



ΕΘΝΙΚΟ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΦΥΣΙΚΗΣ

Οδηγός Εργαστηριών LINUX

Υπολογιστές Ι

Εργαστήριο Υπολογιστών
Χειμερινό Εξάμηνο
Ακαδημαϊκό Έτος 2017--2018

Επιμέλεια - Συγγραφή Εργαστηριακού Οδηγού: Εμ. Τσίλης, Επίκ. Καθηγητης ΕΚΠΑ
(mtsilis@phys.uoa.gr)

A. Μουστάκας, Αναπλ. Καθηγητης ΕΚΠΑ
(arism@phys.uoa.gr)

E. Νισταζάκης, Αναπλ. Καθηγητης ΕΚΠΑ
(enistaz@phys.uoa.gr)

A. Καραδημητράκης, Υποψ. Διδάκτορας ΕΚΠΑ
(apokaradim@phys.uoa.gr)

Σ. Ευαγγελάτος, Υποψ. Διδάκτορας ΕΚΠΑ
(s.evaggelatos@di.uoa.uoa.gr)

Περιεχόμενα

I	Εισαγωγή στο LINUX	5
II	Εργαστηριακές Ασκήσεις	11
1	Εισαγωγική Άσκηση	12
1.1	Αθροίσματα	13
1.2	Υπολογισμός μέσης τιμής	14
2	Άσκηση (εφαρμογές)	15
2.1	Ανταλλαγή Θέσεως Τιμών Δύο Μεταβλητών	15
2.2	Υπολογισμός απόλυτης τιμής	15
2.3	Υπολογισμός Εμβαδού Ισόπλευρου Τριγώνου	16
3	Άσκηση (If-Else)	17
3.1	Εφαρμογή: Επίλυση Τριωνύμου	17
3.2	Άσκηση: Καθορισμός Δίσεκτου Έτους	19
4	Άσκηση (απευθείας υπολογισμός τιμών συναρτήσεων και αναδρομική διαδικασία)	20
4.1	Υπολογισμός αθροίσματος $1 + \dots + n$	20
4.2	Υπολογισμός τόκων κεφαλαίου	20
4.3	Υπολογισμός σειράς Fibonacci	21
5	Άσκηση (αριθμητική ολοκλήρωση)	22
5.1	Εφαρμογή	23
5.2	Άσκηση: Υπολογισμός με τον κανόνα του παραλληλογράμμου	23
6	Άσκηση (υπολογισμός αθροίσματος - γινομένου)	25
6.1	Εφαρμογή	25
6.2	Χρήση των εντολών <code>do...while</code>	26
6.3	Άσκηση: Πολλαπλασιασμός με μεγάλο αριθμό όρων	26
6.4	Άσκηση: Επιλογή με την εντολή <code>switch</code>	26
7	Άσκηση (πράξεις με πίνακες)	27
7.1	Εφαρμογή	27
7.2	Άσκηση: Υπολογισμός αθροίσματος πινάκων	29
8	Άσκηση (αποθήκευση και ανάκτηση δεδομένων από το δίσκο)	30
8.1	Εφαρμογή	30
8.2	Άσκηση	31
8.3	Ανάγνωση αρχείου	31
9	Συναρτήσεις, καθολικές και τοπικές μεταβλητές	33
9.1	Εφαρμογή (Πολλαπλασιασμός μιγαδικών αριθμών)	34
9.2	Εφαρμογή (Παραγοντικό)	35

III	Παράρτημα	36
9.3	Τα πιο συχνά λάθη στη C	37
9.3.1	Μεταβλητές που δεν έχουν δηλωθεί - Undeclared Variables	37
9.3.2	Υπερβολική χρήση του στοιχείου ";" - Semicolon	38
9.3.3	Διαίρεση ακέραιου	38
9.3.4	Χρήση απλού "=" αντί διπλού "==" για τον έλεγχο ισότητας.	39
9.3.5	Προσπάθεια να μεταγλωττίσουμε κώδικα που δεν έχει αποθηκευτεί.	40

Κανονισμός Λειτουργίας Εργαστηρίου

Κάθε εργαστηριακή άσκηση διαρκεί δύο ώρες και ο κάθε φοιτητής εργάζεται σε ένα υπολογιστή. **Η προσέλευση στο εργαστήριο γίνεται ακριβώς την αναγραφόμενη ώρα και όχι αργότερα.** Σε περίπτωση καθυστέρησης προσέλευσης μεγαλύτερης των 10 λεπτών, ο φοιτητής δεν μπορεί να παρακολουθήσει το εργαστήριο τη συγκεκριμένη μέρα. Ερχόμενος στο εργαστήριο, κάθε φοιτητής θα πρέπει να έχει προετοιμαστεί διαβάζοντας προσεκτικά τη θεωρία που περιλαμβάνει το φυλλάδιο με τις ασκήσεις και τη θεωρία του εργαστηρίου. Θα πρέπει να έχει κατανοήσει τα προγράμματα που παρουσιάζονται, να έχει προετοιμάσει τα προγράμματα που ζητούνται και **να έχει, οπωσδήποτε, εντοπίσει τα εσκεμμένα λογικά και προγραμματιστικά σφάλματα στα προγράμματα που περιλαμβάνονται στο φυλλάδιο του “Εργαστηρίου Υπολογιστών Ι”.**

Οι διάρκειες του Εργαστηρίου είναι εννέα (9) εβδομάδες. Οι φοιτητές, κατά τη διάρκεια των εβδομάδων αυτών είναι υποχρεωμένοι να υλοποιήσουν και τις εννέα (9) ασκήσεις του φυλλαδίου του “Εργαστηρίου Υπολογιστών”. Μετά την υλοποίηση της κάθε άσκησης, θα πρέπει οπωσδήποτε να ειδοποιούν κάποιον από τους επιβλέποντες να επιβεβαιώσει την πραγματοποίηση της άσκησης και μετά να προχωρούν σε κάθε επόμενη άσκηση. Αν κάποιος φοιτητής δεν έχει πιστοποιηθεί από τους διδάσκοντες ότι πραγματοποίησε και τις εννέα ασκήσεις, κατά τη διάρκεια του εξαμήνου, τότε ο φοιτητής αυτός, **δεν μπορεί να λάβει μέρος στις εξετάσεις του εργαστηρίου.**

Κατά τη διάρκεια ενός δίωρου τμήματος “Εργαστηρίου Υπολογιστών Ι” ένας φοιτητής θα μπορεί να υλοποιήσει και περισσότερες από μία ασκήσεις, αν το επιθυμεί και είναι προετοιμασμένος για κάτι τέτοιο. Αντίστοιχα, ένας φοιτητής ο οποίος έχει κάνει κάποια απουσία κατά τη διάρκεια του εξαμήνου, θα πρέπει τις επόμενες εβδομάδες να αναπληρώσει την άσκηση αυτή και να πραγματοποιήσει και όλες τις υπόλοιπες μέχρι την ένατη άσκηση. Κάποιος φοιτητής που έχει τελειώσει και τις εννέα ασκήσεις πριν την ένατη εβδομάδα, μπορεί να μην προσέλθει στα τελευταία εργαστήρια. **Συμπληρωματικό εργαστήριο, δεν θα πραγματοποιηθεί δεδομένου ότι ο κάθε φοιτητής μπορεί να υλοποιήσει και περισσότερες από μία ασκήσεις κατά τη διάρκεια του κάθε δίωρου. Επομένως θα μπορεί να αναπληρώσει κάποια, πιθανή απουσία του.**

Στο τέλος του χειμερινού εξαμήνου, οι φοιτητές θα εξεταστούν στο εργαστήριο και ο βαθμός που θα λάβουν θα αποτελεί το 20% του συνολικού τους βαθμού στο μάθημα “Υπολογιστές Ι”. Το υπόλοιπο 80% θα προκύπτει από την τελική εξέταση του μαθήματος στην εξεταστική περίοδο. Ο βαθμός που θα λάβουν στο εργαστήριο, δεν μπορεί να αλλάξει κατά τη διάρκεια του ακαδημαϊκού έτους και όποιος φοιτητής επιθυμεί να βελτιώσει το βαθμό του στο εργαστήριο θα πρέπει να εξεταστεί και πάλι στο εργαστήριο, στο τέλος του χειμερινού εξαμήνου της επόμενης ακαδημαϊκής χρονιάς, αφού πρώτα το έχει δηλώσει στους διδάσκοντες σε μέρες και ώρες που θα ανακοινωθούν στην ιστοσελίδα του μαθήματος στο **eclass**. Ο βαθμός που θα έχει λάβει ο κάθε φοιτητής στο εργαστήριο, θα συνυπολογιστεί με το βαθμό του μαθήματος ανεξάρτητα από το πότε (σε ποια εξεταστική περίοδο) θα περάσει το μάθημα, εντός τριών (3) ετών. Μετά την πάροδο των τριών ετών, θα πρέπει υποχρεωτικά να επαναλάβει τις εξετάσεις του εργαστηρίου.

Επίσης, είναι εξαιρετικά σημαντικό οι φοιτητές να εγγραφούν στην ιστοσελίδα του μαθήματος και του εργαστηρίου στο **eclass**, ώστε να ενημερώνονται άμεσα και αντικειμενικά, από τους ίδιους τους διδάσκοντες για οποιοδήποτε θέμα αφορά στο μάθημα ή/και στο εργαστήριο. Πρέπει να τονιστεί εδώ ότι η ιστοσελίδα του μαθήματος και του εργαστηρίου στο **eclass** καθώς και οι ανακοινώσεις του Εργαστηρίου Υπολογιστών και της Γραμματείας του Τομέα Ηλεκτρονικής, Υπολογιστών, Τηλεπικοινωνιών και Αυτοματισμού, αποτελούν τη μοναδική έγκυρη πηγή ενημέρωσης σχετικά με θέματα του εργαστηρίου/μαθήματος.

Μέρος I

Εισαγωγή στο LINUX

Διαχείριση Αρχείων στο Linux

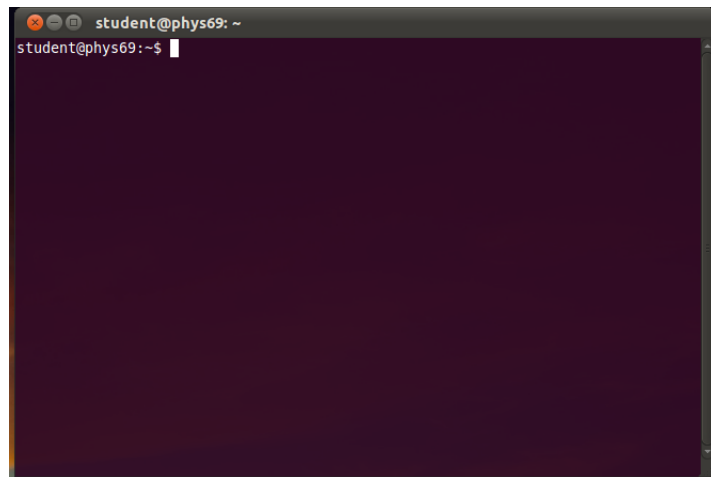
Όλες οι εντολές που θα περιγράψουμε σε αυτή την ενότητα εκτελούνται στο τερματικό του Linux. Για να το ανοίξετε, ακολουθήστε τα εξής βήματα:

Applications → Accessories → Terminal

ή για κάποιους υπολογιστές που έχουν εγκατεστημένο το KDE περιβάλλον

Applications → Accessories → Konsole

Μόλις ολοκληρώσετε τη παραπάνω διαδικασία θα δείτε να ανοίγει ένα καινούργιο παράθυρο όπου θα πληκτρολογείτε τις εντολές που αφορούν τη μεταγλώτιση του κάθε προγράμματος και κάποιες άλλες που σχετίζονται με τη διαχείριση αρχείων και φακέλων.



Σχήμα 1: Το τερματικό στο Linux

☒ Το τερματικό και ο editor είναι δύο τελείως διαφορετικά προγράμματα. Με άλλα λόγια, η γραμμή εντολών του τερματικού μπορεί να "δείχνει" σε άλλο φάκελο από τον οποίο εργαζόμαστε με κάποιον editor.

Η γραμμή Εντολών

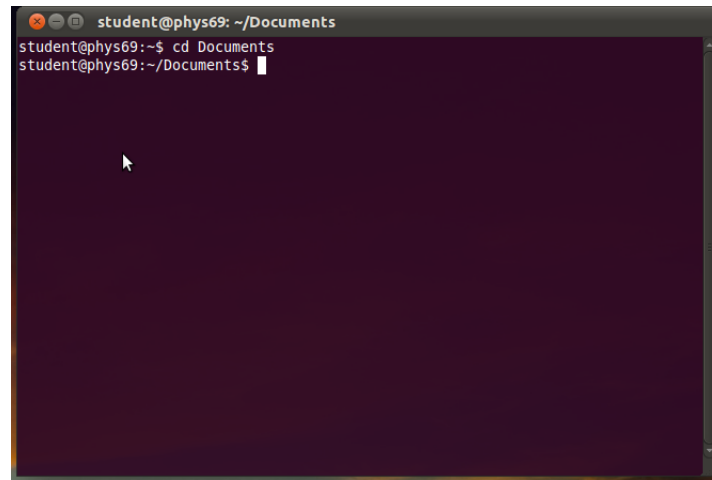
Η γραμμή εντολών στο τερματικό περιέχει κάποιες χρήσιμες πληροφορίες που θα αναλύσουμε στη συνέχεια και είναι της μορφής:

```
student@phys69:~/2017000111$
```

Υπάρχουν τρία μέρη σε αυτή τη γραμμή που χωρίζονται από τα @, : και \$. Ας τα δούμε πιο λεπτομερώς.

1. `student` → δείχνει το χρήστη που είναι συνδεδεμένος εκείνη τη χρονική στιγμή. Στη δική μας περίπτωση είναι ο χρήστης με το όνομα `student`.
2. `phys69` → δείχνει το όνομα του υπολογιστή που δουλεύει ο χρήστης. Στη δική μας περίπτωση είναι ο υπολογιστής με όνομα `phys69`.
3. `~/2017000111` → δείχνει τον τρέχοντα φάκελο εργασίας. Ο δρομέας εισαγωγής είναι το σύμβολο `$` το οποίο αναμένει την εισαγωγή εντολών από το χρήστη.

☑ Το σύμβολο ~ είναι η συντόμευση για τον αρχικό φάκελο του χρήστη, για παράδειγμα, ο φάκελος ~/2017000111 είναι ο /home/2017000111. Προφανώς, όποτε αλλάζετε το φάκελο στον οποίο δουλεύετε θα αλλάζει και ο τρέχων φάκελος εργασίας πριν το δρομέα εισαγωγής.



Σχήμα 2: Το τερματικό στο Linux

Γρήγορη βοήθεια

Όποτε χρειάζεστε βοήθεια για κάποια εντολή στο Linux, μπορείτε να διαβάσετε τη σελίδα του εγχειριδίου, που περιέχει λεπτομερείς πληροφορίες σχετικά με την εντολή και τις επιλογές που υπάρχουν. Απλά γράψτε στη γραμμή εντολών

```
man <command name>
```

όπου <command name> η εντολή που θέλετε να χρησιμοποιήσετε. Όλες οι πληροφορίες που ζητάτε θα εμφανιστούν στην οθόνη του υπολογιστή. Χρησιμοποιήστε τα βελάκια του πληκτρολογίου για να μετακινηθείτε προς τα πάνω ή προς τα κάτω του κειμένου. Για να φύγετε από τη συγκεκριμένη σελίδα, απλά πατήστε το γράμμα q.

☑ Στο τερματικό μπορείτε να χρησιμοποιήσετε μόνο τα βελάκια. Θα δείτε ότι με το ποντίκι δεν μπορείτε να μετακινηθείτε σε άλλο σημείο της γραμμής εντολών.

Φάκελος εργασίας

Για να δείτε το τρέχοντα φάκελο εργασίας οποιαδήποτε στιγμή, μπορείτε να πληκτρολογήσετε την εντολή

```
pwd
```

που είναι τα αρχικά της πρότασης: **p**rint name of **w**orking **d**irectory. Για να δείτε όλα τα περιεχόμενα του φακέλου εργασίας μπορείτε να πληκτρολογήσετε την εντολή

```
ls
```

που είναι τα αρχικά της πρότασης: **l**ist **d**irectory **c**ontents. Στη περίπτωση που θέλετε να δείτε περισσότερες λεπτομέρειες πληκτρολογήστε την εντολή

```
ls -l
```

Παρατηρήστε ότι οι υπο-φάκελοι και τα αρχεία που βρίσκονται στο τρέχοντα φάκελο εργασίας φαίνονται με διαφορετικό χρώμα στο τερματικό. Όπως είναι αναμενόμενο, τα διάφορα αρχεία εμφανίζονται και με την επέκτάσή τους (π.χ. test.c).

Αλλαγή Φακέλου

Για να αλλάξετε το φάκελο στον οποίο θέλετε να εργαστείτε και να αποθηκεύσετε τα προγράμματά σας, πληκτρολογήστε

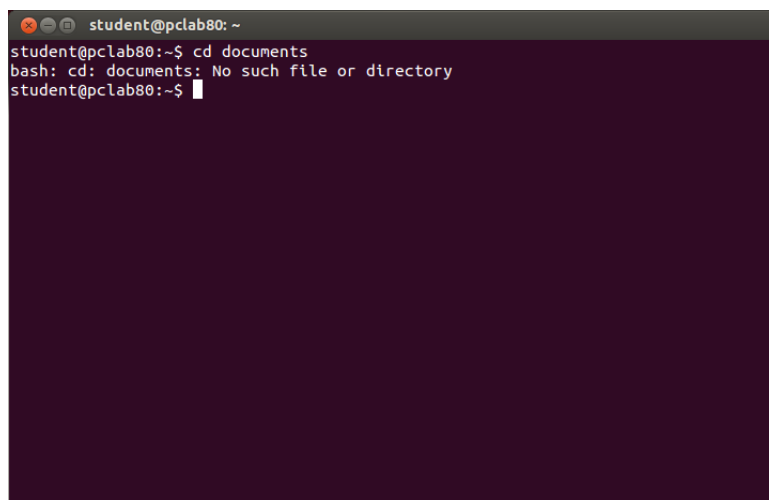
```
cd <new path on disk>
```

που είναι τα αρχικά της πρότασης: **change current directory**. Στη περίπτωση που θέλετε να μεταβείτε στο φάκελο Documents θα χρειαστεί να γράψετε στο τερματικό

```
cd Documents
```

☑ Θυμηθείτε ότι όλες οι εντολές είναι case sensitive. Στη περίπτωση δηλαδή που πληκτρολογήσετε `cd documents` θα σας εμφανιστεί ένα μήνυμα σφάλματος που φαίνεται στο Σχ. 3 καθώς ο φάκελος που υπάρχει μέσα στον υπολογιστή λέγεται Documents και όχι documents.

☑ Αν θέλετε να μεταβείτε σε ένα φάκελο που βρίσκεται μέσα στο φάκελο Documents π.χ. στο φάκελο 2017000111 τότε μπορείτε να πάτε σταδιακά, δηλαδή πληκτρολογώντας `cd Documents` και αμέσως μετά `cd 2017000111` ή μπορείτε απευθείας με ένα βήμα απλά πληκτρολογώντας `cd Documents/2017000111`



Σχήμα 3: Μήνυμα σφάλματος κατά την αλλαγή φακέλου.

Δημιουργία φακέλου

Ένας νέος φάκελος μπορεί να δημιουργηθεί αν πληκτρολογήσετε

```
mkdir <directory name>
```

που είναι τα αρχικά της πρότασης: **make directory**

Διαγραφή αρχείων

Η διαγραφή αρχείων γίνεται χρησιμοποιώντας την ακόλουθη εντολή

```
rm <file name>
```

που προέρχεται από την πρόταση: **remove files**

Επεξεργασία Προγραμμάτων

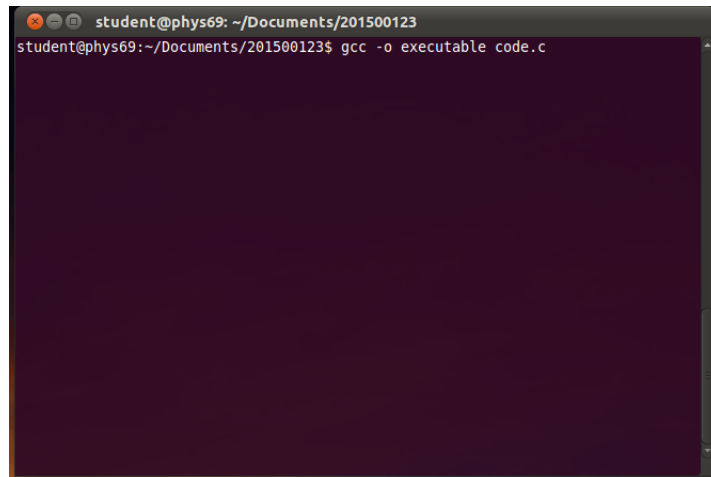
Το Linux προσφέρει αρκετά προγράμματα για επεξεργασία και άνοιγμα κειμένων. Τα περισσότερα από αυτά έχουν απλή διεπαφή με το χρήστη και συνήθως περιέχουν γραφικό περιβάλλον. Στους υπολογιστές του εργαστηρίου υπάρχουν εγκατεστημένα τα ακόλουθα:

- `gedit &`
- `kwrite &`

Μεταγλώττιση προγραμμάτων C

Για να μεταγλωττίσουμε τα προγράμματα που έχουμε γράψει στο εργαστήριο, από το τερματικό, απλά πληκτρολογούμε

```
gcc -o executable code.c
```

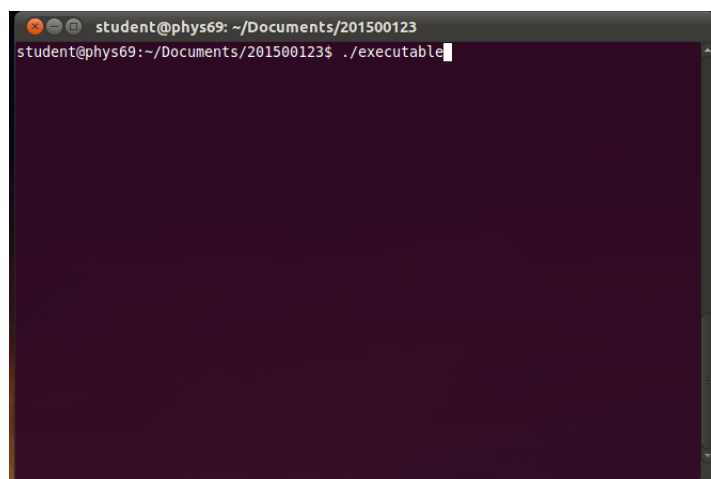


Σχήμα 4: Μεταγλώττιση προγράμματος C

όπου `executable` το όνομα που εκτελέσιμου αρχείου που θέλουμε να φτιάξουμε και `code.c` το όνομα του αρχείου που περιέχει των κώδικα που θέλουμε να μεταγλωττίσουμε. Μπορούμε να χρησιμοποιήσουμε ό,τι ονόματα θέλουμε.

ΠΡΟΣΟΧΗ: το αρχείο του κώδικα πρέπει, πάντα, να έχει κατάληξη `.c`

Μετά την εκτέλεση της εντολής, εμφανίζονται στο τερματικό τα σχόλια ή αλλιώς τα λάθη που τυχόν έχουμε κάνει. Για να είναι το πρόγραμμα έτοιμο προς εκτέλεση, θα πρέπει να μην μας εμφανίζονται άλλα σχόλια.



Σχήμα 5: Εκτέλεση προγράμματος C

Εκτέλεση των προγραμμάτων

Όταν ο compiler δε μας δώσει άλλα λάθη, είμαστε έτοιμοι να «τρέξουμε» το πρόγραμμά μας πληκτρολογώντας την εντολή:

```
./executable
```

Σύνοψη

Στο πρώτο εργαστήριο, ακολουθούμε τα παρακάτω βήματα (πληκτρολογούμε τις παρακάτω εντολές στο τερματικό)

1. `cd Documents`
2. `mkdir 2017000xxx`
3. `cd 2017000xxx`
4. `gedit &`

Μόλις έχουμε πληκτρολογήσει το πρόγραμμα μας στον editor, το μεταγλωττίζουμε, διορθώνουμε τα λάθη και "τρέχουμε" το εκτελέσιμο με τις ακόλουθες εντολές

5. `gcc -o executable code.c`
6. `./executable`

Στα υπόλοιπα εργαστήρια (φροντίζουμε να καθίσουμε στον ίδιο υπολογιστή) πληκτρολογούμε μόνο τις παρακάτω εντολές

1. `cd Documents/2017000xxx`
2. `gedit &`

Μέρος II

Εργαστηριακές Ασκήσεις

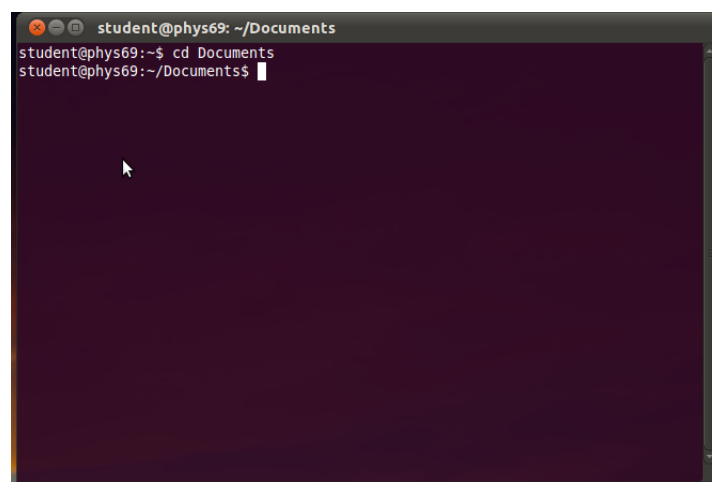
Άσκηση 1

Εισαγωγική Άσκηση

Διαδικασία πριν την Άσκηση

Είναι σημαντικό οι φοιτητές να έχουν διαβάσει την εισαγωγή του φυλλαδίου και τη θεωρία της κάθε άσκησης. Κάθε φοιτητής θα έχει συγκεκριμένο υπολογιστή στο εργαστήριο, στον οποίο θα εκτελεί και θα σώζει τις ασκήσεις. Για το λόγο αυτό, οι φοιτητές θα πρέπει να δημιουργήσουν Φάκελο με όνομα το δικό τους ΑΜ μέσα στον οποίο θα σώζονται οι εργασίες. Για τη δημιουργία του Φακέλου, εκτελούμε τις εντολές από το terminal:

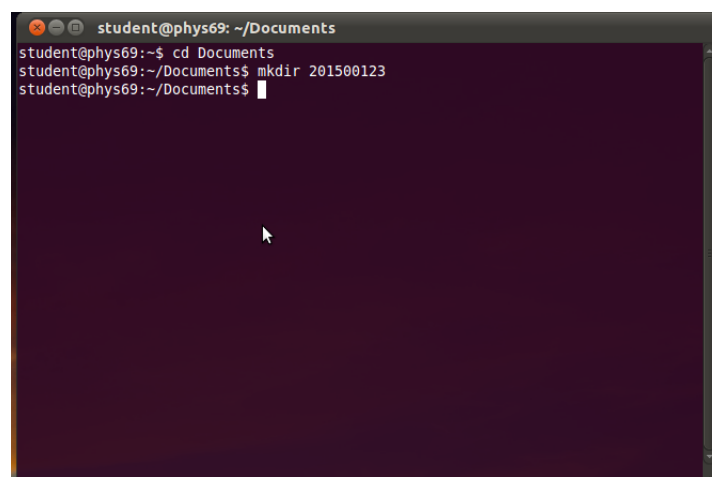
1. `cd Documents` που μας πηγαίνει στο directory Documents.



```
student@phys69: ~/Documents
student@phys69:~$ cd Documents
student@phys69:~/Documents$
```

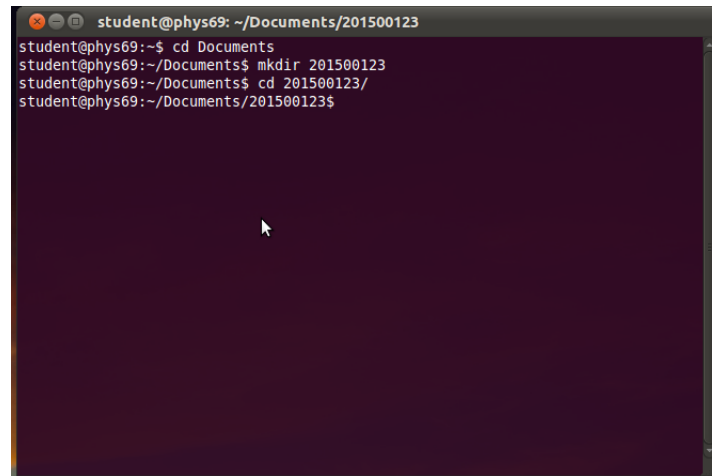
Σχήμα 1.1: Το τερματικό στο Linux

2. `mkdir 2017XXXXX` (το ΑΜ του κάθε φοιτητή)



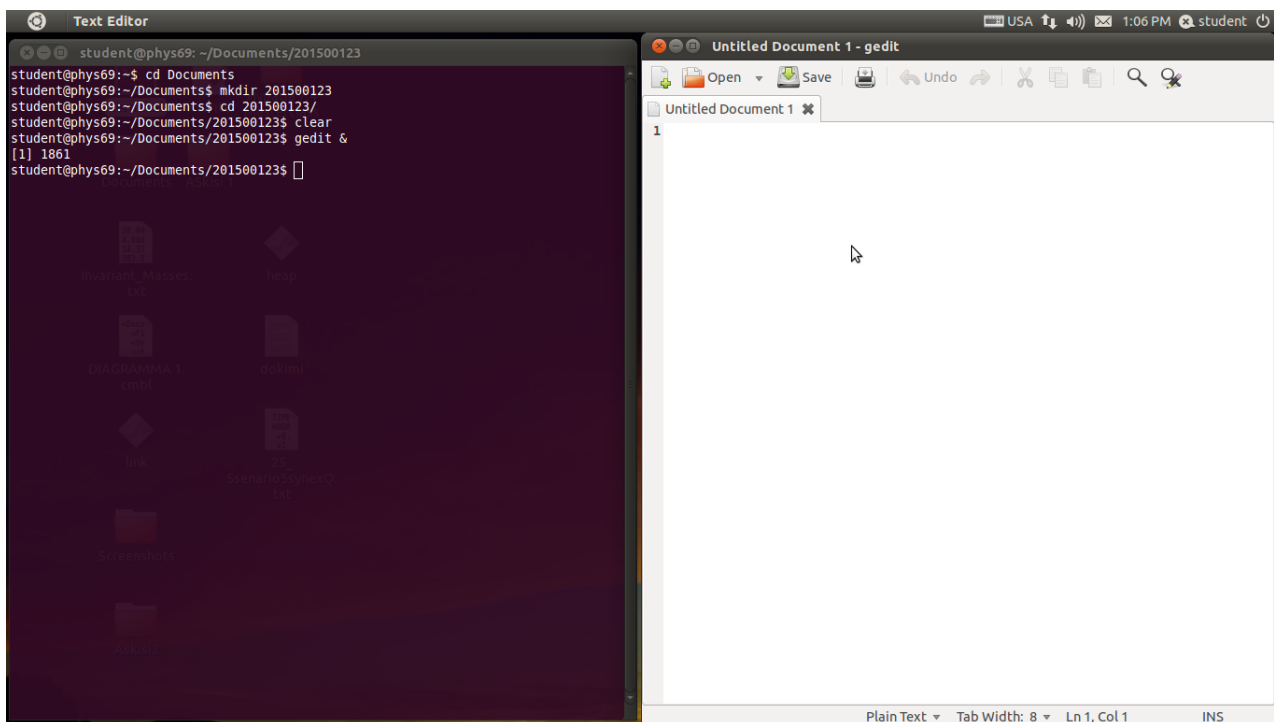
```
student@phys69: ~/Documents
student@phys69:~$ cd Documents
student@phys69:~/Documents$ mkdir 201500123
student@phys69:~/Documents$
```

3. `cd 2017XXXXXX`



```
student@phys69: ~/Documents/201500123
student@phys69:~$ cd Documents
student@phys69:~/Documents$ mkdir 201500123
student@phys69:~/Documents$ cd 201500123/
student@phys69:~/Documents/201500123$
```

Στη συνέχεια, για να εκκινήσει ο editor, εκτελούμε την εντολή `gedit` ή `kwrite` &. Το σύμβολο "&" χρησιμοποιείται για να μας επιτραπεί να εισάγουμε περισσότερες εντολές στο terminal.



1.1 Αθροίσματα

Παρακάτω δίνεται το πρώτο σας πρόγραμμα, το οποίο τυπώνει μηνύματα στην οθόνη και εκτελεί μία πρόσθεση με προσθετέους που εσείς εισάγετε. Επίσης τυπώνει και το αποτέλεσμα της πρόσθεσης.

```

#include <stdio.h>

int main()
{
//Define Variables
double a,b,c;

printf("\n Hello!");
printf("\n Give me a= ");
scanf("%lf",&a);
printf("\n Give me b= ");
scanf("%lf",&b);

c = a+b;

printf("\n The sum is c = %lf ", c);
printf("\n The sum with 7 total digits and
3 decimal digits is c = %7.3", c);
printf("\n The sum written scientifically is c = %e \n", c);

return 0;
}

```

1.2 Υπολογισμός μέσης τιμής

Με βάση τις γνώσεις από την παραπάνω άσκηση, μπορείτε τώρα να γράψετε ένα πρόγραμμα από την αρχή, το οποίο υπολογίζει τη μέση τιμή τεσσάρων αριθμών. Η ιδέα είναι η ίδια με την παραπάνω, μόνο που, αντί να εισάγετε δύο αριθμούς, a , b , πρέπει τώρα να εισάγετε τέσσερις αριθμούς, a , b , c , d και να διαιρέσετε το άθροισμα με το τέσσερα. Προσοχή, οι αριθμοί πρέπει να είναι πραγματικοί (και όχι ακέραιοι), και έτσι για να μην μπερδέψετε τον compiler, πρέπει να διαιρέσετε με το (4.0) ή το (4.), αντί απλά το (4). Επίσης, αρκεί να τυπώνεται ο τελικός αριθμός (η μέση τιμή δηλαδή) με έναν από τους τρεις τρόπους που χρησιμοποιήθηκαν στο παραπάνω πρόγραμμα.

Όταν τελειώσετε το πρόγραμμα, πρέπει να το σώσετε στον δίσκο (δηλ. να κάνετε SAVE), κατόπιν να το κάνετε COMPILE και μετά να βεβαιωθείτε ότι «τρέχει» ακολουθώντας τις παραπάνω οδηγίες. Στη συνέχεια πρέπει οπωσδήποτε να το δείξετε σε κάποιον από τους επιβλέποντες ο οποίος θα επιβεβαιώσει τη σωστή λειτουργία του.

Άσκηση 2

Άσκηση (εφαρμογές)

2.1 Ανταλλαγή Θέσεως Τιμών Δύο Μεταβλητών

Το παρακάτω πρόγραμμα είναι ένα βασικό πρόγραμμα στη λογική του προγραμματισμού: Διαβάζει δύο πραγματικούς αριθμούς και να τους τοποθετεί στις μεταβλητές x , y και μετά να τους ανταλλάσει θέσεις.

ΠΡΟΣΟΧΗ: ΣΤΟ ΠΡΟΓΡΑΜΜΑ ΥΠΑΡΧΟΥΝ ΛΟΓΙΚΑ ΚΑΙ ΣΥΝΤΑΚΤΙΚΑ ΛΑΘΗ ΚΑΘΩΣ ΚΑΙ ΠΑΡΑΛΕΙΨΕΙΣ ΠΟΥ ΠΡΕΠΕΙ ΝΑ ΕΝΤΟΠΙΣΕΤΕ ΓΙΑ ΝΑ ΤΡΕΧΕΙ ΚΑΝΟΝΙΚΑ

```
#include <stdio.h>
int main(void)
{
    //Define variables
    double x, y, prosorini;
    /*function body*/
    printf("\n Give values of x and y");
    scanf("%lf %lf ", &x, &y);
    printf("\nx = %f y = %f \n", x, y);
    prosorini = x;
    x = y;
    y = prosorini;
    printf("\n x = %f \t y = %f", y, x);
    return 0;
}
```

2.2 Υπολογισμός απόλυτης τιμής

Μία απλή εφαρμογή της εντολής `if` είναι να υπολογίστε την απόλυτη τιμή ενός πραγματικού αριθμού που δίνεται από το πληκτρολόγιο (χωρίς να χρησιμοποιείτε την εντολή `fabs(a)`). Η διαδικασία είναι η εξής: Πρώτα εισάγεται η τιμή του αριθμού από το πληκτρολόγιο. Κατόπιν ελέγχεται το πρόσημο του αριθμού, δηλαδή αν το $a < 0$. Αν το a είναι αρνητικό του αλλάζουμε πρόσημο με την εντολή " $a = -a$ ". Αν το a είναι θετικό δεν κάνουμε τίποτε. Κατόπιν τυπώνουμε τον αριθμό a , ο οποίος τώρα έχει σίγουρα θετικό πρόσημο.

ΠΡΟΣΟΧΗ: ΣΤΟ ΠΡΟΓΡΑΜΜΑ ΥΠΑΡΧΟΥΝ ΛΟΓΙΚΑ ΚΑΙ ΣΥΝΤΑΚΤΙΚΑ ΛΑΘΗ ΚΑΘΩΣ ΚΑΙ ΠΑΡΑΛΕΙΨΕΙΣ ΠΟΥ ΠΡΕΠΕΙ ΝΑ ΕΝΤΟΠΙΣΕΤΕ ΓΙΑ ΝΑ ΤΡΕΧΕΙ ΚΑΝΟΝΙΚΑ

```
#include <stdio.h>

int main(void)
{
    // Define variables

    double a;

    printf("\n Give value "a);
    scanf("%lf", &a); εισαγωγή           //{ στοιχείων}

    //
    // Main Program
    //

    return 0;
}
```

2.3 Υπολογισμός Εμβαδού Ισόπλευρου Τριγώνου

ΠΡΟΣΟΧΗ: ΣΤΟ ΠΡΟΓΡΑΜΜΑ ΥΠΑΡΧΟΥΝ ΛΟΓΙΚΑ ΚΑΙ ΣΥΝΤΑΚΤΙΚΑ ΛΑΘΗ ΚΑΘΩΣ ΚΑΙ ΠΑΡΑΛΕΙΨΕΙΣ ΠΟΥ ΠΡΕΠΕΙ ΝΑ ΕΝΤΟΠΙΣΕΤΕ ΓΙΑ ΝΑ ΤΡΕΧΕΙ ΚΑΝΟΝΙΚΑ

```
#include <stdio.h>
#include <math.h>

int main(void)
{

    double x, h;

    printf("\n Typose to mikos tis pleyras tou trigonou\n");
    scanf("%lf", x);

    h = sqrt(3)/2.0*x;
    E = 0.5*x*h;

    printf("\n To ypsos tou trigonou einai: %lf", h);
    printf("\n To embadon tou trigonou einai: %lf\n", E);

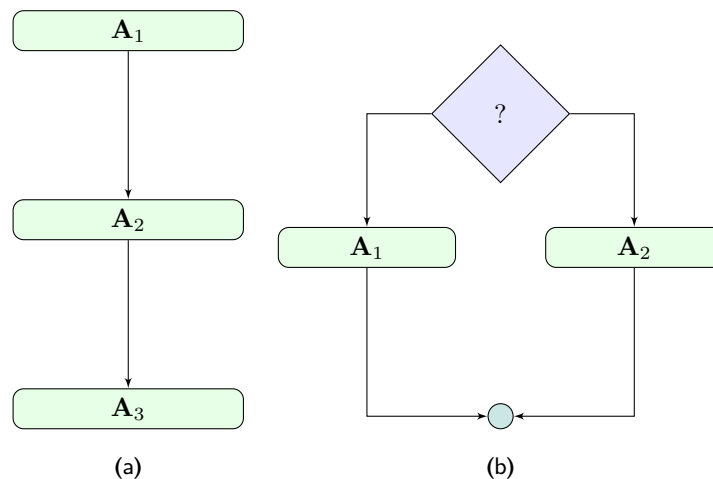
    return 0;
}
```


Άσκηση 3

Άσκηση (If-Else)

Λογικά διαγράμματα

Στα πλαίσια του δομημένου προγραμματισμού χρησιμοποιούνται λογικά διαγράμματα που παριστούν βασικές λογικές δομές. Στο πρόγραμμα χρησιμοποιούνται η σειριακή δομή (Σχ. 3.1a) και η δομή απλής επιλογής (If-else) (Σχ. 3.1b).



3.1 Εφαρμογή: Επίλυση Τριωνύμου

Όπως είναι γνωστό ένα τριώνυμο δευτέρου βαθμού $\rho(x)$ ορίζεται μονοσήμαντα από τους συντελεστές του α, β, γ . Δοθέντων των συντελεστών μπορούμε να αποφανθούμε κατά πόσο το τριώνυμο έχει πραγματικές ή μιγαδικές ρίζες. Το πρόγραμμα κάνει αυτή ακριβώς τη δουλειά: ζητάει τους συντελεστές και βγάζει αποτελέσματα. Ιδιαίτερη προσοχή χρειάζεται όταν εισάγεται τιμή για τον α : **δεν πρέπει να δοθεί τιμή μηδέν σε αυτό το συντελεστή.**

Παρακάτω δίνεται το πρόγραμμα, το οποίο δίνει τις ρίζες τριωνύμου που εσείς εισάγετε δίνοντας τους συντελεστές του.

ΠΡΟΣΟΧΗ: ΣΤΟ ΠΡΟΓΡΑΜΜΑ ΥΠΑΡΧΟΥΝ ΛΟΓΙΚΑ ΚΑΙ ΣΥΝΤΑΚΤΙΚΑ ΛΑΘΗ ΚΑΘΩΣ ΚΑΙ ΠΑΡΑΛΕΙΨΕΙΣ ΠΟΥ ΠΡΕΠΕΙ ΝΑ ΕΝΤΟΠΙΣΕΤΕ ΓΙΑ ΝΑ ΤΡΕΧΕΙ ΚΑΝΟΝΙΚΑ

```

#include <stdio.h>
#include <math.h>

int main(void)
{
// Define variables

double a, b, c, d, x1;

printf("\nGive value a (non-zero)");
scanf("%lf", &a);
printf("\nGive value "b");
scanf("%lf", b);
printf("\n Give value of "c");
scanf("%lf", c);

d=b*b-4*a*c;

if (d == 0.0)
{
    x1 = -0.5*b/a;
    printf("\nThe double root is x= %f, x1);
}
else
{
    if (d>0)
        printf("\nNo real "roots);
    else
    {
        x1=0.5*(-b + sqrt(d))/a;
        x2=0.5*(-b - sqrt(d))/a;
        print("\nThe two roots are x1= %f and x2=%f, x1, x2);
    }
}
return 0;
}

```

Το πρόγραμμα ζητάει τους a, b, c και υπολογίζει τη διακρίνουσα. Έπειτα χρησιμοποιεί τη δομή : If-else και αναλόγως τυπώνει στην οθόνη τα αποτελέσματα.

Η δομή If-else έχει την εξής σύνταξη:

```

If (condition)
    command list
else
    command list

```

Σε μια command list μπορεί να εμπεριέχεται μια δομή If -else. Τότε έχουμε φωλιασμένο (nested) if, όπως και στο πρόγραμμα μας.

ΣΗΜΑΝΤΙΚΟ: Όταν κάνουμε compile το παραπάνω πρόγραμμα, καθώς και κάθε πρόγραμμα στο οποίο συμπεριλαμβάνουμε το #include <math.h>, χρησιμοποιούμε την εντολή:

```
gcc -o onoma onomaexec.c -lm
```

Το έξτρα τελευταίο κομμάτι (το -lm) καλεί τη βιβλιοθήκη math.h

3.2 Άσκηση: Καθορισμός Δίσεκτου Έτους

Γράψτε ένα πρόγραμμα που καθορίζει αν ένα συγκεκριμένο έτος είναι δίσεκτο ή όχι.

ΠΡΟΣΟΧΗ: ΣΤΟ ΠΡΟΓΡΑΜΜΑ ΥΠΑΡΧΟΥΝ ΛΟΓΙΚΑ ΚΑΙ ΣΥΝΤΑΚΤΙΚΑ ΛΑΘΗ ΚΑΘΩΣ ΚΑΙ ΠΑΡΑΛΕΙΨΕΙΣ ΠΟΥ ΠΡΕΠΕΙ ΝΑ ΕΝΤΟΠΙΣΕΤΕ ΓΙΑ ΝΑ ΤΡΕΧΕΙ ΚΑΝΟΝΙΚΑ

```
#include <stdio.h>

int main()
{
    int year;

    printf("Enter a year to check if it is a leap year\n");
    scanf("%d", year);

    if ( year%400 == 0)
        printf("%d is a leap year.\n", year);
    else if ( year%100 == 0)
        printf("%d is not a leap year.\n", year);
    else if ( year%4 == 0 )
        printf("%d is a leap year.\n", year);
    else
        printf("%d is not a leap year.\n", year);
    return 0;
}
```

Άσκηση 4

Άσκηση (απευθείας υπολογισμός τιμών συναρτήσεων και αναδρομική διαδικασία)

4.1 Υπολογισμός αθροίσματος $1 + \dots + n$

ΠΡΟΣΟΧΗ: ΣΤΟ ΠΡΟΓΡΑΜΜΑ ΥΠΑΡΧΟΥΝ ΛΟΓΙΚΑ ΚΑΙ ΣΥΝΤΑΚΤΙΚΑ ΛΑΘΗ ΚΑΘΩΣ ΚΑΙ ΠΑΡΑΛΕΙΨΕΙΣ ΠΟΥ ΠΡΕΠΕΙ ΝΑ ΕΝΤΟΠΙΣΕΤΕ ΓΙΑ ΝΑ ΤΡΕΧΕΙ ΚΑΝΟΝΙΚΑ

```
#include <stdio.h>

int main(void)
{
    int n, i;
    printf("\n Dose tin timi tou n = ");
    scanf("%d", &n);

    S = 0;
    for (i=1; i<=n; ++i)
        S=S+i;

    printf("\n To athroisma S_n = %d \n", S);

    return 0;
}
```

4.2 Υπολογισμός τόκων κεφαλαίου

Στο παρακάτω πρόγραμμα μας δίνεται το αρχικό κεφάλαιο μίας κατάθεσης, καθώς και το επιτόκιο, και ο αριθμός ετών της κατάθεσης. Ζητούμενο είναι η αξία του κεφαλαίου σαν συνάρτηση του χρόνου.

ΠΡΟΣΟΧΗ: ΣΤΟ ΠΡΟΓΡΑΜΜΑ ΥΠΑΡΧΟΥΝ ΛΟΓΙΚΑ ΚΑΙ ΣΥΝΤΑΚΤΙΚΑ ΛΑΘΗ ΚΑΘΩΣ ΚΑΙ ΠΑΡΑΛΕΙΨΕΙΣ ΠΟΥ ΠΡΕΠΕΙ ΝΑ ΕΝΤΟΠΙΣΕΤΕ ΓΙΑ ΝΑ ΤΡΕΧΕΙ ΚΑΝΟΝΙΚΑ

```

#include <stdio.h>

int main(void)
{
    double epitokio, kefalaio, tokos, axia_kef;
    int etos, plithos_eton;
    printf("Doste times gia kefalaio, plithos eton, epitokio \n");
    scanf("%lf %d %lf", &kefalaio, &plithos_eton, &epitokio);
    axia_kef = kefalaio;
    printf("Etos          Tokos          Axia \n");
    for (etos = 1; etos <= plithos_eton; ++kefalaio)
    {
        tokos = axia_kef * tokos;
        axia_kef += tokos;
        printf("%4d %15f %15f\n", etos, tokos, axia_kef);
    }

    return 0;
}

```

4.3 Υπολογισμός σειράς Fibonacci

Το παρακάτω πρόγραμμα είναι ένα κλασικό πρόγραμμα υπολογισμού σειράς. Συγκεκριμένα ζητείται ο υπολογισμός των όρων της σειράς Fibonacci, η οποία δίνεται με τη σχέση

$$S_{n+1} = S_n + S_{n-1}$$

με αρχικές τιμές $S_0 = 0$ και $S_1 = 1$. Η σειρά Fibonacci εμφανίζεται συχνά στα μαθηματικά και στη φύση. Δείτε για παράδειγμα την ιστοσελίδα Wikipedia

ΠΡΟΣΟΧΗ: ΣΤΟ ΠΡΟΓΡΑΜΜΑ ΥΠΑΡΧΟΥΝ ΛΟΓΙΚΑ ΚΑΙ ΣΥΝΤΑΚΤΙΚΑ ΛΑΘΗ ΚΑΘΩΣ ΚΑΙ ΠΑΡΑΛΕΙΨΕΙΣ ΠΟΥ ΠΡΕΠΕΙ ΝΑ ΕΝΤΟΠΙΣΕΤΕ ΓΙΑ ΝΑ ΤΡΕΧΕΙ ΚΑΝΟΝΙΚΑ

```

#include<stdio.h>
int main(void)
{
    int i, n, an, an1, an2;
    printf("Doste to n>=2: n = ");
    scanf("%d", &n);
    /* Υπολογισμος του athroismatos */
    an = 1; /*arxiki timi */
    an1 = 1; /*arxiki timi */
    printf("\n a[0] =%d\n", an);
    printf("\n a[1] =%d\n", an1);
    for (i = 2; i<=n; ++i)
    {
        an2 = an1+ an;
        an = an1;
        an1 = an2;
        printf("\n a[%d] =%d\n", i, an2);
    }
    return 0;
}

```

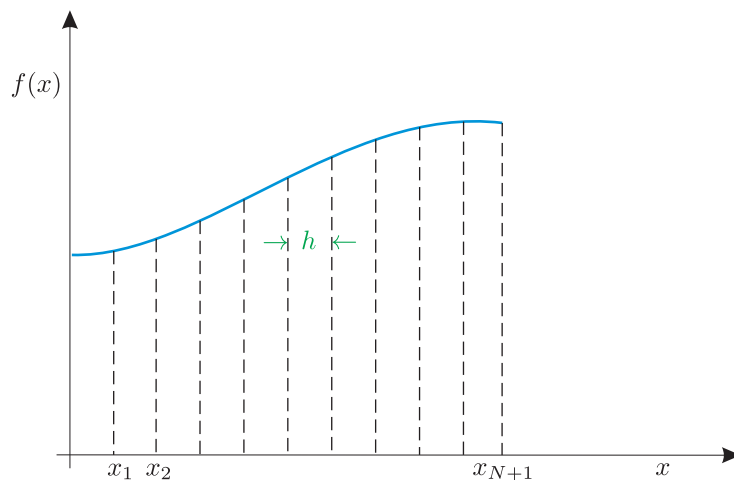
Άσκηση 5

Άσκηση (αριθμητική ολοκλήρωση)

Εισαγωγή

Σε αυτήν την άσκηση θα ασχοληθούμε με μία πολύ χρήσιμη εφαρμογή των υπολογιστών στη Φυσική, την αριθμητική ολοκλήρωση συναρτήσεων. Υπάρχουν πολλές μέθοδοι ολοκλήρωσης, με ευρύ φάσμα σε πολυπλοκότητα και ακρίβεια υπολογισμών. Εδώ, θα αναπτύξουμε την μέθοδο ολοκλήρωσης με τον κανόνα του τραπεζίου. Σύμφωνα με τη μέθοδο αυτή, όταν τα σημεία x_1, x_2 είναι πολύ κοντά μεταξύ τους (δηλαδή το $\Delta x = x_2 - x_1 = h \rightarrow 0$), τότε το ολοκλήρωμα μεταξύ των σημείων x_1 και x_2 μπορεί να θεωρηθεί ότι ισούται, με μικρό σφάλμα υπολογισμού, με το στοιχειώδες τραπέζιο που σχηματίζεται. Δηλαδή:

$$\int_{x_1}^{x_2} f(x) dx \approx \frac{1}{2} (f(x_1) + f(x_2)) (x_2 - x_1) = \frac{h}{2} (f(x_1) + f(x_2))$$



Σχήμα 5.1: Μέθοδος ολοκλήρωσης με τη τον κανόνα του τραπεζίου.

Έτσι, το ολοκλήρωμα της $f(x)$ από x_1 έως x_{N+1} μπορεί να υπολογιστεί από το άθροισμα των εμβαδών των στοιχειωδών τραπεζίων (βλ. Σχ. 5) που σχηματίζονται:

$$\begin{aligned} \int_{x_1}^{x_{N+1}} f(x) dx &\approx \frac{h}{2} (f_1 + f_2) + \frac{h}{2} (f_2 + f_3) + \dots + \frac{h}{2} (f_N + f_{N+1}) \\ &= h \left(\frac{f_1}{2} + f_2 + f_3 + \dots + \frac{f_{N+1}}{2} \right) \end{aligned}$$

Επομένως το ολοκλήρωμα αυτό θα μπορεί να υπολογιστεί ως:

$$\int_{x_1}^{x_{N+1}} f(x) dx \approx h (f_1 + f_2 + f_3 + \dots + f_{N+1}) - \frac{h}{2} (f_1 + f_{N+1})$$

όπου $f_1 = f(x_1)$, $f_2 = f(x_2)$ κλπ και τα $x_1, x_2, \dots, x_N, x_{N+1}$ απέχουν μεταξύ τους, δηλαδή

$$x_2 - x_1 = x_3 - x_2 = \dots = x_{N+1} - x_N = h = \frac{x_{N+1} - x_1}{N}$$

Για απλότητα ονομάζουμε τα πρώτα και τελευταία x_1 και x_{N+1} ως a και b . Δηλαδή:

$$x_1 = a \quad \text{και} \quad x_{N+1} = b$$

5.1 Εφαρμογή

Στην παρακάτω εφαρμογή θα υπολογίσουμε το ολοκλήρωμα της συνάρτησης $\cos(x)$ από a έως b και θα το συγκρίνουμε με το σωστό αποτέλεσμα $\int_a^b \cos(x) dx = \sin(b) - \sin(a)$ για διάφορες τιμές των a, b και του αριθμού των βημάτων N .

ΠΡΟΣΟΧΗ: ΣΤΟ ΠΡΟΓΡΑΜΜΑ ΥΠΑΡΧΟΥΝ ΛΟΓΙΚΑ ΚΑΙ ΣΥΝΤΑΚΤΙΚΑ ΛΑΘΗ ΚΑΘΩΣ ΚΑΙ ΠΑΡΑΛΕΙΨΕΙΣ ΠΟΥ ΠΡΕΠΕΙ ΝΑ ΕΝΤΟΠΙΣΕΤΕ ΓΙΑ ΝΑ ΤΡΕΧΕΙ ΚΑΝΟΝΙΚΑ

```
#include <stdio.h>
#include <math.h>

int main(void)
{
    double a, b, n, h, sum;
    int i;
    printf("To programma ayto tha oloklirosei to
cos(x) apo to a os to b se N vimata.\n");
    printf("Dose tin arxiki timi olokliromatos, a\n");
    scanf("%lf", &a);
    printf("Dose tin teliki timi olokliromatos, b\n");
    scanf("%lf", &b);
    printf("Dose ton synoliko arithmo vimaton N\n");
    scanf("%lf", &n);

    h = (b-a)/n;
    sum = 0.0;

    for (i=0; i<n+1, ++i)
        sum += cos(a+h*i)*h;

    sum -= 0.5*h*(cosa+cos(b));

    printf("arithmitiko apotelesma = %lf \n", sum);
    exact_res = sin(b)-sin(a);
    printf("akrives apotelesma = %lf \n", exact_res);
    diff = fabs(sum - exact_res);
    printf("diafora = %e \n", &diff );
}
```

5.2 Άσκηση: Υπολογισμός με τον κανόνα του παραλληλογράμμου

Ένας άλλος τρόπος αριθμητικού υπολογισμού του ολοκληρώματος είναι η μέθοδος του παραλληλογράμμου. Η μεθοδολογία αυτή είναι λιγότερο ακριβής από αυτή του τραπεζίου που αναλύθηκε παραπάνω, αλλά

απαιτεί μικρότερο αριθμό πράξεων, άρα είναι ταχύτερη και συνεπώς, πολλές φορές, πιο χρήσιμη. Σύμφωνα με τη μέθοδο αυτή, όπως φαίνεται στο παραπάνω σχήμα, χωρίζουμε το διάστημα που θέλουμε να ολοκληρώσουμε σε μικρά κομμάτια που απέχουν μεταξύ τους h . Θεωρούμε ότι, όταν τα σημεία π.χ. x_1, x_2 , είναι πολύ κοντά μεταξύ τους, τότε το μπορεί να προσομοιωθεί με ένα παραλληλόγραμμο με τη μία πλευρά του να είναι μήκους x_1 και την άλλη h . Οπότε το στοιχειώδες εμβαδό από την πρώτη διαμέριση θα είναι $hf(x_1)$, ενώ από κάθε επόμενη θα είναι $hf(x_i)$, με $i = 1, \dots, N$. Άρα, το συνολικό εμβαδό και το ολοκλήρωμα θα δίνεται από τη σχέση:

$$\int_{x_1}^{x_{N+1}} f(x)dx \approx h(f_1 + f_2 + f_3 + \dots + f_N) = h(f_1 + f_2 + f_3 + \dots + f_{N+1}) - hf_{N+1}$$

Με βάση την παραπάνω ανάλυση της μεθόδου και την εξίσωση που προέκυψε να αλλάξετε το παραπάνω πρόγραμμα, ώστε να υπολογίζει το ολοκλήρωμα με τη μέθοδο του παραλληλογραμμού και όχι με αυτή του τραπεζίου.

Η συνάρτηση $f(x)$ θα σας δοθεί στο εργαστήριο από τους διδάσκοντες.

Άσκηση 6

Άσκηση (υπολογισμός αθροίσματος - γινομένου)

Η έννοια του μεγάλου αριθμού προσθετέων

Συχνά σε αριθμητικούς υπολογισμούς υπάρχει η ανάγκη υπολογισμού ενός αθροίσματος, γινομένου, κλπ με άπειρους όρους. Δεδομένου ότι στην πράξη η πρόσθεση άπειρων αριθμών δεν είναι εφικτή, πρέπει να γράψουμε ένα πρόγραμμα στο οποίο θα έχουμε ορίσει μία παράμετρο η οποία να ``ελέγχει" το πόσο κοντά βρισκόμαστε σε αυτό που σε κάθε εφαρμογή μπορούμε να θεωρήσουμε ως «άπειρο». Ας μελετήσουμε λοιπόν το ακόλουθο πρόβλημα που υπολογίζει το άθροισμα των όρων μίας σειράς της μορφής:

$$\sum_k \frac{1}{k} = \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \dots$$

Επειδή το άθροισμα αυτό αποτελείται από άπειρους όρους, πρέπει να βάλουμε μία συνθήκη σύμφωνα με την οποία, ο υπολογιστής θα σταματήσει να αθροίζει. Η συνθήκη αυτή θα μπορούσε να είναι π.χ. ότι ο H/Y θα σταματήσει τη διαδικασία του αθροίσματος όταν φτάσει στον όρο εκείνο που θα είναι μικρότερος από κάποιο συγκεκριμένη τιμή, έστω $\epsilon = 5 \times 10^{-4}$. Για να το πετύχουμε αυτό, χρησιμοποιούμε είτε την εντολή `while` είτε την `do . . . while`. Η εντολή αυτή επαναλαμβάνει μία διαδικασία τόσες φορές όσες χρειάζεται μέχρι κάποια, ορισμένη, παράμετρος να πάρει μία συγκεκριμένη τιμή.

6.1 Εφαρμογή

Παρακάτω δίνεται το πρόγραμμα, το οποίο υπολογίζει, μέσω της εντολής `while`, το παραπάνω άθροισμα.

ΠΡΟΣΟΧΗ: ΣΤΟ ΠΡΟΓΡΑΜΜΑ ΥΠΑΡΧΟΥΝ ΛΟΓΙΚΑ ΚΑΙ ΣΥΝΤΑΚΤΙΚΑ ΛΑΘΗ ΚΑΘΩΣ ΚΑΙ ΠΑΡΑΛΕΙΨΕΙΣ ΠΟΥ ΠΡΕΠΕΙ ΝΑ ΕΝΤΟΠΙΣΕΤΕ ΓΙΑ ΝΑ ΤΡΕΧΕΙ ΚΑΝΟΝΙΚΑ

```

include <stdio.h>
int main()
{
double athroisma, prostheteos, epsilon;
int i;
athroisma = 0.0;
prostheteos = 1.0;
epsilon = 5e-04;
i = 1;
while (prostheteos >= epsilon)
{
prostheteos = 1/i;
athroisma = athroisma + prostheteos;
i=i+1;
}
printf("\n To plthtos ton orwn einai = %d\n", i-1);
printf("\n To athroisma einai = %.5f\n", athroisma);
return 0;
}

```

6.2 Χρήση των εντολών **do...while**

Να αλλάξετε το παραπάνω πρόγραμμα, ώστε να δημιουργήσετε ένα καινούριο, με τέτοιο τρόπο ώστε να μην χρησιμοποιεί την εντολή `while` για τη λειτουργία του αλλά τη σύνθετη εντολή `do...while`. Θυμίζουμε ότι, όπως είναι γνωστό και από το μάθημα, για να λειτουργήσει η εντολή `while` πρέπει να έχει κάτι να συγκρίνει πριν την πρώτη επανάληψη της διαδικασίας. Αντίθετα, η σύνθετη εντολή `do...while`, πρώτα κάνει την πρώτη επανάληψη και μετά ζητάει να συγκρίνει για να αποφασίσει αν θα συνεχίσει ή όχι τις επαναλήψεις.

6.3 Άσκηση: Πολλαπλασιασμός με μεγάλο αριθμό όρων

Να γράψετε ένα πρόγραμμα το οποίο δεν θα υπολογίζει το άθροισμα άπειρων όρων όπως παραπάνω, αλλά το ακόλουθο γινόμενο πολλών όρων:

$$\prod_{k=1}^{+\infty} a_k = \prod_{k=1}^{+\infty} \left(1 - \frac{1}{4k^2}\right) = \frac{3}{4} \times \frac{15}{16} \times \frac{35}{36} \times \frac{63}{64} \times \dots$$

Όπως φαίνεται, οι όροι για μεγάλα k πλησιάζουν ολοένα τη μονάδα. Το πρόγραμμα θα πρέπει να σταματάει τον υπολογισμό όταν κάποιος από τους όρους του γινομένου γίνει φτάσει αρκετά κοντά στη μονάδα. Δηλαδή θα πρέπει η ποσότητα $|1 - a_k|$ να είναι ίση ή μικρότερη από μία τιμή ϵ , η οποία θα πρέπει να ζητηθεί από τον χρήστη στην αρχή του προγράμματος (να χρησιμοποιηθεί η εντολή `scanf`). Όταν θα σταματήσει τον υπολογισμό, θα πρέπει να εκτυπώνει στην οθόνη το συνολικό γινόμενο, την τιμή του k στην οποία έφτασε και την τιμή του τελευταίου παράγοντα του γινομένου, πριν να σταματήσει η διαδικασία. Έχει ενδιαφέρον ότι το παραπάνω απειρογινόμενο ισούται με $\frac{2}{\pi}$.

6.4 Άσκηση: Επιλογή με την εντολή **switch**

Με βάση τα προγράμματα που δημιουργήσατε στις παραγράφους 6.1 και 6.3 να κατασκευάσετε ένα νέο το οποίο θα ζητάει από τον χρήστη να εισάγει την επιλογή του αν θέλει να κάνει πρόσθεση ή πολλαπλασιασμό. Η επιλογή του χρήστη θα εισάγεται μέσω της εντολής `scanf` ενώ ο τρόπος με τον οποίο θα επιλέγεται ο πολλαπλασιασμός ή η πρόσθεση θα γίνεται μέσω των εντολών `switch...case...default`.

Άσκηση 7

Άσκηση (πράξεις με πίνακες)

Πολλαπλασιασμός Πινάκων

Όπως είναι γνωστό δύο πίνακες $N \times P$, $P \times K$ πολλαπλασιαζόμενοι με αυτή τη σειρά δίνουν ένα πίνακα $N \times K$. Σε αυτή την άσκηση εισάγονται δύο πίνακες και το πρόγραμμα δίνει το αποτέλεσμα τους. Μπορείτε όμως να αλλάξετε τους N , P , K , οι οποίοι δίνονται ως σταθερές στην αρχή του προγράμματος και να εκτελέσετε έτσι όποιο πολλαπλασιασμό θέλετε.

7.1 Εφαρμογή

Παρακάτω δίνεται το πρόγραμμα, το οποίο εκτελεί πολλαπλασιασμό πινάκων.

ΠΡΟΣΟΧΗ: ΣΤΟ ΠΡΟΓΡΑΜΜΑ ΥΠΑΡΧΟΥΝ ΛΟΓΙΚΑ ΚΑΙ ΣΥΝΤΑΚΤΙΚΑ ΛΑΘΗ ΚΑΘΩΣ ΚΑΙ ΠΑΡΑΛΕΙΨΕΙΣ ΠΟΥ ΠΡΕΠΕΙ ΝΑ ΕΝΤΟΠΙΣΕΤΕ ΓΙΑ ΝΑ ΤΡΕΧΕΙ ΚΑΝΟΝΙΚΑ

```

#include <stdio.h>

int main(void)
{
    int n, m, p;
    int i, j;

    printf("\n In this program we will multiply a MxN matrix
    A with an NxP matrix "B");
    printf("\n Give number of rows of matrix A, M= ");
    printf("\n Give number of columns of matrix A, N= ");
    scanf("%d", n);
    printf("\n Give number of columns of matrix B, P= ");
    scanf("%d",&p);

    // Here we define the matrices A, B
    double a[m][n];
    double bb[n][p];
    double c[m][p];

    // Read elements of matrix A

    // Read elements of matrix B
    printf("\nGive elements of B\n");
    for (i=0; i<n; ++i)
    {
        {
            printf("\nB[%d][%d] = ", i+1 ,j+1);
            scanf("%lf",&b[i][j]);
        }
    }

    // Multiply A*B, element by element
    for (i=0; i<m; ++i)
    {
        for (j=0; j<p; ++j)
        {
            temp = 0.0;
            for (k=0; k<n; ++k)
                temp += a[i][k]*b[k][j];
            c[i][j] = temp;
            printf("\n C[%d][%d] = %f, i+1, j+1, c[i][j]);
        }
    }
    return 0;
}

```

Παρατηρήστε ότι για τον υπολογισμό του πίνακα $C = A \times B$ χρησιμοποιούνται 3 δομές for, η μία μέσα την άλλη.

7.2 Άσκηση: Υπολογισμός αθροίσματος πινάκων

Είναι γνωστό ότι η πρόσθεση και η αφαίρεση πινάκων μπορεί να πραγματοποιηθεί μόνο μεταξύ πινάκων που έχουν την ίδια διάσταση. Αν θεωρήσουμε δύο πίνακες A και B με ίδιες διαστάσεις, έστω $N \times M$, τότε η πρόσθεση ή η αφαίρεση τους θα δίνει έναν πίνακα C , επίσης $N \times M$, τα στοιχεία του οποίου θα υπολογίζονται ως εξής:

$$c_{ij} = a_{ij} + b_{ij}, \quad \text{όπου } i = 1, \dots, N \quad \text{και} \quad j = 1, \dots, M$$

Με βάση τον ορισμό αυτό, να αλλάξετε το παραπάνω πρόγραμμα ώστε να υπολογίζει το άθροισμα $C = A + B$ και να τυπώνει τα περιεχόμενα του πίνακα C στο αρχείο `test3.dat`, με κάθε στήλη του πίνακα να απέχει από την άλλη κατά ένα tab.

Άσκηση 8

Άσκηση (αποθήκευση και ανάκτηση δεδομένων από το δίσκο)

Ποια είναι η λογική της αποθήκευσης

Πολύ συχνά τα δεδομένα τα οποία προκύπτουν από ένα πρόγραμμα πρέπει να επεξεργαστούν με κάποια άλλη εφαρμογή ή τα αποτελέσματα αυτά πρέπει να αποθηκευτούν ώστε να μπορούν, σε κάποια άλλη στιγμή, να ανακτηθούν από το χρήστη για να τα χρησιμοποιήσει. Επομένως, είναι σημαντικό να έχουμε τη δυνατότητα να αποθηκεύουμε δεδομένα στο δίσκο του υπολογιστή μας. Στην ακόλουθη εφαρμογή, υπολογίζουμε τις τιμές μιας συνάρτησης και τα αποτελέσματα τα αποθηκεύουμε στον δίσκο.

8.1 Εφαρμογή

Το ακόλουθο πρόγραμμα υπολογίζει N_{max} τυχαίους αριθμούς μεταξύ του $(0, 1)$. Για να το κάνει αυτό χρησιμοποιεί πρώτα την συνάρτηση `rand()` για να δημιουργήσει ακέραιους αριθμούς μέχρι καποιον μέγιστο που δίνεται από τον χρήστη (μικρότερο του `RAND_MAX=32767`) και κατόπιν τους διαιρεί με το μέγιστο αριθμό. Επειδή η `rand()` έχει παντα το ίδιο `seed`, χρησιμοποιούμε την `srand(seed)` με `seed` που δίνεται από τη συγκεκριμένη ώρα μέσω της συνάρτησης `time()`. Τα αποτελέσματα αποθηκεύονται σε ένα αρχείο της μορφής `.dat` και μπορούν να χρησιμοποιηθούν για επεξεργασία από οποιοδήποτε πρόγραμμα υποστηρίζει αυτού του τύπου τα αρχεία.

ΠΡΟΣΟΧΗ: ΣΤΟ ΠΡΟΓΡΑΜΜΑ ΥΠΑΡΧΟΥΝ ΛΟΓΙΚΑ ΚΑΙ ΣΥΝΤΑΚΤΙΚΑ ΛΑΘΗ ΚΑΘΩΣ ΚΑΙ ΠΑΡΑΛΕΙΨΕΙΣ ΠΟΥ ΠΡΕΠΕΙ ΝΑ ΕΝΤΟΠΙΣΕΤΕ ΓΙΑ ΝΑ ΤΡΕΧΕΙ ΚΑΝΟΝΙΚΑ

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main()
{
    FILE* fp;
    double z;
    int y, Nmax, i;
    int seed = time(NULL);

    srand(seed);
    fp=fopen("RandUniform.dat", "w"); /*ανοίγει το αρχείο για εγγραφή */
    printf("How many random numbers do you want?\n");
    scanf("%d", &Nmax);

    printf("\n What is the random number step?    ");
    scanf("%lf", &step);

    y = ceil(step);

    for(i=0; i<Nmax; i++)
    {
        z = (rand()%y)*step;

        fprintf(fp, "%d, %lf \n", i, z);
    }

    fclose(fp);
    return 0;
}
```

8.2 Άσκηση

Να κάνετε τις απαραίτητες τροποποιήσεις στο παραπάνω πρόγραμμα ώστε να ανοίγει ένα αρχείο για αποθήκευση στοιχείων, που θα ονομάζεται `tzoker.dat`, το οποίο υπολογίζει αριθμούς TZOKER. Δηλαδή χρησιμοποιώντας την συνάρτηση `rand()` να επιλέγονται 5 αριθμοί από το 1 έως το 45 και 1 αριθμός από το 1 έως το 20.

8.3 Ανάγνωση αρχείου

Όπως αναφέρθηκε και στην εισαγωγή της παρούσας άσκησης, τα αρχεία στα οποία έχουμε αποθηκεύσει τα διάφορα δεδομένα, μπορούν να διαβαστούν και τα δεδομένα αυτά να χρησιμοποιηθούν. Με βάση αυτή τη δυνατότητα, να κατασκευάσετε ένα πρόγραμμα, το οποίο όταν θα τρέχει, θα διαβάζει τα στοιχεία του αρχείου `RandUniform.dat` και στη συνέχεια θα εκτυπώνει στην οθόνη τη μέση τιμή και τη διασπορά των τιμών του

$f(x)$. Υπενθυμίζεται ότι, η μέση τιμή μ και η διασπορά σ^2 ορίζονται ως εξής

$$\mu = \frac{\sum_{k=1}^N f(x_k)}{N}$$
$$\sigma^2 = \frac{\sum_{k=1}^N (f(x_k) - \mu)^2}{N} = \frac{\sum_{k=1}^N f^2(x_k)}{N} - \mu^2$$

ΠΡΟΣΟΧΗ: ΣΤΟ ΠΡΟΓΡΑΜΜΑ ΥΠΑΡΧΟΥΝ ΛΟΓΙΚΑ ΚΑΙ ΣΥΝΤΑΚΤΙΚΑ ΛΑΘΗ ΚΑΘΩΣ ΚΑΙ ΠΑΡΑΛΕΙΨΕΙΣ ΠΟΥ ΠΡΕΠΕΙ ΝΑ ΕΝΤΟΠΙΣΕΤΕ ΓΙΑ ΝΑ ΤΡΕΧΕΙ ΚΑΝΟΝΙΚΑ

```
include <stdio.h>
include <stdlib.h>

int main()
{
FILE* fp;
double timi_f, athroisma_x, athroisma_x2, mesos, diaspora;
int timi_x, plithos;

fp=fopen("RandUniform.dat", "r");

fscanf(fp, "%d,%lf", &timi_x, &timi_f);

while(!feof(fp)){
printf("\n%d\t%lf\t", timi_x, timi_f);
fscanf(fp, "%d,%lf", &timi_x, &timi_f);
plithos +=1;
athroisma_x += timi_f;
athroisma_x2 += pow(timi_f,5);
}
mesos = athroisma_x/plithos;
diaspora = athroisma_x2;

printf("\nN = \t %d\n", plithos);
printf("Mesos = \t %lf\n", mesos);
printf("Diaspora = \t%lf \n", diaspora);

fclose(fp);
return 0;
}
```


Άσκηση 9

Συναρτήσεις, καθολικές και τοπικές μεταβλητές

Συναρτήσεις (Functions)

Μερικά τμήματα ενός προγράμματος είναι σχεδόν αυτόνομα και μπορούν να γραφούν ξεχωριστά από το κύριο πρόγραμμα. Η C παρέχει αυτή τη δυνατότητα με τις functions. Μια συνάρτηση έχει δομή όμοια με αυτήν ενός προγράμματος, καλείται δε από το κυρίως πρόγραμμα ή από άλλες συναρτήσεις. Οι συναρτήσεις επιστρέφουν μια τιμή στο πρόγραμμα μέσω του ονόματος τους, ή αλλάζουν την τιμή κάποιας μεταβλητής του κυρίως ή άλλου προγράμματος..

Καθολικές και τοπικές μεταβλητές

Κατά την εκτέλεση ενός προγράμματος δεσμεύεται μνήμη-χώρος για τις μεταβλητές που χρησιμοποιούνται. Οι μεταβλητές αυτές έχουν καθολική ισχύ και ονομάζονται καθολικές (global) μεταβλητές. Όταν κληθεί μια συνάρτηση δεσμεύεται μνήμη για τις εσωτερικές (local) μεταβλητές τις. Αυτές έχουν ισχύ εσωτερικά στη διαδικασία και η μνήμη που καταλαμβάνουν αποδεσμεύεται μετά το τέλος της διαδικασίας. Χρησιμοποιώντας τοπικές μεταβλητές οι διαδικασίες αποκτούν αυτονομία και μεγαλύτερη εμβέλεια χρήσης.

Παράμετροι τιμής

Η C δίνει τη δυνατότητα να περαστούν τιμές σε μεταβλητές της διαδικασίας όταν αυτή καλείται. Έτσι μια διαδικασία επικοινωνεί με τον έξω χώρο και μπορεί να **παραμετροποιηθεί**. Με αυτό τον τρόπο καθίσταται αυτόνομη και μπορεί να χρησιμοποιηθεί ακόμα και από άλλα προγράμματα. Στο πρόγραμμα μας δηλώνουμε τη συνάρτηση factorial με τον τρόπο που φαίνεται πιο κάτω:

```
int factorial(int n_int)
```

Η παράμετρος n_int είναι μια ακέραια παράμετρος η οποία κληροδοτείται από το κύριο πρόγραμμα κατά την κλήση της συνάρτησης. Έτσι στο πρόγραμμα καλούμε την factorial με τον εξής τρόπο :

```
k = factorial(n);
```

Η τιμή της μεταβλητής n δίνεται στην παράμετρο n_int της συνάρτησης. Σε αυτήν την περίπτωση η συνάρτηση επιστρέφει τιμές μέσω του ονόματος της.

Η χρήση συνάρτησης με δείκτες

Ο δεύτερος τρόπος χρήσης συνάρτησης παραπάνω (void calculate_factorial(int n, int *m)) είναι μία κλασική χρήση δεικτών στη C. Αν δεν είχαμε βάλει τον αστερίσκο στο όρισμα της συνάρτησης, η τιμή της m δε θα άλλαζε όταν την καλούσαμε από την main. Ουσιαστικά με αυτόν τον τρόπο, καλώντας τη μεταβλητή με αναφορά, της δίνουμε τη δυνατότητα να αλλάξει τιμή στο κυρίως πρόγραμμα.

$N!$ (N παραγοντικό)

Το παραγοντικό ($N!$), ως γνωστό, δίνεται από τη σχέση

$$N! = \begin{cases} 1 \times 2 \times 3 \times \dots \times (N-1) \times N, & \text{αν } N > 0 \\ 1, & \text{αν } N = 0 \end{cases}$$

Στο πρόγραμμα μας υπολογίζουμε με δύο τρόπους το $N!$. Στη μία η συνάρτηση επιστρέφει την τιμή $N!$ μέσω του ονόματός της. Στην άλλη με κλήση μιας μεταβλητής του κυρίου προγράμματος με αναφορά (με δείκτη).

9.1 Εφαρμογή (Πολλαπλασιασμός μιγαδικών αριθμών)

Μία απλή άσκηση που χρησιμοποιεί συναρτήσεις στην C είναι ο υπολογισμός του γινομένου μιγαδικών αριθμών.

ΠΡΟΣΟΧΗ: ΣΤΟ ΠΡΟΓΡΑΜΜΑ ΥΠΑΡΧΟΥΝ ΛΟΓΙΚΑ ΚΑΙ ΣΥΝΤΑΚΤΙΚΑ ΛΑΘΗ ΚΑΘΩΣ ΚΑΙ ΠΑΡΑΛΕΙΨΕΙΣ ΠΟΥ ΠΡΕΠΕΙ ΝΑ ΕΝΤΟΠΙΣΕΤΕ ΓΙΑ ΝΑ ΤΡΕΧΕΙ ΚΑΝΟΝΙΚΑ

```
#include<stdio.h>

double real_product(double real1, double imag1, double real2, double imag2);
double imag_product(double real1, double imag1, double real2, double imag2);

int main(void)
{
    /*Orismos metavliton*/
    double x1, y1, x2, y2, x3, y3;

    printf("Doste to pragmatiko meros tou protou migadikou arithmou: x1 = ");
    scanf("%lf", &x1);
    printf("Doste to fantastiko meros tou protou migadikou arithmou: y1 = ");
    scanf("%lf", &y1);

    printf("Doste to pragmatiko meros tou 2ou migadikou arithmou: x2 = ");
    scanf("%lf", &x2);
    printf("Doste to fantastiko meros tou 2ou migadikou arithmou: y2 = ");
    scanf("%lf", &y2);
    x3 = real_product(x1,y1,x2,y2);
    y3 = imag_product(x1,y1,x2,y2);
    printf("\n z3 = %f + i%f\n", x3, y3);

    return 0;
}

double real_product(double real1, double imag1, double real2, double imag2)
{
    double temp = real1*real2 - imag1*imag2;
    return temp;
}

double imag_product(double real1, double imag1, double real2, double imag2)
{
    return real1*imag2 + imag1*real2;
}
```

9.2 Εφαρμογή (Παραγοντικό)

Παρακάτω δίνεται το πρόγραμμα, το οποίο διαθέτει δύο συναρτήσεις για υπολογισμό του $N!$. Η πρώτη χρησιμοποιεί τη δομή `for` ενώ η δεύτερη είναι μια αναδρομική συνάρτηση.

ΠΡΟΣΟΧΗ: ΣΤΟ ΠΡΟΓΡΑΜΜΑ ΥΠΑΡΧΟΥΝ ΛΟΓΙΚΑ ΚΑΙ ΣΥΝΤΑΚΤΙΚΑ ΛΑΘΗ ΚΑΘΩΣ ΚΑΙ ΠΑΡΑΛΕΙΨΕΙΣ ΠΟΥ ΠΡΕΠΕΙ ΝΑ ΕΝΤΟΠΙΣΕΤΕ ΓΙΑ ΝΑ ΤΡΕΧΕΙ ΚΑΝΟΝΙΚΑ

```
#include <stdio.h>
//Dilosi synartiseon:
//Edo i exodos einai typou int
int factorial(int n_int);
//Edo den yparhei exodos alla mia metabliti kaleitai me anafora (i n_factorial)
void calculate_factorial(int m_int, int *n_factorial);
int main(void)
{
    int n, m, k ;
    printf("\nGive me non-negative value of N = ");
    scanf("%d", &n);
    //O enas tropos:
    k = factorial(n);
    printf("\n%d!=%d",n,k);
    //O allos tropos:
    calculate_factorial(n,&m);
    printf("\n%d!=%d",n,m);
    return 0;
}
int factorial(int n_int)
{
    int i, fact; //i, fact einai topikes metablites
    if (n_int == 0)
        return 1;
    else
    {
        fact = 1;
        for (i=1; i<=n_int; ++i)
            fact *= i;
        return fact;
    }
}
void calculate_factorial(int m_int, int *n_factorial)
{
    int i, fact;
    if (m_int == 0)
        *n_factorial = 1;
    else
    {
        fact = 1;
        for (i=1; i<=m_int; ++i)
            fact *= i;
        *n_factorial = fact;
    }
}
```

Μέρος ΙΙΙ

Παράρτημα

Βιβλιοθήκες στη C

Πολλές φορές καλούμαστε να χρησιμοποιήσουμε συναρτήσεις που δεν ανήκουν στο βασικό σύνολο εντολών της γλώσσας C και οι οποίες είναι αποθηκευμένες στις βιβλιοθήκες (libraries). κα για να χρησιμοποιηθούν θα πρέπει πρώτα να δηλωθούν. Η δήλωση βιβλιοθηκών στη C γίνεται με την εντολή `#include` της οποίας η σύνταξη είναι

```
#include<library.h>
```

Προσοχή! Η εντολή (ή οι εντολές αν πρόκειται να χρησιμοποιήσουμε πολλές βιβλιοθήκες) γράφεται πριν την έναρξη του βασικού προγράμματος

```
#include<library1.h>
#include<library2.h>
```

```
int main(void)
{
```

Main Programm

```
}
```

Στις εργαστηριακές ασκήσεις θα χρησιμοποιηθούν τρεις βασικές βιβλιοθήκες:

1. `stdio.h`: Περιέχει όλες τις βασικές εντολές/συναρτήσεις εισόδου-εξόδου.
2. `stdlib.h`: Περιέχει επιπρόσθετες εντολές εισόδου εξόδου σχετιζόμενες με δημιουργία και προσπέλαση αρχείων.
3. `math.h`: Περιέχει μαθηματικές συναρτήσεις.

Ειδικότερα για την `math.h`, κατά την διαδικασία μεταγλώττισης του προγράμματος θα πρέπει να δηλωθεί επιπλέον στον μεταγλωττιστή η ανάγνωσή της χρησιμοποιώντας το όρισμα `-lm` (όπου `l` από το `library` και `m` από το `math`) δηλαδή

```
gcc -o executable source.c -lm
```

9.3 Τα πιο συχνά λάθη στη C

9.3.1 Μεταβλητές που δεν έχουν δηλωθεί - Undeclared Variables

Ένα συχνό λάθος στον προγραμματισμό C, είναι η μη δήλωση των μεταβλητών που χρησιμοποιούνται μέσα στο πρόγραμμα, π.χ.

```
#include <stdio.h>
int main ()
{
printf("Give variable a");
scanf("%a", &a);
return(0);
}
```

Για να είναι σωστό το πρόγραμμα, θα έπρεπε στην αρχή να έχει δηλωθεί η μεταβλητή `a`. Δηλαδή,

```
#include <stdio.h>
int main ()
{
    int a;
    printf("Give variable a");
    scanf("%a", &a);
    return(0);
}
```

Απουσία του στοιχείου ";" - Semicolon needed

Πολύ συχνά, κατά την πληκτρολόγηση του προγράμματος, ξεχνάμε το στοιχείο ";" στο τέλος των εντολών, π.χ.

```
#include <stdio.h>
int main ()
{
    int a;
    printf("Give variable a")
    scanf("%a", &a)
    return(0);
}
```

Η σωστή μορφή του προγράμματος είναι

```
#include <stdio.h>
int main ()
{
    int a;
    printf("Give variable a");
    scanf("%a", &a);
    return(0);
}
```

9.3.2 Υπερβολική χρήση του στοιχείου ";" - Semicolon

Προσοχή στη χρήση του στοιχείου ";" . Το semicolon δεν μπαίνει μετά από τις δηλώσεις if, βρόχους ή δηλώσεις συναρτήσεων.

```
#include <stdio.h>
int main ()
{
    int i;

    for (i=1;i=10;i++)
    printf("Hello World")

    return(0);
}
```

9.3.3 Διαίρεση ακέραιου

Συχνό λάθος είναι η διαίρεση ακεραίων, ενώ το αποτέλεσμα θέλουμε να είναι πραγματικός αριθμός, π.χ.

```
#include <stdio.h>
int main ()
{
double outcome;
int i;

for (i=2;i=10;i++)
{
outcome = 1/i;
printf("apotelesma =%lf",outcome);
}

return(0);
}
```

Στο παραπάνω πρόγραμμα, το `apotelesma` είναι πάντα ίσο με 0 διότι το 1 και το `i` έχουν δηλωθεί ως ακέραιοι. Δηλαδή το πρόγραμμα κρατάει μόνο το ακέραιο μέρος της διαίρεσης, που είναι 0. Για τη διόρθωση του λάθους πρέπει να γράψουμε τη διαίρεση ως πράξη πραγματικών αριθμών. Δηλαδή

```
#include <stdio.h>
int main ()
{
double outcome;
int i;

for (i=2;i=10;i++)
{
outcome = 1.0/i;
printf("apotelesma =%lf",outcome);
}

return(0);
}
```

9.3.4 Χρήση απλού "=" αντί διπλού "==" για τον έλεγχο ισότητας.

Συχνό λάθος κατά τον έλεγχο μιας ισότητας είναι η χρήση απλού "=", π.χ.

```
#include <stdio.h>
int main ()
{
int a;

a = 0;

if (a=1)
printf("Hello World");

return(0);
}
```

Στο παραπάνω πρόγραμμα, η μεταβλητή `a` αλλάζει τιμή και γίνεται 1. Για να είναι σωστό το πρόγραμμα, πρέπει να χρησιμοποιήσουμε διπλό "=", το οποίο είναι το σύμβολο της λογικής πράξης της ισότητας,

```
#include <stdio.h>
int main ()
{
int a;

a = 0;

if (a==1)
printf("Hello World");

return(0);
}
```

9.3.5 Προσπάθεια να μεταγλωττίσουμε κώδικα που δεν έχει αποθηκευτεί.

Συχνά, προσπαθούμε να μεταγλωττίσουμε (compile) κώδικα τον οποίο δεν έχουμε σώσει. Αυτό έχει ως συνέπεια, η διαδικασία του Compiling να γίνεται σε κάποια προηγούμενη εκδοχή του κώδικα και να μας εμφανίζονται λάθη που θεωρούμε πως τα έχουμε διορθώσει. Για το λόγο αυτό, μετά από κάθε αλλαγή που κάνουμε στον κώδικά μας, πρέπει πάντα να σώζουμε και μετά να μεταγλωττίζουμε.

Προβλήματα για εξάσκηση

1. Πρόγραμμα που υπολογίζει αν ένας αριθμός είναι άρτιος ή περιττός

```
#include<stdio.h>

main()
{
    int n;

    printf("Enter an integer\n");
    scanf("%d",&n);

    if ( n%2 == 0 )
        printf("Even\n");
    else
        printf("Odd\n");

    return 0;
}
```

2. Πρόγραμμα που να υπολογίζει τον μεγαλύτερο από 3 αριθμούς

```
#include<stdio.h>

int main() {
    int a, b, c;

    printf("\nEnter value of a, b & c : ");
    scanf("%d %d %d", &a, &b, &c);

    if ((a > b) && (a > c))
        printf("\na is greatest");

    if ((b > c) && (b > a))
        printf("\nb is greatest");

    if ((c > a) && (c > b))
        printf("\nc is greatest");

    return(0);
}
```

3. Πρόγραμμα που να προσθέτει τα ψηφία ενός αριθμού

```
#include <stdio.h>

int main()
{
    int n, t, sum = 0, remainder;

    printf("Enter an integer\n");
    scanf("%d", &n);

    t = n;

    while (t != 0)
    {
        remainder = t % 10;
        sum      = sum + remainder;
        t        = t / 10;
    }

    printf("Sum of digits of %d = %d\n", n, sum);

    return 0;
}
```

4. Πρόγραμμα που να υπολογίζει 10 τυχαίους αριθμούς

```
#include <stdio.h>
#include <stdlib.h>

int main() {
    int c, n;

    printf("Ten random numbers in [1,100]\n");

    for (c = 1; c <= 10; c++) {
        n = rand() % 100 + 1;
        printf("%d\n", n);
    }

    return 0;
}
```

5. Πρόγραμμα που να υπολογίζει το μέγιστο κοινό διαιρέτη και το ελάχιστο κοινό πολλαπλάσιο 2 αριθμών

```

#include <stdio.h>

int main() {
    int a, b, x, y, t, gcd, lcm;

    printf("Enter two integers\n");
    scanf("%d%d", &x, &y);

    a = x;
    b = y;

    while (b != 0) {
        t = b;
        b = a % b;
        a = t;
    }

    gcd = a;
    lcm = (x*y)/gcd;

    printf("Greatest common divisor of %d and %d = %d\n", x, y, gcd);
    printf("Least common multiple of %d and %d = %d\n", x, y, lcm);

    return 0;
}

```

6. Πρόγραμμα που να εκτυπώνει το τρίγωνο του Floyd ¹

```

#include <stdio.h>
int main()
{
    int n, i, c, a = 1;

    printf("Enter the number of rows of Floyd's triangle to print\n");
    scanf("%d", &n);

    for (i = 1; i <= n; i++)
    {
        for (c = 1; c <= i; c++)
        {
            printf("%d ", a);
            a++;
        }
        printf("\n");
    }
    return 0;
}

```

7. Πρόγραμμα που να ελέγχει αν ένας αριθμός είναι πρώτος ή όχι

¹To τρίγωνο του Floyd's

```

#include <stdio.h>
int main()
{
    int n, i, flag=0;
    printf("Enter a positive integer: ");
    scanf("%d",&n);
    for(i=2;i<=n/2;++i)
    {
        if(n%i==0)
        {
            flag=1;
            break;
        }
    }
    if (flag==0)
        printf("%d is a prime number.",n);
    else
        printf("%d is not a prime number.",n);
    return 0;
}

```

8. Πρόγραμμα που να τυπώνει τους πρώτους αριθμούς που βρίσκονται μεταξύ ενός διαστήματος που δίνεται από το χρήστη

```

#include <stdio.h>
int main()
{
    int n1, n2, i, j, flag;
    printf("Enter two numbers(intevals): ");
    scanf("%d %d", &n1, &n2);
    printf("Prime numbers between %d and %d are: ", n1, n2);
    for(i=n1+1; i<n2; ++i)
    {
        flag=0;
        for(j=2; j<=i/2; ++j)
        {
            if(i%j==0)
            {
                flag=1;
                break;
            }
        }
        if(flag==0)
            printf("%d ",i);
    }
    return 0;
}

```

9. Πρόγραμμα που να προσθέτει, να αφαιρεί, να πολλαπλασιάζει και να διαιρεί δύο μιγαδικούς αριθμούς που δίνονται από το χρήστη.

```

#include <stdio.h>
#include <stdlib.h>

struct complex
{
    int real, img;
};

int main()
{
    int choice, temp1, temp2, temp3;
    struct complex a, b, c;

    while(1)
    {
        printf("Press 1 to add two complex numbers.\n");
        printf("Press 2 to subtract two complex numbers.\n");
        printf("Press 3 to multiply two complex numbers.\n");
        printf("Press 4 to divide two complex numbers.\n");
        printf("Press 5 to exit.\n");
        printf("Enter your choice\n");
        scanf("%d",&choice);

        if( choice == 5)
            exit(0);

        if(choice >= 1 && choice <= 4)
        {
            printf("Enter a and b where a + ib is the first complex number.");
            printf("\na = ");
            scanf("%d", &a.real);
            printf("b = ");
            scanf("%d", &a.img);
            printf("Enter c and d where c + id is the second complex number.");
            printf("\nc = ");
            scanf("%d", &b.real);
            printf("d = ");
            scanf("%d", &b.img);
        }
        if ( choice == 1 )
        {
            c.real = a.real + b.real;
            c.img = a.img + b.img;

            if ( c.img >= 0 )
                printf("Sum of two complex numbers = %d + %di",c.real,c.img);
            else
                printf("Sum of two complex numbers = %d %di",c.real,c.img);
        }
        else if ( choice == 2 )
        {
            c.real = a.real - b.real;

```

```

c.img = a.img - b.img;

if ( c.img >= 0 )
    printf("Difference of two complex numbers = %d + %di",c.real,c.img);
else
    printf("Difference of two complex numbers = %d %di",c.real,c.img);
}
else if ( choice == 3 )
{
    c.real = a.real*b.real - a.img*b.img;
    c.img = a.img*b.real + a.real*b.img;

    if ( c.img >= 0 )
        printf("Multiplication of two complex numbers = %d + %di",c.real,c.img);
    else
        printf("Multiplication of two complex numbers = %d %di",c.real,c.img);
}
else if ( choice == 4 )
{
    if ( b.real == 0 && b.img == 0 )
        printf("Division by 0 + 0i is not allowed.");
    else
    {
        temp1 = a.real*b.real + a.img*b.img;
        temp2 = a.img*b.real - a.real*b.img;
        temp3 = b.real*b.real + b.img*b.img;

        if ( temp1%temp3 == 0 && temp2%temp3 == 0 )
        {
            if ( temp2/temp3 >= 0)
                printf("Division of two complex numbers = %d + %di",
                    temp1/temp3,temp2/temp3);
            else
                printf("Division of two complex numbers = %d %di",
                    temp1/temp3,temp2/temp3);
        }
        else if ( temp1%temp3 == 0 && temp2%temp3 != 0 )
        {
            if ( temp2/temp3 >= 0)
                printf("Division of two complex numbers = %d + %d/%di",
                    temp1/temp3,temp2,temp3);
            else
                printf("Division of two complex numbers = %d %d/%di",
                    temp1/temp3,temp2,temp3);
        }
        else if ( temp1%temp3 != 0 && temp2%temp3 == 0 )
        {
            if ( temp2/temp3 >= 0)
                printf("Division of two complex numbers = %d/%d + %di",
                    temp1,temp3,temp2/temp3);
            else

```

```

        printf("Division of two complex numbers = %d %d/%di",
               temp1,temp3,temp2/temp3);
    }
    else
    {
        if ( temp2/temp3 >= 0)
            printf("Division of two complex numbers = %d/%d + %d/%di",
                   temp1,temp3,temp2,temp3);
        else
            printf("Division of two complex numbers = %d/%d %d/%di",
                   temp1,temp3,temp2,temp3);
    }
}
}
else
    printf("Invalid choice.");

printf("\nPress any key to enter choice again...\n");
}
}

```