη γλώσσα Java είναι οι ακόλουθοι:

Σχεσιακοί Τελεστές
< (μικρότερο)
<=(μικρότερο ή ίσο)
> (μεγαλύτερο)
>=(μεγαλύτερο ή ίσο)
== (ισότητα)

!= (διάφορο)

Σημείωση

Πρέπει να τονίσουμε τη διαφορά ανάμεσα στην εντολή καταχώρισης η οποία συμβολίζεται με το = και τον τελεστή της ισότητας ο οποίος συμβολίζεται με ==.

16.9.4 Λογικοί Τελεστές

Οι λογικοί τελεστές χρησιμοποιούνται προκειμένου να συνδυάσουν πολλές απλές συνθήκες μέσα σε μια σύνθετη συνθήκη. Η τιμή μιας συνθήκης απλής ή σύνθετης είναι είτε true (αληθής) είτε false (ψευδής). Στη γλώσσα Java οι λογικοί τελεστές είναι οι ακόλουθοι είναι οι ακόλουθοι:

Λογικοί Τελεστές
&& (AND) ή &(AND)
|| (OR) ή >(OR)
! (NOT)

17 Εντολές Java

17.19. Εντολές Εισόδου-Εξόδου

17.1.1 Εντολή Εισόδου

Στη Java μπορούμε να εισάγουμε δεδομένα με 2 διαφορετικούς τρόπους: είτε από τη γραμμή εντολών (command line) είτε από το πληκτρολόγιο. Ακολούθως περιγράφουμε διεξοδικά αυτούς τους 2 τρόπους.

Α Τρόπος: Εισαγωγή Δεδομένων από Γραμμή εντολών

Όλα τα δεδομένα που εισάγουμε στη γραμμή εντολών κατά την εκτέλεση ενός προγράμματος Java θεωρούνται συμβολοσειρές (strings). Για να τα μετατρέψουμε στον επιθυμητό τύπο δεδομένων χρησιμοποιούμε τις ακόλουθες συναρτήσεις μετατροπής οι οποίες προσφέρονται από την ίδια τη γλώσσα):

Συνάρτηση	Λειτουργία
Integer.parseInt(String)	μετατροπή συμβολοσειράς αριθμητικών ψηφίων στον αντίστοιχο ακέραιο
Float.parseFloat(String)	μετατροπή συμβολοσειράς αριθμητικών ψηφίων στον αντίστοιχο πραγματικό απλής ακρίβειας
Double.parseDouble(String)	μετατροπή συμβολοσειράς αριθμητικών ψηφίων στον αντίστοιχο πραγματικό διπλής ακρίβειας

Β Τρόπος: Εισαγωγή Δεδομένων από Πληκτρολόγιο

Η εντολή (συνάρτηση) εισόδου για διάβασμα δεδομένων από το πληκτρολόγιο είναι η readLine() η οποία διαβάζει επίσης μια συμβολοσειρά (string) από το πληκτρολόγιο. Υλοποιείται με τον ακόλουθο κώδικα:

```
public static String readstring()
{
    BufferedReader br=new BufferedReader(new
InputStreamReader(System.in), 1);
    String st=" ";

    try
    {
      st=br.readLine();
    }
    catch (IOException ex)
    {
      System.out.print(ex);
    }

    return st;
}
```

Προκειμένου να μετατρέψουμε τα δεδομένα εισόδου τα οποία είναι πάντα συμβολοσειρά (τύπου String) όπως αναφέραμε, χρησιμοποιούμε τις ακόλουθες συναρτήσεις μετατροπής:

α)Συνάρτηση readint() μετατροπής συμβολοσειράς (String) σε ακέραιο (int)

```
public static int readint()
{
    return Integer.parseInt(readstring());
}
```

β)Συνάρτηση reafloat() μετατροπής συμβολοσειράς (String) σε πραγματικό απλής ακρίβειας (float)

```
public static float readfloat()
{
    return Float.parseFloat(readstring());
}
```

γ)Συνάρτηση readdouble() μετατροπής συμβολοσειράς (String) σε πραγματικό διπλής ακρίβειας(double)

```
public static double readdouble()
{
    return Double.parseDouble(readstring());
}
```

17.1.2 Εντολή Εξόδου

Η εντολή εξόδου χρησιμοποιείται για να εμφανίσουμε μηνύματα και αποτελέσματα στην οθόνη του Η/Υ. Έχει την ακόλουθη σύνταξη:

System.out.println ("σχόλιο"); για να τυπώσουμε ένα σχόλιο

ή

System.out.println (μεταβλητή); για να τυπώσουμε ένα αποτέλεσμα

ή

System.out.println ("σχόλιο"+μεταβλητή); για να τυπώσουμε ένα σχόλιο και ένα αποτέλεσμα

17.2Εντολή Καταχώρισης

Η εντολή καταχώρισης χρησιμοποιείται για την απόδοση τιμών σε μεταβλητές. Έχει την ακόλουθη σύνταξη:

```
μεταβλητή=τιμή;
μεταβλητή=μεταβλητή;
μεταβλητή=παράσταση;
```

Ακολούθως δίνονται παραδείγματα εντολών καταχώρισης:

x=5; //Στη μεταβλητή x καταχωρείται η τιμή 5

x=y; //Η τιμή της y καταχωρείται στην x

radius = 1.5; //Καταχώριση του 1.5 στη radius;

a = 'A';// Καταχώριση του χαρακτήρα 'A' στη μεταβλητή a;

x = 5*(3/2) + 3*2;// Καταχώριση του αποτελέσματος της παράστασης στη μεταβλητή x;

Βασικές Παρατηρήσεις

1. Σε μια εντολή καταχώρισης ο τύπος των δεδομένων της μεταβλητής στα αριστερά πρέπει να είναι συμβατός με τον τύπο των δεδομένων από της αξίας στα δεξιά. Για παράδειγμα x =1.0 θα μπορούσε να είναι λάθος αν ο τύπος της x είναι ***int*** δηλαδή δεν μπορούμε να καταχωρίσουμε μια πραγματική διπλής ακριβείας τιμή (1.0) σε μια ***int*** μεταβλητή χωρίς την κατάλληλη μετατροπή τύπου

2. Το όνομα της μεταβλητής πρέπει να είναι στα αριστερά. Έτσι, 1 = x μπορεί να είναι λάθος.

3. Μια έκφραση αναπαριστά έναν υπολογισμό που εμπεριέχει τιμές, μεταβλητές, και τελεστές.

17.3Εντολές Επιλογής

17.3.1Απλή επιλογή

Στην απλή επιλογή ελέγχεται μια συνθήκη και εφόσον είναι αληθής τότε εκτελείται μια ομάδα εντολών 1, ενώ αν είναι ψευδής τότε εκτελείται μια ομάδα εντολών 2. Έχει την ακόλουθη σύνταξη:

```
if (συνθήκη)
{
   ομάδα εντολών 1;
}
else
{
   ομάδα εντολών 2;
}
```

17.3.2Περιορισμένη επιλογή

Στην περιορισμένη επιλογή ελέγχεται μια συνθήκη και εφόσον είναι αληθής τότε εκτελείται μια ομάδα εντολών 1 ενώ αν είναι ψευδής τότε δεν εκτελείται τίποτα και το πρόγραμμα συνεχίζει απλώς στην επόμενη εντολή μετά το if. Έχει την ακόλουθη σύνταξη:

```
if (συνθήκη)
{
   ομάδα εντολών 1;
}
```

17.3.3 Εμφωλευμένη επιλογή

Στην εμφωλευμένη επιλογή ελέγχεται μια συνθήκη 1 και εφόσον είναι αληθής τότε εκτελείται μια ομάδα εντολών 1. Αν είναι ψευδής τότε ελέγχεται μια συνθήκη 2 και εφόσον είναι αληθής τότε εκτελείται μια ομάδα εντολών 2. Αν είναι ψευδής τότε ελέγχεται μια συνθήκη 3 κ.ο.κ. Συνολικά ελέγχονται n συνθήκες και εκτελείται μόνο μια ομάδα εντολών. Αν δεν είναι καμία συνθήκη αληθής τότε εκτελείται μια ομάδα εντολών n+1. Έχει την ακόλουθη σύνταξη:

```
if (συνθήκη 1)
{
   ομάδα εντολών 1;
}
else
   if (συνθήκη 2)
   {
      ομάδα εντολών 2;
   }
   else
      …….
      if (συνθήκη n)
      {
```

```
    ομάδα εντολών n;
}
else
{
    ομάδα εντολών n+1;
}
```

Παρατηρήσεις

1. Σε κάθε εντολή επιλογής η συνθήκη είναι μια λογική παράσταση η οποία παίρνει είτε την τιμή αλήθεια (true) είτε την τιμή ψέμα (false). Το 0 αντιστοιχεί στη λογική τιμή αλήθεια και ένας αριθμός ≠0 αντιστοιχεί στην λογική τιμή ψέμα.

2. Κάθε ομάδα εντολών μπορεί να είναι είτε μια μόνο εντολή ή ένα μπλοκ από εντολές μέσα σε άγκιστρα. Αν είναι μόνο μια εντολή τότε μπορούμε να παραλείψουμε τα άγκιστρα.

17.3.4 Τριαδικός τελεστής

Ο τριαδικός τελεστής είναι συντομευμένη μορφή της εντολής if. Έχει την ακόλουθη σύνταξη:

```
συνθήκη?εντολή1:εντολή2
```

Αν η συνθήκη είναι αληθής τότε εκτελείται η εντολή1 αλλιώς εκτελείται η εντολή2.

Παράδειγμα

```
z=(x>y)?5:10
```

δηλαδή αν το x>y τότε στο z καταχωρείται η τιμή 10 αλλιώς τότε στο z καταχωρείται η τιμή 5.

17.3.5 Εντολή πολλαπλής επιλογής switch

Η εντολή πολλαπλής επιλογής χρησιμοποιείται όταν ελέγχουμε την τιμή μιας μεταβλητής (όχι συνθήκης) και ανάλογα με την τιμή αυτή εκτελούμε είτε την ομάδα εντολών 1 είτε την ομάδα εντολών 2 είτε την ομάδα εντολών ν. Αν η μεταβλητή δεν πάρει καμία από τις τιμές 1, 2, ... ν τότε εκτελείται η ομάδα εντολών που βρίσκεται στο default. Η εντολή switch γενικά έχει την ακόλουθη μορφή:

```
switch (μεταβλητή)
{
   case τιμή 1:     ομάδα εντολών 1;
                    break;

   case τιμή 2:     ομάδα εντολών 2;
                    break;

   case τιμή ν:     ομάδα εντολών ν;
                    break;

   default:         ομάδα εντολών ν+1;
}
```

Σημείωση

Η εντολή break είναι προαιρετική. Όμως ουσιαστικά είναι επιβεβλημένη η χρησιμοποίηση της γιατί αν απουσιάζει από το τέλος εντολών μιας ομάδας τότε εκτελούνται οι εντολές και της επόμενης ομάδας γεγονός βέβαια που δεν έχει νόημα. Συνεπώς η break στην εντολή switch απαγορεύει την εκτέλεση των εντολών των υπολοίπων ομάδων πέρα της ομάδας εντολών που θα εκτελεστεί.

17.4Εντολές Επανάληψης

Οι εντολές επανάληψης (ανακύκλωσης) προκαλούν την εκτέλεση μιας εντολής ή μιας ομάδας εντολών μέσα σε άγκιστρα είτε ένα προκαθορισμένο αριθμό επαναλήψεων είτε για όσο χρόνο μια συνθήκη είναι αληθής. Στη Java χρησιμοποιούμε 3 εντολές επανάληψης, τις for, while και do..while οι οποίες έχουν **ακριβώς την ίδια σύνταξη και την ίδια λειτουργία με αυτές που έχουμε περιγράψει στη C++** οπότε παραλείπεται η περιγραφή τους

17.5Συναρτήσεις (υποπρογράμματα)

Οι μέθοδοι είναι αντίστοιχες με τις λειτουργίες (functions) της C, τις διαδικασίες και λειτουργίες (procedures and functions) της Pascal και τις λειτουργίες και υπορουτίνες (functions and subroutines) της Fortran.

Μπορούμε να γράψουμε και να καλέσουμε τις δικές μας μεθόδους. Ας δούμε ένα απλό παράδειγμα. Το παρακάτω είναι ένα απλό πρόγραμμα που ζητά έναν αριθμό από τον χρήστη και μετά υπολογίζει την παράγωγο του αριθμού. Θα χρησιμοποιήσουμε δύο μεθόδους στο πρόγραμμα, μία που ελέγχει αν ο χρήστης εισήγαγε έναν έγκυρο θετικό ακέραιο και μία άλλη που υπολογίζει το

παραγοντικό του. Θα ξεκινήσουμε γράφοντας την κύρια μέθοδο του προγράμματος:

```
class Factorial
{
   public static void main(String [] args)
   {
      int n;

      while ((n = getNextInteger()) >= 0)
      System.out.println (factorial(n));
   }
}
```

Εκτός των άλλων ο κώδικας αυτός αποδεικνύει ότι οι μέθοδοι καθιστούν δυνατό να σχεδιαστεί η ροή του προγράμματος χωρίς περιττές λεπτομέρειες. Απλά ονομάσαμε δύο μεθόδους getNextInteger() και factorial() χωρίς να ανησυχούμε για την εφαρμογή τους. Μπορούμε να προσθέσουμε τον υπόλοιπο κώδικα σε μικρότερα, ευκολονόητα τμήματα. Ας γράψουμε πρώτα τη μέθοδο factorial:

```
long factorial (long n)
{
   int i;
   long result=1;

   for (i=1; i <= n; i++)
```

```
            result *= i;


    return result;
}
```

Θα μπορούσαμε να συμπεριλάβουμε αυτό τον κώδικα στη μέθοδο main αλλά είναι πιο εύκολο να κατανοήσουμε τον αλγόριθμο σπάζοντας τον κώδικα σε μικρότερα τμήματα. Είναι επίσης πιο εύκολο να ελέγξουμε και να διορθώσουμε. Μπορούμε να γράψουμε ένα απλό πρόγραμμα που θα μας επιτρέπει να ελέγχουμε τη μέθοδο factorial προτού μας απασχολήσει το αν είναι αποδεκτή και έγκυρη η είσοδος του χρήστη. Ακολουθεί το πρόγραμμα ελέγχου:

```
class FactorialTest
{
   public static void main(String [] args)
   {
      int n;
      int i;
      long result;

      for (i=1; i <=10; i++)
      {
         result = factorial(i);
         System.out.println (result);
      }
   }
```

```
   static long factorial (int n)
   {
      int i;
      long result=1;


      for (i=1; i <= n; i++)
      result *= i;
      return result;
   }
}
```

Οι προγραμματιστές της C++ θα πρέπει να προσέξουν ότι και οι δύο μέθοδοι προσδιορίζονται μέσα σε ορισμό κλάσης (class definition). Παρατηρούμε για άλλη μια φορά ότι τα πάντα στη Java ανήκουν σε κλάσεις. Ας δούμε με μια πιο προσεκτική ματιά τη σύνταξη της μεθόδου:

```
static long factorial (int n)
{
      int i;
      long result=1;


      for (i=1; i <= n; i++)
          result *= i;
      return result;
}
```

Οι μέθοδοι ξεκινούν με μία **δήλωση**. Αυτό μπορεί να περιλαμβάνει από 3 έως 5 τμήματα. Πρώτα είναι ένας **προαιρετικός καθαριστής πρόσβασης** (optional access specifier), που μπορεί να είναι γενικός (***public***), ιδιωτικός (***private***) ή προστατευόμενος (protected). Μία γενική μέθοδο μπορούμε να την καλέσουμε από παντού. Μία ιδιωτική μέθοδος μπορεί να χρησιμοποιηθεί μόνο από την κλάση που προσδιορίζεται. Μία προστατευμένη μέθοδος μπορεί να χρησιμοποιηθεί οπουδήποτε εντός του πακέτου που προσδιορίζεται. Οι μέθοδοι που δεν έχουν δηλωθεί σαν γενικές ή ιδιωτικές θεωρούνται προστατευόμενες. Στη συνέχεια αποφασίζουμε αν μια μέθοδος είναι **στατική ή όχι**. Οι στατικές μέθοδοι έχουν ένα στιγμιότυπο ανά κλάση παρά ένα παράδειγμα ανά αντικείμενο. Όλα τα αντικείμενα μιας κλάσης μοιράζονται ένα αντίγραφο της στατικής μεθόδου. Εξορισμού οι μέθοδοι δεν είναι στατικές. Στη συνέχεια εξειδικεύουμε τον **τύπο επιστροφής**. Αυτό είναι η τιμή που θα επιστραφεί στην καλούμενη μέθοδο όταν τελειώσουν όλοι οι υπολογισμοί μέσα στη μέθοδο. Για παράδειγμα αν ο τύπος επιστροφής είναι ***int*** μπορούμε να χρησιμοποιήσουμε τη μέθοδο όπου χρησιμοποιούμε ένα σταθερό ακέραιο. Αν ο τύπος επιστροφής είναι ***void*** τότε δεν επιστρέφεται καμία τιμή. Έπειτα ακολουθεί το **όνομα της μεθόδου**. Στη συνέχεια έχουμε τις **παρενθέσεις**. Μέσα στις παρενθέσεις **δίνουμε ονόματα και τύπους στα ορίσματα της μεθόδου**. Μία μέθοδος μπορεί να έχει κανένα, ένα ή πολλά ορίσματα. Τα ορίσματα αυτά μπορούν να χρησιμοποιηθούν μέσα στη μέθοδο όπως οι τοπικές μεταβλητές. Τέλος το **υπόλοιπο της μεθόδου** περικλείεται μέσα σε άγκιστρα τα οποία το κάνουν ένα απλό μπλοκ. Το τμήμα μέσα στα άγκιστρα είναι όπως οι κύριες μέθοδοι. Υπάρχουν δηλώσεις μεταβλητών, κώδικας και τέλος μία δήλωση επιστροφής, η οποία στέλνει μια

τιμή πίσω στην καλούμενη μέθοδο. Ο τύπος της τιμής αυτής πρέπει να συμφωνεί με τον τύπο που δηλώθηκε στη μέθοδο.

18 Πίνακες

Οι πίνακες (Arrays) είναι ο πιο αποτελεσματικός τρόπος αποθήκευσης συνόλων μεταβλητών. Ένας πίνακας είναι ένα σύνολο μεταβλητών που μοιράζονται το ίδιο όνομα και συντάσσονται με τη σειρά από το 0 μέχρι το πλήθος των μεταβλητών μειωμένο κατά 1. Ο αριθμός των μεταβλητών που μπορεί να αποθηκευτεί ονομάζεται διάσταση του πίνακα. Κάθε μεταβλητή ονομάζεται στοιχείο του πίνακα. Ένας πίνακας μπορεί να παρασταθεί σαν στήλη δεδομένων όπως πιο κάτω:

I[0]	1
I[1]	2
I[2]	76
I[3]	-90
I[4]	2

Σε αυτή την περίπτωση βλέπουμε ένα ακέραιο πίνακα που ονομάζεται I με 5 στοιχεία. Ο τύπος του πίνακα είναι int και έχει διάσταση 5.

18.1Μονοδιάστατοι πίνακες

18.1.1Δήλωση Πινάκων

Όπως και οι άλλες μεταβλητές στη Java ένας πίνακας πρέπει να έχει ένα συγκεκριμένο τύπο όπως byte, ***int***, ***String*** ή ***double***. Μόνο μεταβλητές του αντίστοιχου τύπου μπορούν να αποθηκευτούν σε έναν πίνακα. Για παράδειγμα δεν μπορούμε να έχουμε ένα πίνακα που να αποθηκεύει και ***int*** και ***String***. Όπως και οι άλλες μεταβλητές στη Java έτσι και οι πίνακες πρέπει να δηλώνονται.

Όταν δηλώνουμε ένα πίνακα βάζουμε στον τύπο και το [] που δείχνει ότι η μεταβλητή είναι ένας πίνακας. Ακολουθούν μερικά παραδείγματα:

int[] k;

float[] yt;

String[] names;

Με άλλα λόγια δηλώνουμε έναν πίνακα όπως ακριβώς δηλώνουμε οποιαδήποτε άλλη μεταβλητή μόνο που βάζουμε και αγκύλες στο τέλος του τύπου της μεταβλητής.

18.1.2 Ορισμός Πινάκων

Για να δημιουργήσουμε ένα πίνακα χρησιμοποιούμε τον τελεστή ***new***. Πρέπει να πούμε στον compiler πόσα στοιχεία θα αποθηκευτούν στον πίνακα. Πιο κάτω βλέπουμε πώς δημιουργούμε τις μεταβλητές:

k = ***new int***[3];

yt = ***new float***[7];

names = ***new String***[50];

Τα νούμερα στις αγκύλες καθορίζουν **τη διάσταση του πίνακα δηλαδή πόσες θέσεις έχει**. Ο k μπορεί να κρατήσει 3 ακέραιους (***int***), ο yt μπορεί να κρατήσει 7 πραγματικούς απλής ακρίβειας (***float***) και ο names μπορεί να κρατήσει 50 συμβολοσειρές (***String***). Συνήθως αυτό ονομάζεται ορισμός του πίνακα αφού καθορίζει το τμήμα της RAM που χρειάζεται ο πίνακας. Εδώ βλέπουμε για πρώτη φορά τον τελεστή ***new***. Ο ***new*** είναι δεσμευμένη λέξη στη Java και

χρησιμοποιείται όχι μόνο για να ορίσει πίνακες αλλά και άλλα είδη αντικειμένων.

18.1.3 Αρχικοποίηση Πινάκων

Κάθε στοιχείο του πίνακα προσδιορίζεται από το όνομα του πίνακα και από έναν ακέραιο που φανερώνει τη θέση του στον πίνακα. Οι αριθμοί που χρησιμοποιούνται ονομάζονται δείκτες του πίνακα. Οι δείκτες είναι συνεχόμενοι αριθμοί που ξεκινούν από το 0. Γιαυτό και ο πίνακας k έχει τα στοιχεία k[0], k[1] και k[2]. Εφόσον ξεκινάμε από το 0 δεν υπάρχει k[3] και αν προσπαθήσουμε να έχουμε πρόσβαση σε αυτό θα δημιουργήσουμε ένα ArrayIndexOutOfBoundsException. Παρακάτω θα δούμε πώς αποθηκεύουμε τιμές στους πίνακες:

```
k[0] = 2;
k[1] = 5;
k[2] = -2;
yt[6] = 7.5f;
names[4] = "Fred";
```

Αυτό το βήμα ονομάζεται **αρχικοποίηση του πίνακα ή καλύτερα αρχικοποίηση των στοιχείων του πίνακα**. Ακόμα και για τους πίνακες μεσαίου μεγέθους είναι σπάνιο να αρχικοποιούμε κάθε στοιχείο ξεχωριστά. Συχνά είναι πιο εύκολο να χρησιμοποιούμε τον βρόγχο for. Για παράδειγμα ακολουθεί μια επανάληψη που γεμίζει ένα πίνακα με τα τετράγωνα των αριθμών από το 0 ως το 100.

```
float[] squares = new float[101];

for (int i=0, i <= 100; i++)
        squares[i] = i*i;
```

Θα πρέπει να προσέξουμε 2 σημεία του κώδικα:

- Εφόσον οι δείκτες ξεκινούν από το 0 χρειαζόμαστε 101 στοιχεία αν θέλουμε να συμπεριλάβουμε και το τετράγωνο του 100.
- Παρόλο που το i είναι ***int*** γίνεται ***float*** όταν αποθηκεύει τετράγωνο αφού δηλώσαμε τα τετράγωνα σαν πίνακα ***float***.

Ο τρόπος για να αποφύγουμε το πρώτο σημείο είναι το παράδειγμα που ακολουθεί:

```
float[] squares = new float[101];
for (int i=0, i < squares.length; i++)
    squares[i] = i*i;
```

18.2Δισδιάστατοι πίνακες

Ο πιο συνηθισμένος τύπος πολυδιάστατου πίνακα είναι ο δισδιάστατος. Ένας δισδιάστατος πίνακας σαν πίνακας τιμών είναι όπως ο παρακάτω:

3	3	1	9
4	-2	3	0
5	67	65	2

45	7	2	3
87	8	0	8

Εδώ έχουμε ένα πίνακα με 5 γραμμές και 4 στήλες. Έχει συνολικά 20 στοιχεία. Έχουμε πει ότι έχει διαστάσεις 5x4 , όχι 20. Αυτός ο πίνακας δεν είναι ίδιος με τον 4x5 πίνακα που ακολουθεί πιο κάτω:

3	23	1	9	5
4	-2	3	0	6
5	67	65	2	7
45	7	2	3	9
87	8	0	8	0

Πρέπει να χρησιμοποιούμε 2 αριθμούς για να αναγνωρίσουμε τη θέση σε ένα δισδιάστατο πίνακα. Αυτές είναι οι θέσεις των γραμμών και των στηλών που βρίσκονται τα στοιχεία. Για παράδειγμα αν ο παραπάνω πίνακας λέγεται J τότε J[0][0] είναι 3, J[0][1] είναι 23, J[0][2] είναι 1, J[0][3] είναι 9, J[1][0] είναι 4 κτλ.

Οι δισδιάστατοι πίνακες δηλώνονται, ορίζονται και αρχικοποιούνται όπως και οι μονοδιάστατοι πίνακες. Γιαυτό πρέπει να καθορίζουμε 2 διαστάσεις αντί για μία και να χρησιμοποιήσουμε δύο εμφωλευμένες εντολές επανάληψης για όλες τις λειτουργίες των πινάκων

18.2.1 Γενικοί Αλγόριθμοι Μονοδιάστατων Πινάκων στη Java

18.2.1.1 Διάβασμα Μονοδιάστατου Πίνακα

Για να διαβάσουμε (γεμίσουμε) ένα μονοδιάστατο πίνακα με τυχαίες τιμές γράφουμε τον ακόλουθο κώδικα:[4]

```
static void diabasma(int x[])
{
    int i;

    for (i=0;i<5;i++)
        x[i]=(int)(Math.random()*100)+1;
}
```

Για να διαβάσουμε (γεμίσουμε) ένα μονοδιάστατο πίνακα εισάγοντας τιμές από το πληκτρολόγιο γράφουμε τον ακόλουθο κώδικα:[5]

```
static void diabasma(int x[])
{
    int i;

    for (i=0; i<x.length; i++)
    {
        System.out.println ("Δώσε "+(i+1)+" στοιχείο");
        x[i]=readint();
    }
```

[4] Υποθέτουμε ότι σε όλες τις λειτουργίες που αναφέρουμε ότι έχει προηγηθεί η δήλωση ***int*** x[5];
[5] Υποθέτουμε ότι σε όλες τις λειτουργίες που αναφέρουμε ότι έχει προηγηθεί η δήλωση ***int*** x[5];

```
}
```

Οι συναρτήσεις readstring() που διαβάζει συμβολοσειρά από πληκτρολόγιο και readint που μετατρέπει τη συμβολοσειρά στον αντίστοιχο ακέραιο παραθέτονται ακολούθως:

```
public static String readstring()
{
    BufferedReader br=new BufferedReader(new
InputStreamReader(System.in), 1);
    String st=" ";

    try
    {
      st=br.readLine();
    }
    catch (IOException ex)
    {
      System.out.print(ex);
    }

    return st;
}

public static int readint()
{
    return Integer.parseInt(readstring());
}
```

Παρατηρήσεις

1. Για την υλοποίηση της συνάρτησης readstring είναι απαραίτητη η εντολή import java.io.* στη αρχή της κλάσης που περιέχει τη συνάρτηση readstring. Η εντολή import είναι αντίστοιχη της include στη C++ και η συγκεκριμένη εισάγει στο πρόγραμμα όλες τις συναρτήσεις του πακέτου java.io.
2. Η συνάρτηση random την οποία λαμβάνουμε μέσα από το πακέτο Math επιστρέφει ένα τυχαίο double αριθμό από 0 μέχρι τον προηγούμενο από αυτόν που γράφουμε στο *. Π.χ. γράφοντας Math.random()*100 παράγεται ένας τυχαίος αριθμός από 0 μέχρι 99. Η δήλωση (int) που γράφουμε μπροστά από την random() μετατρέπει τον double αριθμό της rand σε int. Στο αποτέλεσμα προσθέτουμε 1 για να πάρουμε ακέραιο από 1 μέχρι 100

18.2.1.2 Εκτύπωση Μονοδιάστατου Πίνακα

Για να εκτυπώσουμε ένα μονοδιάστατο πίνακα γράφουμε τον ακόλουθο κώδικα:[6]

```
static void ektyposi(int x[])
{
    int i;

    for (i=0;i<5;i++)
        System.out.print(x[i]+"\t");
```

[6] Υποθέτουμε ότι σε όλες τις λειτουργίες που αναφέρουμε ότι έχει προηγηθεί η δήλωση ***int*** x[5];

```
        System.out.println();
}
```

18.2.1.3 Υπολογισμός μέσου όρου Μονοδιάστατου Πίνακα

Για να υπολογίσουμε το μέσο όρο των στοιχείων ενός μονοδιάστατου πίνακα αθροίζουμε πρώτα όλα τα στοιχεία του πίνακα και στη συνέχεια διαιρούμε το άθροισμα με το συνολικό πλήθος των στοιχείων του πίνακα.

Αυτή η λειτουργία υλοποιείται με τον ακόλουθο κώδικα:

```
static float mesos_oros(int x[])
{
        int i,sum=0;

        for (i=0;i<5;i++)
                sum+=x[i];

        return (float)sum/5;
}
```

18.2.1.4 Υπολογισμός πλήθους άρτιων-περιττών στοιχείων

Για να υπολογίσουμε το πλήθος άρτιων και περιττών στοιχείων όλου του πίνακα χρησιμοποιούμε 2 μετρητές, τον μετρητή art ο οποίος υπολογίζει το πλήθος των άρτιων στοιχείων και το μετρητή per ο οποίος υπολογίζει αντίστοιχα το πλήθος των περιττών στοιχείων του πίνακα. Οι 2 μετρητές πρέπει να αρχικοποιηθούν με

μηδέν πριν την επανάληψη. Η λειτουργία αυτή υλοποιείται με το ακόλουθο τμήμα κώδικα:

```
static void artioi_perittoi(int x[])
{
    int i, a=0,p=0;

    for (i=0; i<x.length; i++)
        if (x[i]%2==0)
            a++;
        else
            p++;

    System.out.println ("Άρτιοι = "+a+" Περιττοί = "+p);
}
```

18.2.1.5 Υπολογισμός πλήθους πρώτων στοιχείων πίνακα

Για να υπολογίσουμε το πλήθος των πρώτων στοιχείων του πίνακα (πρώτοι αριθμοί είναι αυτοί που διαιρούνται μόνο με τον εαυτό τους και τη μονάδα) μετράμε τους διαιρέτες του κάθε στοιχείου του πίνακα και αν το πλήθος των διαιρετών είναι <=2 τότε το στοιχείο είναι πρώτος αριθμός αλλιώς όχι. Τους πρώτους αριθμούς τους μετράμε με ένα ξεχωριστό μετρητή ενώ τους μη πρώτους τους αγνοούμε.

Η λειτουργία αυτή υλοποιείται με το ακόλουθο τμήμα κώδικα:

```
static int protoi_arithmoi (int x[])
```

```
{
    int i, j, p=0, d;

    for (i=0;i<x.length;i++)
    {
        d=0;

        for (j=1;j<=x[i];j++)
            if (x[i]%j==0)
                d++;

        if (d<=2)
            p++;
     }

    return p;
}
```

Παρατήρηση

Ο μετρητής d που υπολογίζει το πλήθος διαιρετών του κάθε στοιχείου του πίνακα μηδενίζεται σε κάθε εξωτερική επανάληψη η οποία «σαρώνει» τα στοιχεία του πίνακα.

18.2.1.6 Υπολογισμός πλήθους τέλειων στοιχείων πίνακα

Για να υπολογίσουμε το πλήθος των τέλειων στοιχείων του πίνακα (τέλειοι αριθμοί είναι αυτοί που το άθροισμα των διαιρετών τους ισούται με το διπλάσιο του αριθμού) αθροίζουμε τους

διαιρέτες του κάθε στοιχείου του πίνακα και ελέγχουμε αν το άθροισμα αυτό είναι ίσο με το διπλάσιο του στοιχείου οπότε τότε το στοιχείο είναι τέλειος αριθμός αλλιώς όχι. Τους τέλειους αριθμούς τους μετράμε με ένα ξεχωριστό μετρητή ενώ τους μη τέλειους τους αγνοούμε.

Η λειτουργία αυτή υλοποιείται με το ακόλουθο τμήμα κώδικα:

```
static int teleioi_arithmoi (int x[])
{
    int i, j, t=0, sum;

    for (i=0;i<x.length;i++)
    {
        sum=0;

        for (j=1; j<=x[i]; j++)
            if (x[i]%j==0)
                sum+=j;

        if (sum<=2*x[i])
            t++;
     }

    return t;
}
```

Παρατήρηση

Ο μετρητής sum που υπολογίζει το άθροισμα διαιρετών του κάθε στοιχείου του πίνακα μηδενίζεται σε κάθε εξωτερική επανάληψη η οποία «σαρώνει» τα στοιχεία του πίνακα.

18.2.1.7 Εύρεση μέγιστου-ελάχιστου στοιχείου πίνακα και των θέσεων τους σε μονοδιάστατο πίνακα

Για να βρούμε το μέγιστο και το ελάχιστο στοιχείο του πίνακα και τις θέσεις τους στον πίνακα υποθέτουμε αρχικά ότι ο μέγιστο και το ελάχιστο στοιχείο του πίνακα είναι το πρώτο (αποθηκεύουμε το μέγιστο και το ελάχιστο στοιχείο του πίνακα στις μεταβλητές max και min αντίστοιχα). Επίσης στις μεταβλητές maxp και minp αποθηκεύουμε τη θέση του μέγιστου και του ελάχιστο στοιχείο του πίνακα αντίστοιχα. Η λειτουργία αυτή υλοποιείται με το ακόλουθο τμήμα κώδικα:

```
static void max_min (int x[])
{
    int i, max, maxp, min, minp;

    max=min=x[0];
    maxp=minp=0;

    for (i=1; i<x.length; i++)
    {
        if (x[i]>max)
        {
            max=x[i];
            maxp=i;
```

```
        }

        if (x[i]<min)
        {
            min=x[i];
            minp=i;
        }
    }
    System.out.println ("Μέγιστος = "+max+" στη θέση "+
(maxp+1));

    System.out.println ("Ελάχιστος = "+min+" στη θέση
"+(minp+1));
}
```

18.2.2 Αλγόριθμοι Αναζήτησης σε Μονοδιάστατους Πίνακες

Οι αλγόριθμοι αναζήτησης σε μονοδιάστατους πίνακες είναι όλοι ίδιοι με αυτούς που περιγράψαμε στη C++ οπότε για λόγους συντομίας παραλείπουμε την περιγραφή τους.

18.2.3 Αλγόριθμοι Ταξινόμησης σε Μονοδιάστατους Πίνακες

Οι αλγόριθμοι ταξινόμησης σε μονοδιάστατους πίνακες είναι επίσης όλοι ίδιοι με αυτούς που περιγράψαμε στη C++ οπότε για λόγους συντομίας παραλείπουμε την περιγραφή τους.

18.2.4 Αριθμητικές Πράξεις Μονοδιάστατων Πινάκων

Οι αριθμητικές πράξεις μονοδιάστατων πινάκων είναι επίσης ίδιες με αυτές που περιγράψαμε στη C++, όμως τώρα παραθέτουμε

ένα πλήρες πρόγραμμα για αυτές προκειμένου να τονίσουμε τις ιδιαιτερότητες της Java όπως για παράδειγμα τη δήλωση των πινάκων και τη δέσμευση μνήμης γιαυτούς καθώς επίσης το διάβασμα και την εκτύπωση των πινάκων.

Ακολούθως παρατίθεται το πλήρες πρόγραμμα:

```
import java.io.*;

public class arithmitikes_praxeis
{
    public static void main(String [] args)
    {
        int x[]=new int[5];
        int y[]=new int[5];
        int a[]=new int[5];
        int p[]=new int[5];

        System.out.println("Διάβασμα Πίνακα x");
        diabasma(x);

        System.out.println ("Διάβασμα Πίνακα y");
        diabasma(y);

        athroisma(x, y, a);

        ginomeno(x, y, p);

        System.out.println("Πίνακας x");
        ektyposi(x);
```

```
        System.out.println("Πίνακας y");

        ektyposi(y);

        System.out.println("Άθροισμα Πινάκων");

        ektyposi(a);

        System.out.println("Γινόμενο Πινάκων");

        ektyposi(p);
    }

    static void diabasma(int x[])
    {
        int i;

        for (i=0;i<x.length;i++)
        {
            System.out.println("Δώσε "+(i+1)+" στοιχείο");
            x[i]=readint();
        }
    }

    static void ektyposi(int x[])
    {
        int i;
```

```
        for (i=0; i<x.length; i++)
                System.out.print(x[i]+"\t");

        System.out.println();
    }

    static void athroisma(int x[], int y[], int a[])
    {
        int i;

        for (i=0;i<x.length;i++)
                a[i]=x[i]+y[i];
    }

    static void ginomeno(int x[], int y[], int p[])
    {
        int i;

        for (i=0;i<x.length;i++)
                p[i]+=x[i]*y[i];
    }
}
```

Παρατήρηση

Για τους πίνακες που χρησιμοποιούνται στο πρόγραμμα είναι απαραίτητη η δέσμευση μνήμης και αυτή γίνεται στο main με την εντολή new και το μέγεθος του κάθε πίνακα.

18.2.5 Γενικοί Αλγόριθμοι Δισδιάστατων Πινάκων στη Java

18.2.5.1 Διάβασμα Δισδιάστατου Πίνακα

Για να διαβάσουμε (γεμίσουμε) ένα δισδιάστατο πίνακα **κατά γραμμές** με τυχαίες τιμές π.χ. από 1 μέχρι 100 γράφουμε τον ακόλουθο κώδικα:[7]

```
static void diabasma(int x[][])
{
    int i, j;

    for (i=0;i<3;i++)
        for (j=0;j<3;j++)
            x[i][j]=(int)(Math.random()*100)+1;
}
```

Για να διαβάσουμε (γεμίσουμε) ένα δισδιάστατο πίνακα **κατά στήλες** με τυχαίες τιμές π.χ. από 1 μέχρι 100 γράφουμε τον ακόλουθο κώδικα:

```
static void diabasma(int x[][])
{
    int i, j;

    for (j=0;j<3;j++)
        for (i=0;i<3;i++)
            x[i][j]=(int)(Math.random()*100)+1;
```

[7] Υποθέτουμε ότι σε όλες τις λειτουργίες που αναφέρουμε ότι έχει προηγηθεί η δήλωση ***int*** x[3][3];

```
}
```

Για να διαβάσουμε (γεμίσουμε) ένα δισδιάστατο πίνακα **κατά γραμμές** εισάγοντας τιμές από το πληκτρολόγιο γράφουμε τον ακόλουθο κώδικα:

```
static void diabasma(int x[][])
{
    int i, j;

    for (i=0;i<3;i++)
        for (j=0;j<3;j++)
        {
            System.out.println ("Δώσε "+(i+1)+", "+(j+1)+ " στοιχείο");
            x[i][j]=(int)(Math.random()*100)+1;
        }
}
```

Για να διαβάσουμε (γεμίσουμε) ένα δισδιάστατο πίνακα κατά στήλες εισάγοντας τιμές από το πληκτρολόγιο γράφουμε τον ακόλουθο κώδικα:

```
static void diabasma(int x[][])
{
    int i, j;
```

```
    for (j=0;j<3;j++)
        for (i=0;i<3;i++)
        {
            System.out.println ("Δώσε "+(j+1)+",”+”(i+1)+” στοιχείο");
            x[i][j]=(int)(Math.random()*100)+1;
        }
}
```

18.2.5.2 Εκτύπωση Δισδιάστατου Πίνακα

Για να εκτυπώσουμε ένα δισδιάστατο πίνακα γράφουμε τον ακόλουθο κώδικα:

```
static void ektyposi(int x[][])
{
    int i, j;

    for (i=0; i<3; i++)
    {
        for (j=0;j<3;j++)
            System.out.print(x[i][j]+"\t");

        System.out.println();
    }
}
```

18.2.5.3 Υπολογισμός Μέσου Όρου Δισδιάστατου Πίνακα

Για να υπολογίσουμε το μέσο όρο των στοιχείων ενός δισδιάστατου πίνακα **με συνάρτηση η οποία τον επιστρέφει** γράφουμε το ακόλουθο τμήμα κώδικα:

```
static float mesos_oros(int x[][])
{
    int i, j, sum=0;

    for (i=0;i<3;i++)
        for (j=0;j<3;j++)
            sum+=x[i][j];

    return (float)sum/9;
}
```

Για να υπολογίσουμε το μέσο όρο των στοιχείων ενός δισδιάστατου πίνακα **με συνάρτηση η οποία τον τυπώνει** γράφουμε το ακόλουθο τμήμα κώδικα:

```
void float mesos_oros(int x[][])
{
    int i, j, sum=0;

    for (i=0;i<3;i++)
        for (j=0;j<3;j++)
            sum+=x[i][j];
```

```
    System.out.println("Μέσος όρος = "+(float)sum/9);
}
```

18.2.5.4 Υπολογισμός Μέσου Όρου Κάθε Γραμμής Δισδιάστατου Πίνακα

Για να υπολογίσουμε το μέσο όρο των στοιχείων κάθε γραμμής ξεχωριστά ενός δισδιάστατου πίνακα γράφουμε το ακόλουθο τμήμα κώδικα:

```
static void mo_grammon(int x[][])
{
    int i, j, sum;
    float mo=0;

    for (i=0;i<3;i++)
    {
        sum=0;

        for (j=0;j<3;j++)
            sum+=x[i][j];

        mo=(float)sum/3;
        System.out.println("Μέσος όρος "+(i+1)+" γραμμής="+mo);
    }
}
```

18.2.5.5 Υπολογισμός Μέσου Όρου Κάθε Στήλης Δισδιάστατου Πίνακα

Για να υπολογίσουμε το μέσο όρο των στοιχείων κάθε στήλης ξεχωριστά ενός δισδιάστατου πίνακα γράφουμε το ακόλουθο τμήμα κώδικα:

```
static void mo_stilon(int x[][])
{
    int i, j, sum;
    float mo=0;

    for (j=0;j<3;j++)
    {
        sum=0;

        for (i=0;i<3;i++)
            sum+=x[i][j];

        mo=(float)sum/3;
        System.out.println ("Μέσος όρος "+(j+1)+" στήλης ="+mo);
    }
}
```

18.2.5.6 Υπολογισμός Μέγιστου-Ελάχιστου στοιχείου Δισδιάστατου Πίνακα και των θέσεων τους στον πίνακα

Για να υπολογίσουμε το μέγιστο και το ελάχιστο στοιχείο ενός δισδιάστατου πίνακα καθώς και τις θέσεις τους στον πίνακα **με**

συνάρτηση η οποία τυπώνει όλα τα αποτελέσματα γράφουμε το ακόλουθο τμήμα κώδικα (στο πρόγραμμα αυτό τυπώνουμε όλες τις θέσεις του δισδιάστατου πίνακα στις οποίες συναντούμε το μέγιστο και το ελάχιστο στοιχείο του πίνακα):

```
static void max_min (int x[][])
{
    int i, j, max, maxp, min, minp, maxpi, minpi, maxpj, minpj;

    max=min=x[0][0];
    maxpi=minpi=maxpj=minpj=0;

    for (i=1;i<3;i++)
        for (j=0;j<3;j++)
        {
            if (x[i][j]>max)
            {
                max=x[i][j];
                maxpi=i;
                maxpj=j;
            }

            if (x[i][j]<min)
            {
                min=x[i][j];
                minpi=i;
                minpj=j;
            }
        }
```

```
        System.out.println ("Μέγιστο = "+max);
        for (i=1;i<3;i++)
            for (j=0;j<3;j++)
                if (x[i][j]==max)
                    System.out.println          ("θέση"+(i+1)+
","+(j+1));

        System.out.println("Ελάχιστο= "+min);
        for (i=1;i<3;i++)
            for (j=0;j<3;j++)
                if (x[i][j]==min)
                    System.out.println          ("θέση"+(i+1)+
","+(j+1));
}
```

Για να υπολογίσουμε το μέγιστο και το ελάχιστο στοιχείο ενός δισδιάστατου πίνακα καθώς και τις θέσεις τους στον πίνακα **με συνάρτηση η οποία επιστρέφει όλα τα αποτελέσματα** γράφουμε το ακόλουθο τμήμα κώδικα:

```
static String max_min (int x[][])
{
    int i, j, max, maxp, min, minp, maxpi, minpi, maxpj, minpj;
    String res="";

    max=min=x[0][0];
    maxpi=minpi=maxpj=minpj=0;
```

```
for (i=1;i<3;i++)
        for (j=0;j<3;j++)
        {
            if (x[i][j]>max)
            {
                max=x[i][j];
                maxpi=i;
                maxpj=j;
            }

            if (x[i][j]<min)
            {
                min=x[i][j];
                minpi=i;
                minpj=j;
            }
        }

        res+="Μέγιστο = "+max;
        for (i=1;i<3;i++)
             for (j=0;j<3;j++)
                  if (x[i][j]==max)
                       res+="θέση"+(i+1)+ ","+(j+1));

        res+="\nΕλάχιστο= "+min);
        for (i=1;i<3;i++)
             for (j=0;j<3;j++)
                  if (x[i][j]==min)
                       res+="θέση"+(i+1)+ ","+(j+1));
```

```
        return res;
}
```

Παρατήρηση

Το χαρακτηριστικό του κώδικα που αναφέραμε είναι ότι συνενώνουμε όλα τα αποτελέσματα στη μεταβλητή res τύπου String και επιστρέφουμε ως αποτέλεσμα της συνάρτησης τη μεταβλητή αυτή. Ο τελεστής + όταν χρησιμοποιείται με αλφαριθμητικά στη Java κάνει συνένωση (concatenation). Ο λόγος που γίνεται αυτό είναι ότι στη Java όπως και στη C++ όλες οι συναρτήσεις μπορούν να επιστρέφουν το πολύ ένα αποτέλεσμα στη συνάρτηση από την οποία καλούνται (είτε αυτή είναι το main είτε οποιαδήποτε άλλη συνάρτηση), οπότε συνενώνοντας όλα τα αποτελέσματα σε μια μεταβλητή τύπου String τότε επιστρέφουμε πολλά αποτελέσματα από τη συνάρτηση αλλά όλα αυτά είναι "τοποθετημένα" σε ένα αλφαριθμητικό (συμβολοσειρά).

18.2.5.7 Πλήθος Άρτιων και Περιττών στοιχείων Δισδιάστατου Πίνακα

Ο υπολογισμός του πλήθους άρτιων και περιττών στοιχείων ενός δισδιάστατου πίνακα υλοποιείται με το ακόλουθο τμήμα κώδικα:

```
static void artioi_perittoi (int x[][])
{
```

```
    int i, j, a=0,p=0;

    for (i=0;i<3;i++)
        for (j=0;j<3;j++)
            if (x[i][j]%2==0)
                a++;
            else
                p++;

    System.out.println("Πλήθος Άρτιων= "+a+" Πλήθος Περιττών ="+p);
}
```

18.2.5.8 Πλήθος Άρτιων και Περιττών στοιχείων σε κάθε γραμμή ενός Δισδιάστατου Πίνακα

Ο υπολογισμός του πλήθους άρτιων και περιττών στοιχείων σε κάθε γραμμή ενός δισδιάστατου πίνακα υλοποιείται με το ακόλουθο τμήμα κώδικα:

```
static void artioi_perittoi_grammon(int x[][])
{
    int i, j, a, p;

    for (i=0; i<3; i++)
    {
        a=0;
        p=0;
```

```
        for (j=0; j<3; j++)
            if (x[i][j]%2==0)
              a++;
            else
              p++;

        System.out.println(" Πλήθος Άρτιων "+(i+1)+" γραμμής = "+a+" Πλήθος περιττών "+(i+1)+" γραμμής = "+p);
    }
}
```

Παρατηρούμε όπως ακριβώς και στον αντίστοιχο αλγόριθμο της C++ ότι οι μετρητές άρτιων και περιττών μηδενίζονται σε κάθε γραμμή.

18.2.5.9 Πλήθος Άρτιων και Περιττών στοιχείων σε κάθε στήλη ενός Δισδιάστατου Πίνακα

Ο υπολογισμός του πλήθους άρτιων και περιττών στοιχείων σε κάθε στήλη ενός δισδιάστατου πίνακα υλοποιείται με το ακόλουθο τμήμα κώδικα:

```
static void artioi_perittoi_stilon(int x[][])
{
    int i, j, a, p;

    for (j=0; j<3; j++)
    {
        a=0;
```

```
        p=0;

        for (i=0; i<3; i++)
            if (x[i][j]%2==0)
              a++;
            else
              p++;

        System.out.println("Πλήθος Άρτιων "+(j+1)+" στήλης = "+a+" Πλήθος περιττών "+(j+1)+" στήλης = "+p);
    }
}
```

Παρατηρούμε όπως ακριβώς και στον αντίστοιχο αλγόριθμο της C++ ότι οι μετρητές άρτιων και περιττών μηδενίζονται σε κάθε στήλη.

18.2.5.10 Σειριακή Αναζήτηση Στοιχείου σε Δισδιάστατο Πίνακα

Η ακόλουθη συνάρτηση διαβάζει ένα στοιχείο αναζήτησης από το πληκτρολόγιο και τυπώνει όλες τις θέσεις στις οποίες αυτό εμφανίζεται καθώς και το πλήθος εμφανίσεων του στον πίνακα. Τα αποτελέσματα τυπώνονται μέσα στη συνάρτηση.

```
static void search(int x[][])
{
    int i, j, y, t=0;
```

```
    System.out.println("Δώστε στοιχείο αναζήτησης ");
    y=readint();

    for (i=0;i<3;i++)
        for (j=0;j<3;j++)
            if (y==x[i][j])
            {
                System.out.println("Θέση"+(i+1)+","+(j+1));
                t++;
            }

    if (t==0)
        System.out.println("Το στοιχείο δεν υπάρχει");
    else
        System.out.println("Το στοιχείο βρέθηκε "+t+" φορές ");
}
```

Μια παραλλαγή της προηγούμενης αναζήτησης είναι η συνάρτηση να επιστρέφει τα αποτελέσματα της (δηλαδή τις θέσεις στις οποίες το στοιχείο υπάρχει στον πίνακα καθώς και το πλήθος των εμφανίσεων του στον πίνακα) συνενωμένα σε ένα αλφαριθμητικό στο main και αυτό να τυπώνεται στο main.

Αυτή η παραλλαγή υλοποιείται με τον ακόλουθο κώδικα:

```
static String search(int x[][])
{
    int i, j, y, t=0;
    String res="";
```

```
    System.out.println("Δώστε στοιχείο αναζήτησης ");
    y=readint();

    for (i=0;i<3;i++)
        for (j=0;j<3;j++)
            if (y==x[i][j])
            {
                res+="Θέση"+(i+1)+","+(j+1);
                t++;
            }

    if (t==0)
        res+="Το στοιχείο δεν υπάρχει";
    else
        res+="Το στοιχείο βρέθηκε "+t+" φορές ";

    return res;
}
```

18.2.5.11 Άθροισμα Διαγωνίων Δισδιάστατου Πίνακα

Με την προϋπόθεση ότι ο πίνακας στον οποίο αναφερόμαστε είναι τετραγωνικός μπορούμε να υπολογίσουμε το άθροισμα της κύριας και της δευτερεύουσας διαγωνίου του με δύο τρόπους. Ο $1^{ος}$ τρόπος είναι με την ακόλουθη συνάρτηση στην οποία εκτυπώνονται και τα δύο αθροίσματα:

```
static void athroisma_diagonion (int x[][])
{
    int i, j, sum1=0, sum2=0;

    for (i=0; i<3; i++)
        for (j=0; j<3; j++)
        {
            if (i==j)
                sum1+=x[i][j];

            if (i+j==3-1)
                sum2+=x[i][j];
        }
    System.out.println ("Άθροισμα Κύριας Διαγωνίου = "+sum1);
    System.out.println ("Άθροισμα Δευτερεύουσας Διαγωνίου = "+sum2);
}
```

Ο 2ος τρόπος είναι με την ακόλουθη συνάρτηση στην οποία επιστρέφονται τα δύο αθροίσματα στο main συνενωμένα σε μια συμβολοσειρά (αλφαριθμητικό):

```
static String athroisma_diagonion (int x[][])
{
    int i, j, sum1=0, sum2=0;
    String res="";

    for (i=0; i<3; i++)
```

```
        for (j=0; j<3; j++)
        {
            if (i == j)
                sum1+=x[i][j];

            if (i+j == 3-1)
                sum2+=x[i][j];
        }
    res+="Άθροισμα Κύριας Διαγωνίου = "+sum1;
    res+="Άθροισμα Δευτερεύουσας Διαγωνίου = "+sum2;

    return res;
}
```

18.2.5.12 Εναλλαγή Γραμμών - Στηλών Δισδιάστατου Πίνακα

Οι ακόλουθοι αλγόριθμοι παρουσιάζουν εναλλαγές γραμμών και στηλών σε ένα δισδιάστατο πίνακα.

Α περίπτωση: Εναλλαγή 1ης-Τελευταίας στήλης

```
static void enallagi_protis_kai_teleytaias_stilis (int x[][])
{
    int i, temp;

    for (i=0;i<3;i++)
    {
        temp=x[i][0];
        x[i][0]=x[i][3-1];
```

```
            x[i][3-1]=temp;
        }
}
```

Β περίπτωση: Εναλλαγή 1ης-Τελευταίας σειράς (γραμμής)

```
static void enallagi_protis_kai_teleytaias_grammis (int x[][])
{
    int j, temp;

    for (j=0;j<3;j++)
    {
        temp=x[0][j];
        x[0][j]=x[3-1][j];
        x[3-1][j]=temp;
    }
}
```

19 Επίλογος – Συμπεράσματα

Οι αλγόριθμοι πινάκων που παρουσιάσαμε στα προηγούμενα κεφάλαια για τις γλώσσες προγραμματισμού C και Java παρουσιάζουν σε πολύ μεγάλο βαθμό αρκετές ομοιότητες καθώς πολλές εντολές των δύο αυτών γλωσσών έχουν τον ίδιο ακριβώς τρόπο σύνταξης και λειτουργίας (τουλάχιστον μέχρι τους πίνακες στους οποίους επικεντρώθηκε αυτή η πτυχιακή εργασία).

Στην παρουσίαση των αλγορίθμων (ειδικά στη Java) χρησιμοποιήσαμε τη δομή των συναρτήσεων προκειμένου να τους παρουσιάσουμε προσφέροντας έτσι στους χρήστες που θα θελήσουν να τους χρησιμοποιήσουν τη δυνατότητα να τους εκτελούν πιο γρήγορα και πιο αποτελεσματικά καλώντας αυτές τις συναρτήσεις οπουδήποτε στον κώδικα και όσες φορές χρειαστεί χωρίς να χρειαστεί να επαναλαμβάνουν ή να πληκτρολογούν τον ίδιο κώδικα.

Επίσης μια πιθανή μελλοντική επέκταση αυτής της πτυχιακής θα ήταν η τοποθέτηση των αλγορίθμων της C++ σε αρχεία επικεφαλίδας (header files) και αυτών της Java σε πακέτα ώστε να δημιουργηθούν βιβλιοθήκες με έτοιμο κώδικα που θα βοηθήσουν πολύ ανθρώπους που θα θέλουν να έχουν έτοιμα εργαλεία για την κατασκευή πιο σύνθετων προγραμμάτων.

Βιβλιογραφία

1. Προγραμματισμός με C++ Κύριος Συγγραφέας: Hubbard, John R. Εκδοτικός Οίκος: Κλειδάριθμος ISBN: 960-461-127-5

2. Αλγόριθμοι σε C++ Συγγραφέας: Sedgewick, Robert Εκδότης: Κλειδάριθμος Έτος Έκδοσης: 2006

3. C++ Συγγραφέας: Deitel, Harvey M., Deitel, Paul J. Εκδότης: Γκιούρδας Μ. Έτος Έκδοσης: 2003

4. Πλήρες εγχειρίδιο της Java 2 Συγγραφέας: Cadenhead, Rogers, Lemay, Laura Εκδότης: Γκιούρδας Μ. Έτος Έκδοσης: 2003

5. Εισαγωγή στη Java Συγγραφέας: Λιακέας, Γιώργος Εκδότης: Κλειδάριθμος Έτος Έκδοσης: 2001

6. Οδηγός Της Java 2 Συγγραφέας: Schildt Herbert Εκδότης: Γκιούρδας Μ. Έτος Έκδοσης: 2007