ΔΗΜΟΚΡΙΤΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΡΑΚΗΣ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Μάθημα: **ΔΟΜΗΜΕΝΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ**

Τομέας: *Λογισμικού και Ανάπτυξης Εφαρμογών*

Εργαστήριο: *Προγραμματισμού και Επεξεργασίας Πληροφοριών*

ΕΡΓΑΣΤΗΡΙΑΚΟ ΕΓΧΕΙΡΙΔΙΟ ΣΤΗ C

Αλέξανδρος Γαζής
Υποψήφιος Διδάκτορας ΗΜΜΥ ΔΠΘ,
Κάτοχος Μεταπτυχιακού Διπλώματος Ειδίκευσης ΗΜΜΥ ΔΠΘ

Ελευθερία Κατσίρη
Επίκουρη Καθηγήτρια ΗΜΜΥ ΔΠΘ

ΞΑΝΘΗ, ΔΕΚΕΜΒΡΙΟΣ 2020

Δ έκδοση

➠ Το παρόν εγχειρίδιο δημιουργήθηκε με σκοπό την παροχή επεξηγήσεων και την καθοδήγηση των προπτυχιακών φοιτητών του Δημοκρίτειου Πανεπιστημίου Θράκης, αναφορικά με το εργαστηριακό τμήμα του μαθήματος «Τεχνικές Προγραμματισμού» και «Δομημένος Προγραμματισμός» όπως διδάσκεται σήμερα. Ειδικότερα, βασίζεται στο υλικό που χρησιμοποιείται στα πλαίσια του μαθήματος, τα τελευταία χρόνια, όπως διανθίστηκε μέσω της καθημερινής αλληλεπίδρασης με τους φοιτητές του εργαστηρίου. Πιστεύουμε ότι αποτελεί ένα χρήσιμο οδηγό, που αποσαφηνίζει τη διδακτέα ύλη και επισημαίνει τα κύρια ζητήματα, που καλείται να αντιμετωπίσει ο κάθε σπουδαστής, κατά την διάρκεια των εργαστηρίων.

➠ Σε περίπτωση που προκύπτουν απορίες, κατά την ανάγνωση του εγχειριδίου, παρακαλούμε όπως επικοινωνήσετε μαζί μας, στα κάτωθι email:

Διδάσκουσα καθηγήτρια:

Ελευθερία Κατσίρη, ekatsiri@ee.duth.gr

Βοηθός καθηγητή και Υπεύθυνος εργαστηρίου:

Αλέξανδρος Γαζής, agazis@ee.duth.gr

➠ Επίσης, ευχαριστούμε τους απόφοιτους του Μεταπτυχιακού Διπλώματος Ειδίκευσης της Σχολής, κο Γιάννη Ρόιδο και κο Κωσταντίνο Σταμάτη για την συμβολή τους, κατά την συγγραφή του εγχειριδίου.

➠ Τέλος, θα θέλαμε να ευχαριστήσουμε τον κ. Καράκο Αλέξανδρος, αφυπηρετήσαντα καθηγητή του τμήματος ΗΜΜΥ του Δημοκριτείου Πανεπιστημίου Θράκης και τον δρ. Συμεωνίδη Σίμο, απόφοιτο του τμήματος ΗΜΜΥ, για την συνεισφορά τους.

ΠΕΡΙΕΧΟΜΕΝΑ

σελ.

ΠΙΝΑΚΑΣ ΕΙΚΟΝΩΝ

Εισαγωγή

1. Εγγραφή στο μάθημα μέσω της πλατφόρμας του eclass

Η πλατφόρμα «**DUTHNET eClass»** αποτελεί ένα ολοκληρωμένο σύστημα διαχείρισης ηλεκτρονικών μαθημάτων. Ακολουθεί τη φιλοσοφία του λογισμικού ανοικτού κώδικα και υποστηρίζει την υπηρεσία Ασύγχρονης Τηλεκπαίδευσης, χωρίς περιορισμούς και δεσμεύσεις. Η πρόσβαση στην υπηρεσία γίνεται με τη χρήση ενός προγράμματος πλοήγησης (web browser), χωρίς την απαίτηση εξειδικευμένων τεχνικών γνώσεων.

Για την παρακολούθηση του μαθήματος απαιτείται η εγγραφή σας στο μάθημα, ώστε να λαμβάνετε ενημερώσεις, τόσο για την θεωρία όσο και για το εργαστηριακό μέρος. Για την εγγραφή, ακολουθείτε τα παρακάτω βήματα:

1. Στη γραμμή διευθύνσεων (web address) του περιηγητή σας, εισάγετε την παρακάτω ηλεκτρονική διεύθυνση:

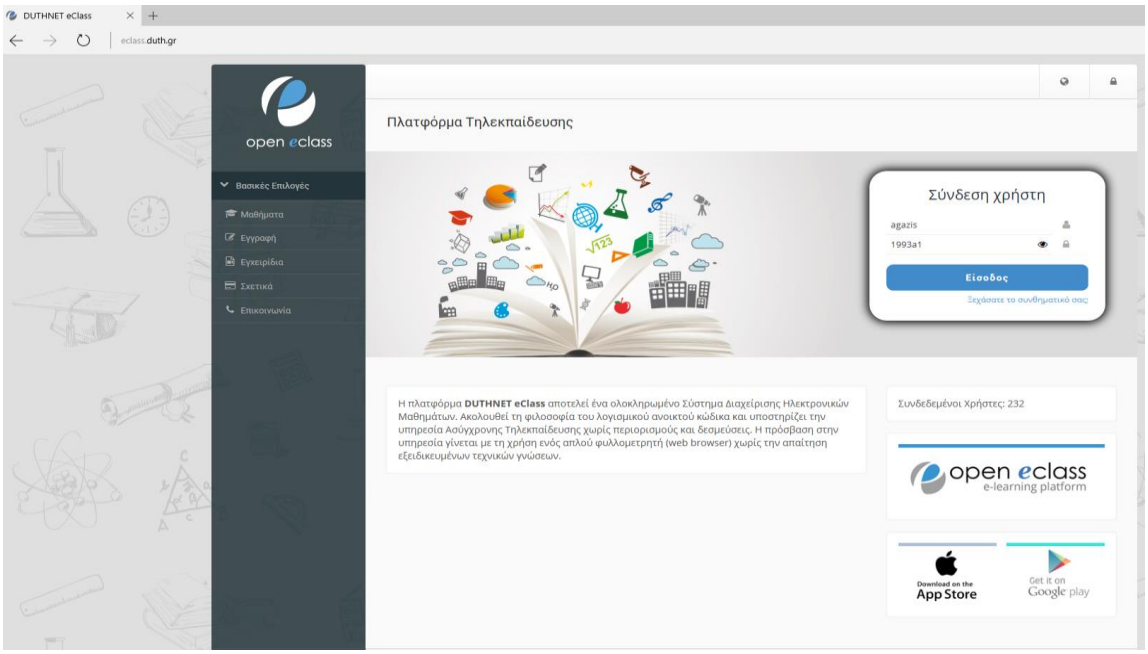
   https://eclass.duth.gr

2. Στο πεδίο «Σύνδεση χρήστη», εισάγετε το όνομα χρήστη καθώς και τον κωδικό που σας παρέχει η γραμματεία του Δημοκρίτειου Πανεπιστημίου Θράκης.
   Τονίζεται ότι το όνομα χρήστη μπορεί να προκύψει και από τα στοιχεία σύνδεσης στην υπηρεσία ηλεκτρονικών μηνυμάτων (https://**webmail.duth**.gr/). Ως εκ τούτου, το όνομα χρήση και ο κωδικός σας στην πλατφόρμα του eclass (λ.χ. agazis και 1993a1) είναι ταυτόσημα με τα στοιχεία που ζητούνται κατά την είσοδο στην υπηρεσία ηλεκτρονικών μηνυμάτων.
   Τέλος, τονίζεται ότι το πανεπιστημιακό email κάθε φοιτητή της σχολής προκύπτει αυτόματα από το όνομα χρήστη, συνοδευόμενο από @ee.duth.gr . Συνεπώς, αν το όνομα χρήστη είναι agazis, το email είναι agazis@ee.duth.gr .
   Οποιαδήποτε επικοινωνία με τον διδάσκοντα ή τον υπεύθυνο εργαστηρίου πραγματοποιείται μέσω του email του πανεπιστημίου.
3. Στο νέο παράθυρο, όπου μεταφέρεστε, μετά την επιτυχή σύνδεση, αναζητείτε στις επιλογές, στο αριστερό τμήμα της οθόνης την καρτέλα «Μαθήματα».
4. Στην συνέχεια, επιλέγετε τον σύνδεσμο για το προπτυχιακό επίπεδο (με γαλάζια γράμματα).

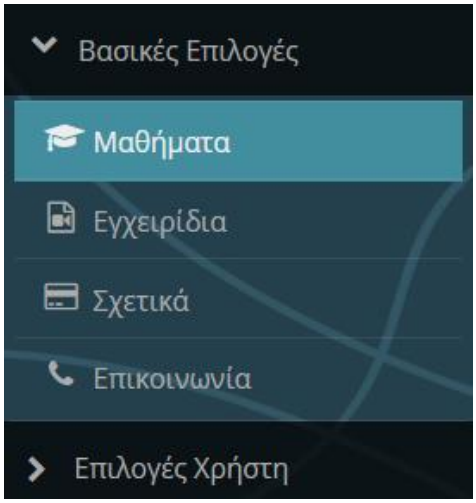
5. Αναζητείτε το νέο παράθυρο που προκύπτει το μάθημα, στο οποίο επιθυμείτε να εγγραφείτε, το ανοίγετε και επιλέγετε το κουτί στα αριστερά του τίτλου. Απαιτείται προσοχή στον εκάστοτε διδάσκοντα και στο ΤΜΑ (κωδικός εκάστοτε μαθήματος), ώστε η επιλογή σας να συμβαδίζει με τις οδηγίες που θα σας δοθούν από τον καθηγητή στο εισαγωγικό μάθημα.

Ακολουθεί σχηματική απεικόνιση των παραπάνω βημάτων:

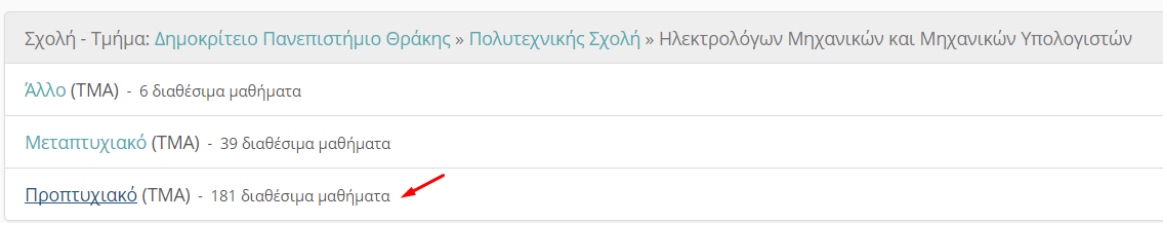
Εικόνα 1 *Βήμα 1: εισαγωγή ηλεκτρονικής διεύθυνσης σε περιηγητή*



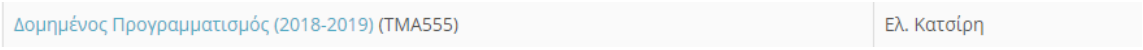
Εικόνα 2 *Βήμα 2: εισαγωγή στοιχείων στην Σύνδεση χρήστη*



Εικόνα 3 *Βήμα 3: επιλογή αριστερά Μαθήματα*



Εικόνα 4 *Βήμα 4: επιλογή Προπτυχιακό*



Εικόνα 5 *Βήμα 5: επιλέγουμε και εγγραφόμαστε στο μάθημα*

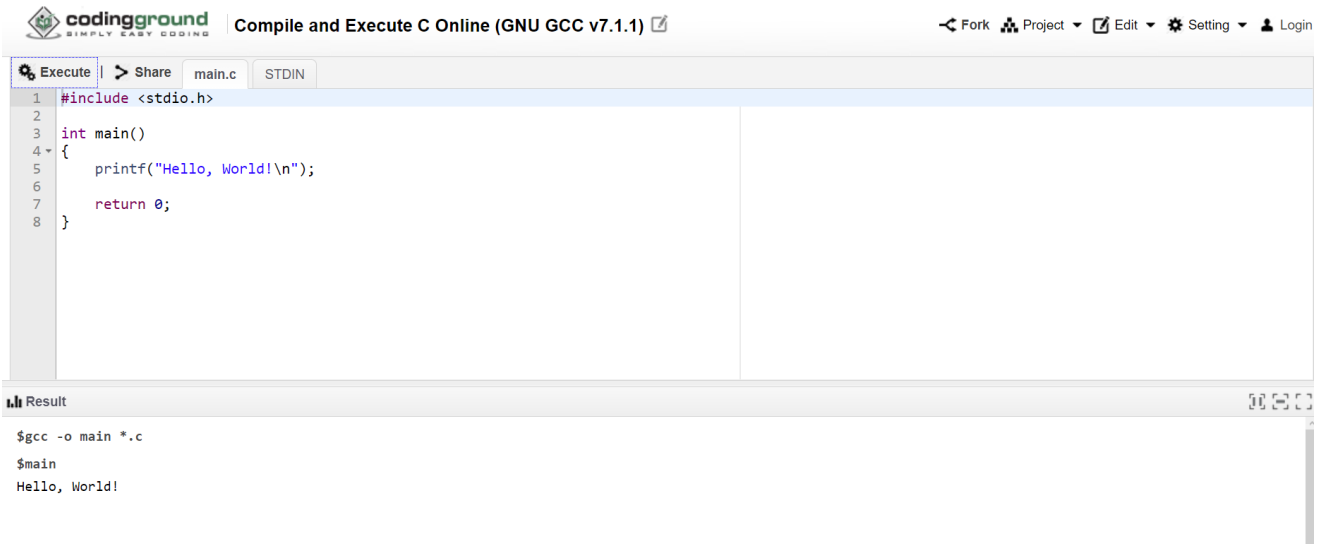
2. Παρουσίαση Εργαστηριακών Εργαλείων και Πρώτο Πρόγραμμα

Για την ορθή παρακολούθηση και συμμετοχή στο εργαστηριακό μέρος του μαθήματος, απαιτείται η εξοικείωση του αναγνώστη με έναν compiler της προγραμματιστική γλώσσας C. Αρχικά, προτείνεται η χρήση ενός από τα εξής εργαλεία:

- ❑ Online Εργαλεία προγραμματισμού σε γλώσσα C
- ❑ Χρήση Τοπικών Εργαλείων - Dev-C++ και ChIDE

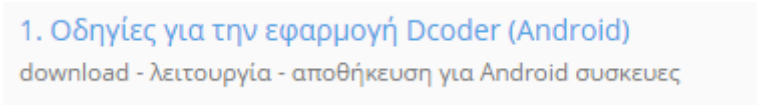
❑ Online Εργαλείο προγραμματισμού σε γλώσσα C

Στην πρώτη κατηγορία, εντάσσεται ο ιστότοπος του tutorialpoint: www.tutorialspoint.com/compile_c_online.php, ο οποίος παρέχει την δυνατότητα ελέγχου και εκτέλεσης κώδικα, από οποιαδήποτε ηλεκτρονική συσκευή, με σύνδεση στο διαδίκτυο. Επιπρόσθετα, προτείνονται εναλλακτικά, οι εφαρμογές Dcoder, για λειτουργικό σύστημα Android καθώς και CppCode ή Xcode, για iOS.

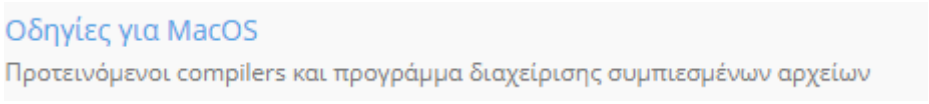


Εικόνα 6 *Περιβάλλον εργασίας διαδικτυακού εργαλείου προγραμματιστικού κώδικα*

Για την εύρεση και εγκατάσταση του Dcoder-εργαλείου, σε Android λογισμικό (Έγγραφα/Εργαστήρια/Βοηθητικό Υλικό/Androis-iOS συσκευές):



Για την εύρεση των εργαλείων CppCode ή Xcode-εργαλείου, σε iOS λογισμικό (Έγγραφα/Εργαστήρια/Βοηθητικό Υλικό/Androis-iOS συσκευές):



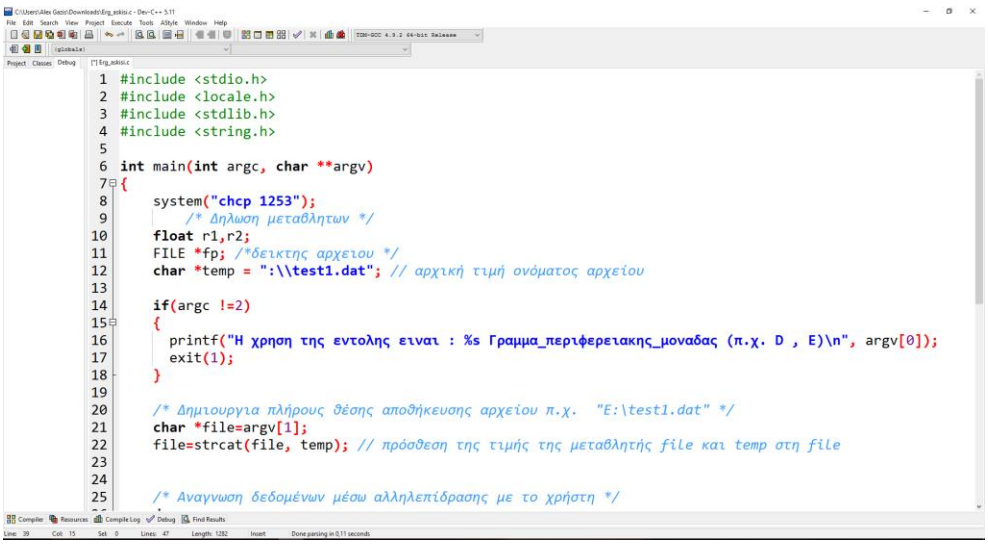
❑ Χρήση Τοπικών Εργαλείων - Dev-C++ και ChIDE

Στη δεύτερη κατηγορία, εντάσσονται τα εργαλεία που αφορούν την εγκατάσταση, τοπικά στον υπολογιστή σας, ενός προγράμματος, με δυνατότητα ελέγχου και εκτέλεσης του κώδικα, που θα είναι γραμμένος σε γλώσσα προγραμματισμού C.

Η πρώτη λύση που προτείνεται, κυρίως σε ό,τι αφορά λογισμικό Windows, αποτελεί το Dev-C++. Ειδικότερα, αυτό το εργαλείο συνθέτει ένα ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) για τις γλώσσες προγραμματισμού C και C++. Ο μεταγλωττιστής που χρησιμοποιείται είναι ο MinGW, ωστόσο μπορεί να χρησιμοποιηθεί οποιοσδήποτε μεταγλωττιστής βασίζεται στη συλλογή GCC (GNU Compiler Collection).

Χαρακτηρίζεται από:

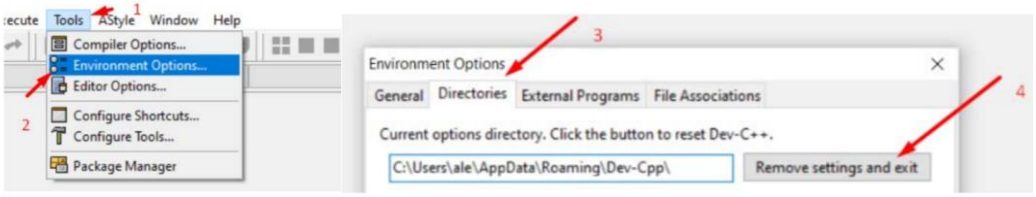
- Ενσωματωμένη αποσφαλμάτωση.
- Διαχειριστή έργων.
- Προσαρμοσμένο επεξεργαστή επισήμανσης κώδικα.
- Αυτόματη συμπλήρωση κώδικα.
- Δημιουργία αρχείων παραγωγής εκτελέσιμων (makefile).
- Υποστήριξη CVS.



Εικόνα 7 *Βασικό παράδειγμα υλοποίησης προγράμματος σε γλώσσα C στο εργαλείο Dev-C++*

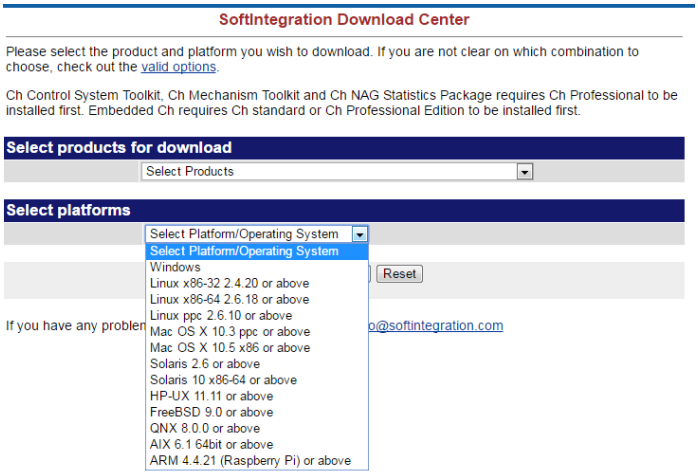
Τονίζεται ότι για το DevCpp, το οποίο κυρίως θα χρησιμοποιηθεί κατά την διάρκεια του εργαστηρίου, σε περίπτωση που κάποια ρύθμισή ή κάποια επιλογή ρυθμίσεων είναι εσφαλμένη, για να το επαναφέρουμε στις προεπιλεγμένες ρυθμίσεις απαιτούνται τα παρακάτω: >> Tools >> Environment Options >> Directories, επιλογή «Remove settings and Exit»

Ακολουθεί γραφική απεικόνιση στην Εικόνα 8 των παραπάνω εντολών:

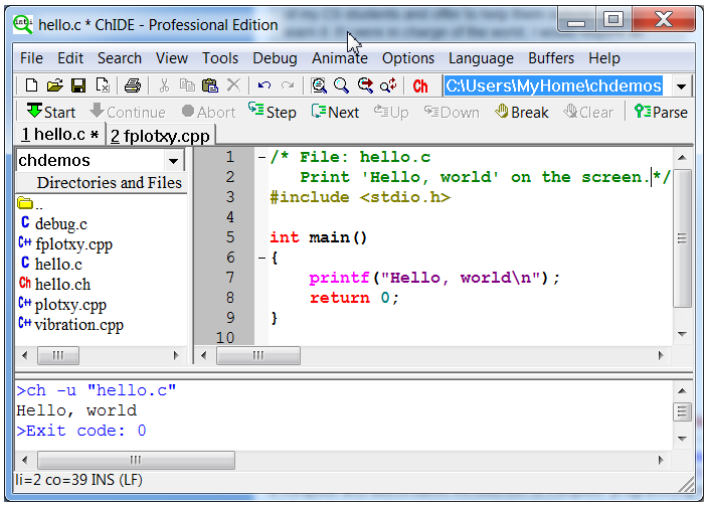


Εικόνα *8 Απαραίτητες επιλογές στο DevCpp για την επαναφορά στις προεπιλεγμένες ρυθμίσεις συστήματος*

Η δεύτερη λύση που προτείνεται, κυρίως σε ό,τι αφορά λογισμικό Windows αλλά και Linux, αποτελεί το ChIDE. Όπως και το προηγούμενο προγραμματιστικό εργαλείο, αποτελεί ένα περιβάλλον μεταγλωττιστής και ερμηνείας (interpret) για C ή C++ . Χρησιμοποιείται ευρέως από καθηγητές, επιστήμονες, φοιτητές και μηχανικούς σε όλο το κόσμο, κυρίως για τη υλοποίηση μαθηματικών, αριθμητικής ανάλυσης προβλημάτων καθώς και άλλων προγραμμάτων σε υπολογιστικά συστήματα είτε ενσωματωμένα είτε γενικής χρήσης και πλατφόρμας. Πιο συγκεκριμένα, συνθέτει μια από τις πληρέστερες εφαρμογές και γνωρίζει ιδιαίτερη άνθηση τα τελευταία χρόνια, λόγω της χρησιμοποίησής του στην εκπαίδευση και διδασκαλία, μέσω Arduino και Rasberry Pi.



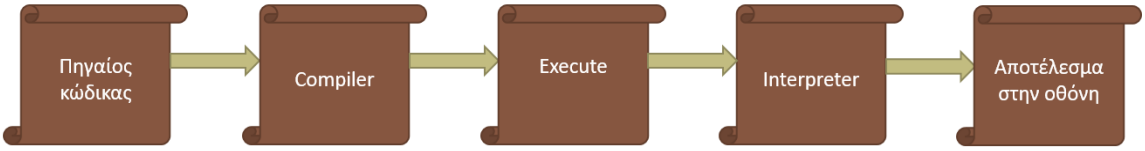
Εικόνα 9 *Λειτουργικά Συστήματα Εγκατάστασης ChIDE*



Εικόνα 10 *Βασικό παράδειγμα υλοποίησης προγράμματος σε γλώσσα C στο εργαλείο ChIDE*

Εργαστήριο 1- Μεταβλητές και Βασικές Εντολές Εκτύπωσης Τιμών

Το παρακάτω σχήμα αντιπροσωπεύει την βασική λειτουργία που υλοποιείται εντός του ηλεκτρονικού υπολογιστή, για την επεξεργασία του προγραμματιστικού κώδικα και την εξαγωγή των αποτελεσμάτων στην οθόνη του υπολογιστή.



Εικόνα 11 *Διαδικασία υλοποίησης για την εξαγωγή αποτελεσμάτων στην οθόνη του χρήστη*

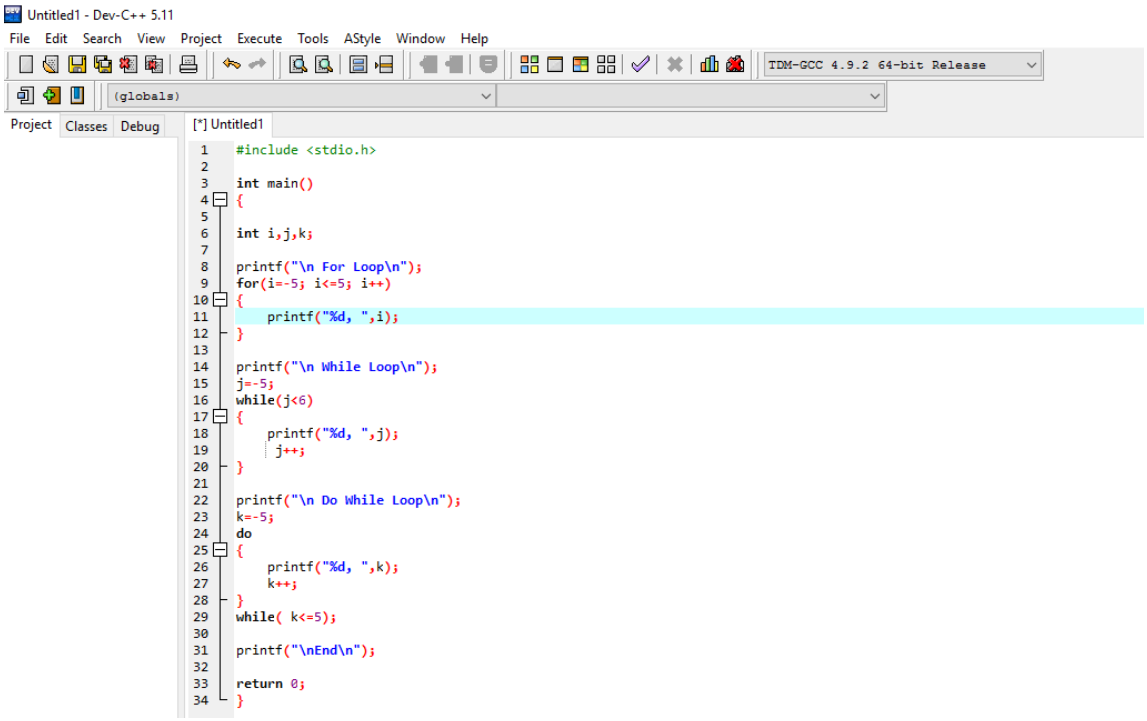
Οι παραπάνω έννοιες επεξηγούνται αναλυτικά στα προτεινόμενο εγχειρίδιο του μαθήματος, από την υπηρεσία «Εύδοξος». Στα πλαίσια του εργαστηρίου, απαιτείται η εξοικείωση και κατανόηση των παρακάτω όρων:

- Compiler: πρόγραμμα που μετατρέπει το πηγαίο κώδικα από μια γλώσσα προγραμματισμού σε μια άλλη (με σκοπό να τρέξουμε ένα πρόγραμμα).
- Interpreter: πρόγραμμα που εκτελεί *μόνο ένα* κώδικα ή πρόγραμμα.

Στα πλαίσια του εργαστηρίου, δεν θα ασχοληθούμε με τα παραπάνω, καθώς όλα τα εργαλεία που χρησιμοποιούμε κατά την εκτέλεση του προγράμματος επιτελούν αυτόματα compile (όπως και interpret). Ακολουθεί η σχετική σχηματική απεικόνιση, στο περιβάλλον του Dev-C++ .

Αρχικά, εγκαθιστούμε την εφαρμογή, από την ιστοσελίδα:
https://sourceforge.net/projects/orwelldevcpp/

Εικόνα 12 *Εγκατάσταση DevC++*



Εικόνα 13 *Περιβάλλον DevC++*

1. Κώδικας υλοποίησης - «Γεια σου Κόσμε»

Στα πλαίσια του εργαστηρίου, απαιτείται η δημιουργία ενός προγράμματος, το οποίο θα εμφανίζει στην οθόνη του χρήστη το μήνυμα «Γεια σου κόσμε. Hello world!». Σημειώνεται ότι, όσον αφορά τον τρόπο, με τον οποίο γράφουμε τον προγραμματιστικό κώδικα στο Dev-C++, ανατρέχετε στο Κεφάλαιο *Τεχνικό Παράρτημα/ Εγκατάσταση και χρήση του Dev-C++.*

Ο κώδικας, ο οποίος απαιτείται για να υλοποιήσουμε το σχετικό μήνυμα ως έξοδο στην οθόνη του χρήστη, είναι ο εξής:

```
#include <stdio.h>
                int main()
        {
        printf("Hello World, Γεια σου Κόσμε!\n"); //σχολια
            return 0;
        }
```

Με βάση το παραπάνω παράδειγμα, στην οθόνη του χρήστη θα εμφανιστεί το μήνυμα:

```
Hello World, Γεια σου Κόσμε!
```

Στην συνέχεια, ακολουθεί σύντομη παρουσίαση των παραπάνω εντολών:

- `#include <stdio.h>`

  Η εντολή include δηλώνει τη εισαγωγή της μιας βιβλιοθήκης, ενώ το κείμενο εντός <…> διευκρινίζει την ονομασία της.

- `int main ()`

  Δηλώνει, τον τύπο επιστροφής της συνάρτησης. [1]

- `printf("Hello World, Γεια σου Κόσμε!\n");`

  Τυπώνει το μήνυμα Hello World, , Γεια σου Κόσμε!

- **;**

  Χρησιμοποιείται στο τέλος κάθε εντολής.

- //

  Χρησιμοποιούνται για να γράφουμε σχόλια στο κώδικα. Δεν εκτελούνται σε καμία περίπτωση και δεν εμφανίζονται *ποτέ* στην οθόνη του χρήστη.

2. Εναλλακτικοί τρόποι δήλωσης main συνάρτησης

Στην προηγούμενη ενότητα, παρουσιάστηκαν ένα πρόγραμμα το οποίο εκτύπωνε στην οθόνη του υπολογιστή το παρακάτω μήνυμα:

```
Hello World, Γεια σου Κόσμε!
```

Με βάση τα παραπάνω, τονίζεται ότι κάθε πρόγραμμα σε γλώσσα προγραμματισμού C επιβάλλεται να έχει μία μόνο main συνάρτηση.

Η main συνάρτηση θα μπορούσε ισοδύναμα να είναι γραμμένη και ως εξής:

```
#include <stdio.h>
        void main()
{
printf("Hello World, Γεια σου Κόσμε!\n"); //σχολια
}
```

[1] Σε κάθε πρόγραμμα που θα υλοποιηθεί στα πλαίσια του εργαστηρίου ή γενικότερα στην προγραμματιστική γλώσσα C, επιβάλλεται η ύπαρξη της εντολής δήλωσης της main.

Αντίστοιχα, η παραπάνω δήλωση της main θα μπορούσε να γραφτεί ως εξής:

```
#include <stdio.h>
        main()
{
printf("Hello World, Γεια σου Κόσμε!\n"); //σχόλια
}
```

Η παραπάνω δήλωση και λειτουργία είναι ίδια με αυτή της void main () καθώς, αφού εκτυπώνεται ένα μήνυμα στη γραμμή εντολών, η συνάρτηση δεν επιστρέφει τιμή. Συνεπώς, παρατηρείται ότι σε αυτή τη περίπτωση το void δεν είναι απαραίτητο όμως υπονοείται.

3. Υλοποίηση αποτελέσματος οθόνης στη Γραμμή Εντολών (των Windows)

Γιατί επιλέγουμε το Dev-C++ ή κάποιο άλλο από τα προτεινόμενα προγράμματα για την ανάπτυξη εφαρμογών και όχι την γραμμή εντολών; Ενδεικτικά, εφόσον έχετε ακολουθήσει επακριβώς τα βήματα για την εγκατάσταση καθώς και υλοποίηση ενός φακέλου για την αποθήκευση των προγραμμάτων σας (workspace), τότε οι εντολές απαιτούν λίγα δευτερόλεπτα για να τρέξουν (interpret) στην οθόνη του υπολογιστή σας. Ωστόσο, αν έχετε υποπέσει σε σφάλμα, τότε το περιβάλλον του εργαλείου που έχετε εγκαταστήσει θα σας υποδείξει (μετά το απαραίτητο compile) το σημείο-γραμμή.

Έστω ότι επιθυμούμε να τρέξουμε ένα παρόμοιο με το παραπάνω πρόγραμμα, το οποίο θα εμφανίζει ως μήνυμα εξόδου: «hello world», μόνο με εντολές. Βασικές εντολές του Cmd είναι: help, dir[2], cd[3], copy, move, del, exit, mkdir, cls (βοήθεια, περιεχόμενα φακέλου, αλλαγή προορισμού, αντιγραφή, μετακίνηση (αποκοπή), διαγραφή, έξοδος, δημιουργία νέου φακέλου (προορισμού), καθαρισμός της γραμμής εντολών). Για να τρέξουμε το παραπάνω πρόγραμμα, απαιτούνται σύνθετα και χρονοβόρα βήματα, αφού πρώτα ανοίγουμε την γραμμή εντολών και πατάμε στην αναζήτηση των Windows την εντολή cmd.exe (Start->Run->cmd). Υπό αυτό το πρίσμα, καθίσταται σαφές ότι ο συγκεκριμένος τρόπος, αν και λειτουργικός, δεν είναι ιδιαίτερα εύχρηστος για τον προγραμματισμό.

[2] dir: directory, παραθέτει αναλυτικά όλα τα αρχεία και υποφακέλους, εντός του φακέλου που είμαστε.
[3] cd: change directory, άλλαξε το φάκελο (όπως κάνουμε διπλό κλικ).

4. Πρόβλημα Παρουσίασης Κώδικα στον Τερματικό - Γραμμή Εντολών

Αν κατά την εκτέλεση του παραπάνω παραδείγματος, κατόπιν επιτυχούς compile του .c αρχείου και της αντίστοιχης δημιουργίας του .exe αρχείου, η οθόνη του τερματικού εκτελείται αλλά κλείνει σε μικρό χρονικό διάστημα, δοκιμάστε τα παρακάτω:

```
#include <stdio.h>
        main()
{
printf("Hello World, Γεια σου Κόσμε!\n"); //σχόλια
 getchar();  /* εντολή η οποία απαιτεί είσοδο χρήστη */
/* η εντολή getchar() χρησιμοποιείται στο τερματικό εντολών(cmd) */
/* ώστε να αναμένεται είσοδος από το χρήστη οπότε, το cmd παραμένει ανοιχτό
μέχρι να δοθεί ένας χαρακτήρας από τον χρήστη (μέσω πληκτρολογίου+enter)  */
}
```

```
#include <stdio.h>
        main()
{
printf("Hello World, Γεια σου Κόσμε!\n"); //σχόλια
system("pause");  /* εντολή η οποία σταματάει την εκτέλεση */
/* η εντολή system("pause") χρησιμοποιείται για να σταματήσουμε την εκτέλεση */
/* στη γραμμή που εισάγεται η εντολή σταματάει προσωρινά η εκτέλεση μέχρώστε
να αναμένεται είσοδος από το χρήστη οπότε, το cmd παραμένει ανοιχτό μέχρι να
πατηθεί κάποιος χαρακτήρας από τον χρήστη (μέσω πληκτρολογίου)  */
}
```

5. Μεταβλητές/Τύποι Δεδομένων

Σε προγενέστερο κεφάλαιο, παρουσιάστηκε το πρόγραμμα εξαγωγής ενός μηνύματος στην οθόνη του χρήστη. Πέρα από την εμφάνιση μηνυμάτων στην οθόνη του χρήστη, υπάρχουν και οι μεταβλητές. Αυτές αποτελούν μεγέθη που δεσμεύουν χώρο στην μνήμη του υπολογιστή μας, για τις οποίες μπορούμε να δηλώσουμε μια συγκεκριμένη ποσότητα. Οι βασικότεροι τύποι δεδομένων είναι οι εξής:

1. char -> χαρακτήρας
   - πχ. char Letter;
     Letter = 'x';

2. int -> ακέραιος αριθμός
   - πχ. int x = 5;

3. float -> κινητής υποδιαστολής
   - πχ. float Miles;
     Miles = 5.6;

4. double -> διπλής ακρίβειας κινητής υποδιαστολής
   - πχ. double Atoms;
     Atoms = 2500000000;

Έμφαση απαιτείται να δοθεί στον όρο **<u>συγκεκριμένη</u>** ποσότητα, καθώς, αναλόγως με το είδος της μεταβλητής, δύνανται να αποθηκευτούν τα αντίστοιχα δεδομένα σε αυτήν. Σημειώνεται ότι **δεν επιτρέπεται** να αποθηκευτεί στοιχείο, το οποίο δεν είναι ίδιου τύπου με αυτό της μεταβλητής (π.χ. για μια μεταβλητή ακεραίου αριθμού, εισάγεται μια ακολουθία χαρακτήρων ή ένας αριθμός ακρίβειας κινητής υποδιαστολής), όπως γίνεται ε

6. Αριθμητικοί και Λογικοί Τελεστές

Έχοντας πλέον κατανοήσει την έννοια των τύπων δεδομένων μιας μεταβλητής καθώς και τη χρησιμότητά τους, σε αυτή την ενότητα παρουσιάζονται οι απαραίτητοι τελεστές, για την πραγματοποίηση αριθμητικών και λογικών πράξεων.

<u>Αριθμητικοί τελεστές</u>

- \+ πρόσθεση
- ++ πρόσθεση κατά 1
- \- αφαίρεση
- -- αφαίρεση κατά 1
- * πολλαπλασιασμός
- / διαίρεση
- % ακέραιο υπόλοιπο διαίρεσης

Παράδειγμα: w = ((x + 3)*(x - y))/10

<u>Λογικοί τελεστές</u>

- > μεγαλύτερο
- >= μεγαλύτερο ή ίσο
- < μικρότερο

- <= μικρότερο ή ίσο
- == ίσο
- != διάφορο
- ! αντίθετο (NOT)
- && λογικό AND
- || λογικό OR

Παράδειγμα: x > 5, !(x > 5) ισοδύναμο με x <= 5, y == x

7. Εντολές Εξόδου Δεδομένων

Με εφαλτήριο την εντολή για την εξαγωγή μηνυμάτων στην οθόνη του χρήστη printf("…"); σε αυτήν την ενότητα, παρουσιάζεται ο τρόπος με τον οποίο εισάγονται δεδομένα από τον χρήστη, για την επίτευξη σχέσης αλληλεπίδρασης χειριστή-κονσόλας.

Για την εισαγωγή δεδομένων από τον χρήστη, χρησιμοποιείται η εντολή:

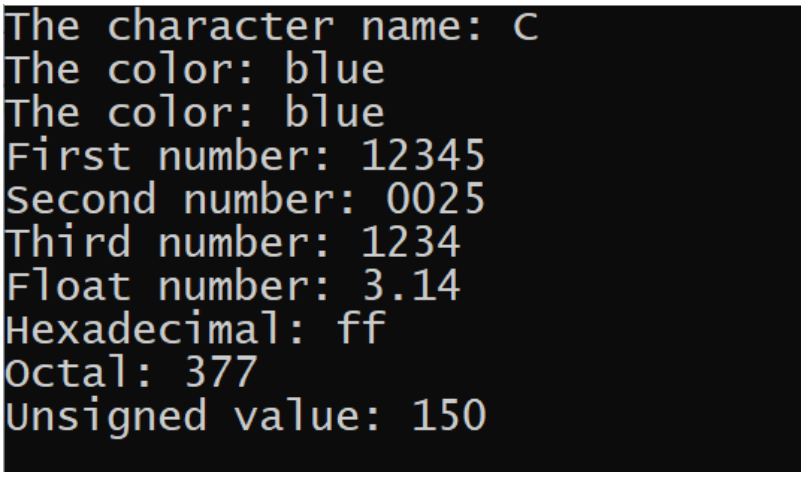
```
int printf(const char *format, ...)
```

Η χρήση της παραπάνω εντολής απαιτεί την ύπαρξη της βιβλιοθήκης *#include <stdio.h>*, επομένως ενεργοποιείται μετά τη δήλωση της συνάρτησης (τύπος_συνάρτησης main()). Στα πλαίσια του εργαστηρίου καθώς και μελλοντικά, στα προγράμματα που θα υλοποιήσετε, προτείνεται να υλοποιείται αυτόματα την διαδικασία, μέσω της δημιουργίας ενός νέου πηγαίου κώδικα στο εκάστοτε περιβάλλον, ώστε να μην δημιουργείται δυσχέρεια κατά την εύρεση και εισαγωγή της ορθής βιβλιοθήκης.

Σε ό,τι αφορά την χρήση της εντολής, ακολουθούν μερικά παραδείγματα:

```
#include<stdio.h>
main()
{
        printf("The character name: %c\n", 'C');      //αποτέλεσμα %s:    C
        printf("The color: %s\n", "blue");            //αποτέλεσμα %s:    blue
        printf("The color: %s\n", "blue");            //αποτέλεσμα %s:    blue
        printf("First number: %d\n", 12345);          //αποτέλεσμα %d:    12345
        printf("Second number: %04d\n", 25);          //αποτέλεσμα %04d: 0025
        printf("Third number: %i\n", 1234);           //αποτέλεσμα %i:    1234
        printf("Float number: %3.2f\n", 3.14159);     //αποτέλεσμα %3.2f: 3.14
        printf("Hexadecimal: %x\n", 255);             //αποτέλεσμα %x:    ff
        printf("Octal: %o\n", 255);                   //αποτέλεσμα %o:    377
        printf("Unsigned value: %u\n", 150);          //αποτέλεσμα %u:    150
}
```

Αποτελέσματα στην οθόνη του χρήστη:



Αναλυτικότερα, αναλόγως των στοιχείων που θέλουμε να εισάγει ο χρήστης στην κονσόλα, επιλέγουμε την εκάστοτε εντολή:

Format	**Τύπος μεταβλητής εισόδου**
%c	Character
%d or %i	Signed decimal integer
%e	Scientific notation (mantissa/exponent) using e character
%E	Scientific notation (mantissa/exponent) using E character
%f	Decimal floating point
%g	Uses the shorter of %e or %f

Τέλος, παρότι δόθηκε έμφαση στην εντολή του printf και στην ορθή εισαγωγή στοιχείων από το πληκτρολόγιο του χρήστη, επισημαίνουμε ότι, εφόσον επιθυμείτε να εισάγετε

άλλα εργαλεία (utilities), ακολουθείτε τον ίδιο τρόπο συμπλήρωσης. Ακολουθούν παραδείγματα μαθηματικών υπολογισμών.

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

int main(int argc, const char * argv[])
{
    /* Define temporary variables */
    double value1, value2;
    double result;

    /* Assign the values we will use for the pow calculation */
    value1 = 4;
    value2 = 2;

    /* Calculate the result of value1 raised to the power of value2 */
    result = pow(value1, value2);

    /* Display the result of the calculation */
    printf("%f raised to the power of %f is %f\n", value1, value2, result);

    return 0;
}/*Υπολογίζει το τετράγωνο του x, όπου x είναι double μεταβλητή*/
```

Αποτελέσματα στην οθόνη του χρήστη:

```
4.000000 raised to the power of 2.000000 is 16.000000
```

8. Μετατροπές σε Μεταβλητές/Τύπους Δεδομένων

Στην C έχουμε την δυνατότητα να μετατρέψουμε μια μεταβλητή σε μεταβλητή άλλου τύπου, με τη τεχνική του **casting** . Η σύνταξη της εντολής έχει την εξής μορφή:

Γενική μορφή: (*τύπος*) μεταβλητή

Ωστόσο, μέσω αυτής της τεχνικής, υπάρχει κίνδυνος να επέλθει απώλεια πληροφοριών. Ακολουθούν αναλυτικά παραδείγματα μετατροπής:

Παράδειγμα 1

```
#include <stdio.h>

main(){
        float x = 12.324;
        printf("PROSOXI stis metatropes dedomenwn! Alli timi exei to x:%f prin tim metatroph kai alli meta opou y: %d",x, (int)x );
```

```
}
```

Αποτελέσματα στην οθόνη του χρήστη:

```
PROSOXI stis metatropes dedomenwn! Alli timi exei to x:12.324000 prin tim metatroph kai alli meta opou y: 12
```

Παράδειγμα 2

```
#include <stdio.h>

main(){
        int x = 21;
        printf("PROSOXI stis metatropes dedomenwn! Alli timi exei to x:%d prin tim metatroph kai alli meta opou y: %f",x, (float)x );
}
```

Αποτελέσματα στην οθόνη του χρήστη:

```
PROSOXI stis metatropes dedomenwn! Alli timi exei to x:21 prin tim metatroph kai alli meta opou y: 21.000000
```

9. C: Case sensitive

Σε αυτό το σημείο και με αφορμή τα προηγούμενα παραδείγματα που παρουσιάστηκαν, κρίνεται αναγκαίο να επισημανθεί ότι η γλώσσα C είναι **case sensitive.** Ειδικότερα, αυτό σημαίνει ότι υπάρχει διάκριση μεταξύ των κεφαλαίων και των μικρών γραμμάτων, ανεξαρτήτως αν γράφουμε το πρόγραμμα με ελληνικούς ή λατινικούς χαρακτήρες. Για να κατανοήσουμε ορθότερα τον όρο, ας εξετάσουμε το εξής παράδειγμα:

```
#include <stdio.h>

main()
{
   int a=12;
   printf("I timi tou a einai: %d \n", a);
}
```

Αποτελέσματα στην οθόνη του χρήστη:

```
I timi tou a einai: 12
```

Στη συνέχεια, αντιγράφουμε το παραπάνω κώδικα και μεταβάλλουμε κάποια από τις παραπάνω εντολές, όπως:

int A=12; ως **Int** ή **iNt** ή **inT**

Παρατηρούμε ότι, εφόσον συντελέσουμε αυτή τη μεταβολή ή γενικότερα ανασυντάξουμε το κείμενο, μετατρέποντας ένα γράμμα από κεφαλαίο σε μικρό, το πρόγραμμα **δεν** έχει την επιθυμητή λειτουργία.

Παραδείγματα Εργαστηρίου

Έχοντας πλέον κατανοήσει βασικές έννοιες και όρους που αφορούν την γλώσσα προγραμματισμού C, θα πραγματοποιηθούν στο 1ο εργαστήριο του μαθήματος, τα ακόλουθα παραδείγματα.

Παράδειγμα 1

Να κατασκευάσετε ένα πρόγραμμα, το οποίο θα υπολογίζει το γινόμενο των ακεραίων αριθμών 12*53, 21*50.555.

Το πρόγραμμά σας επιβάλλεται:

- Να υλοποιεί τις παραπάνω πράξεις, μέσω της δημιουργίας και δήλωσης μιας μόνο συνάρτησης.
- Να μεταγλωττίζεται και να εκτελείται επιτυχώς.
- Να παράγει μηνύματα στην οθόνη του χρήστη (σε διαφορετικές γραμμές), σχετικά με την πράξη που υλοποιήθηκε και το τελικό αποτέλεσμα.

```
#include <stdio.h>

void main( )
{
   int metavliti1,metavliti2,metavliti3; /* δήλωση τύπου μεταβλητών 1,2,3 (integer) */
   metavliti1=12;                        /*  ανάθεση τιμής στην μεταβλητή 1 */
   metavliti2=53;                        /*  ανάθεση τιμής στην μεταβλητή 2 */
   metavliti3=21;                        /*  ανάθεση τιμής στην μεταβλητή 3 */
   float metavliti4;                     /*  δήλωση τύπου μεταβλητής 4 (float) */
   metavliti4=50.555;                    /*  ανάθεση τιμής στην μεταβλητή 4 */
   int ginomeno1;                        /*  δήλωση τύπου μεταβλητης ginomeno1 (int) */
   float ginomeno2;                      /*   δήλωση τύπου μεταβλητης ginomeno2 (float) */


  ginomeno1=metavliti1*metavliti2; /* υπολογισμός τιμής της μεταβλητή ginomeno1 */
  printf("%d", ginomeno1); /* εμφάνιση του αποτελέσματος σε μορφή ακέραιου αριθμού */

  printf("\n");                 /* μεταφορά στην επόμενη γραμμή */

   ginomeno2=metavliti3*metavliti4; /* υπολογισμός τιμής της μεταβλητή ginomeno2 */
   printf("%f", ginomeno2); /* εμφάνιση του αποτελέσματος σε με ακρίβεια δεκαδικών
αριθμών (float) */
}
```

Αποτελέσματα στην οθόνη του χρήστη:

```
636
1061.655029
```

Παράδειγμα 2

Να κατασκευάσετε ένα πρόγραμμα, το οποίο θα υπολογίζει το γινόμενο των ακεραίων αριθμών 12*53, καθώς και 21*50.

Το πρόγραμμά σας επιβάλλεται:

- Να υλοποιεί τις παραπάνω πράξεις, μέσω της δημιουργίας και δήλωσης μίας συνάρτησης, με την ονομασία mul.
- Να μεταγλωττίζεται και να εκτελείται επιτυχώς.
- Να παράγει μηνύματα στην οθόνη του χρήστη (σε διαφορετικές γραμμές), σχετικά με την πράξη που υλοποιήθηκε και το τελικό αποτέλεσμα.

```c
#include <stdio.h>

int mul(x,y)
int x,y;
{
   return(x*y); /* επιστρέφει το αποτέλεσμα του γινομένου */
}

main( )
{
   int x,y,j,k,p;
   x=12;
   y=153;
   p=mul (x,y) ; /* 1η κλήση της συνάρτησης */
   printf("%d", p); /* εμφάνιση του αποτελέσματος σε μορφή ακέραιου αριθμού */
   printf("\n"); /* μεταφορά στην επόμενη γραμμή */
   j=21;
   k=50;
   p=mul(k,j); /* 2η κλήση της συνάρτησης */
   printf("%d \n",p); /* εμφάνιση του αποτελέσματος σε μορφή ακέραιου */
}
```

Αποτελέσματα στην οθόνη του χρήστη:

```
1836
1050
```

Παράδειγμα 3

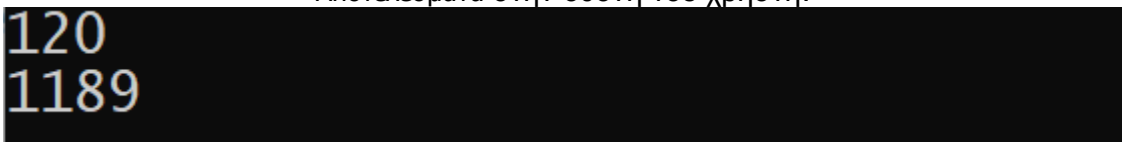
Να κατασκευάσετε ένα πρόγραμμα το οποίο θα υπολογίζει το άθροισμα των ακεραίων αριθμών 75+45, καθώς και τη διαφορά 1234-45.

Το πρόγραμμά σας επιβάλλεται:

- Να υλοποιεί τις παραπάνω πράξεις, μέσω της δημιουργίας και δήλωσης μιας μόνο συνάρτησης.
- Να μεταγλωττίζεται και να εκτελείται επιτυχώς.
- Να παράγει μηνύματα στην οθόνη του χρήστη (σε διαφορετικές γραμμές) σχετικά με την πράξη που υλοποιήθηκε και το τελικό αποτέλεσμα.

```
#include <stdio.h>

int plus(a,b)
int a,b;
{
  return(a+b);
}

int minus(a,b)
int a,b;
{
  return(a-b);
}

main()
{
        int x,y,j,k,p,s;
        x=75;
        y=45;
        p= plus (x,y) ;
        printf("%d", p);
        printf("\n");
        j=1234;
        k=45;
        s=minus(j,k);
        printf("%d",s);
}
```

Αποτελέσματα στην οθόνη του χρήστη:



Παράδειγμα 4

Να κατασκευάσετε ένα πρόγραμμα, το οποίο θα υπολογίζει το γινόμενο των ακεραίων αριθμών 12*33*7*24.

Το πρόγραμμά σας επιβάλλεται:

- Να υλοποιεί τις παραπάνω πράξεις, μέσω της δημιουργίας και δήλωσης μιας μόνο συνάρτησης.
- Να μεταγλωττίζεται και να εκτελείται επιτυχώς.
- Να παράγει μηνύματα στην οθόνη του χρήστη (σε διαφορετικές γραμμές), σχετικά με την πράξη που υλοποιήθηκε και το τελικό αποτέλεσμα.

```
#include <stdio.h>

int mul(a,b,c,d)
int a,b,c,d;
{
  return(a*b*c*d);
}

main()
{
        int w,x,y,z,p;
        w=12;
        x=33;
        y=7;
        z=24;
        p=mul (w,x,y,z) ;
        printf("%d\n", p);
}
```

Αποτελέσματα στην οθόνη του χρήστη:

Εργαστήριο 2 - Εντολές ελέγχου και Επανάληψης

1. Εντολές Λογικού Ελέγχου (ΑΝ)

Στις προηγούμενες ενότητες αναλύθηκαν κύριες έννοιες για την εμφάνιση δεδομένων, τη δήλωση και την χρησιμοποίηση μεταβλητών. Σε αυτό το κεφάλαιο, θα επεκταθούμε σε μια ιδιαίτερα σημαντική έννοια του προγραμματισμού, την εντολή ελέγχου ΑΝ (if).

Πιο αναλυτικά, έστω ότι θέλουμε να χρησιμοποιήσουμε ένα πλήθος εντολών, αποκλειστικά στις περιπτώσεις όπου ισχύει μια συνθήκη, λ.χ σε μια αριθμομηχανή εισάγουμε αριθμούς και, αναλόγως μιας συνθήκης, επιλέγουμε ποια πράξη θα υλοποιηθεί. Η γενική σύνταξη αυτής της εντολής είναι:

```
if(λογική συνθήκη) {
        //Εάν είναι αληθής η συνθήκη, πχ. x > 5
        ...
}else{
       //Εάν είναι ψευδής η συνθήκη
       ...
```

Αντίστοιχα, η παραλλαγή για πολλές υλοποιήσεις αναλόγως των συνθηκών είναι:

```
if(λογική συνθήκη) {
        //Εάν είναι αληθής η συνθήκη, πχ. x > 5
        ...
}else if(λογική συνθήκη){
       //Εάν είναι αληθής η συνθήκη. πχ. x == 0
       ...
}else{
       //Εάν δεν ισχύει κανένα από τα προηγούμενα
       ...
}
```

2. Εισαγωγή στις Δομές Επαναλήψεων

Για (for), Όσο (while), Εφόσον (DoWhile)

Η γενική σύνταξη της *for* είναι:

```
int i;
for(i=0; i<5; i=i+1){
        //εκτέλεσε των κώδικα 6 φορές (από 0-5)
      }
```

Η γενική σύνταξη της *while* είναι:

```
while (λογική συνθήκη) {
        //εκτέλεσε των κώδικα όσο η συνθήκη είναι αληθής
       }
```

Η γενική σύνταξη της *DoWhile* είναι:

```
do {
// εκτέλεσε των κώδικα όσο η συνθήκη είναι αληθής
// αλλά τουλάχιστον μία φορά γιατί ο έλεγχος γίνεται στο τέλος
     } while (λογική συνθήκη) ;
```

Οι τρείς παραπάνω υλοποιήσεις έχουν ταυτόσημη σημασία και χρήση. Ειδικότερα, θα αναλυθεί η εντολή *for* :

Πλήθος επανάληψης	**Τιμή i**	**Έλεγχος Συνθήκης ($i \leq 5$)**	**Εκτέλεση εντολών εντός { }**
1η	0	ΝΑΙ($0 \leq 5$)	ΝΑΙ
2η	0+1=1	ΝΑΙ ($1 \leq 5$)	ΝΑΙ
3η	1+1=2	ΝΑΙ ($2 \leq 5$)	ΝΑΙ
4η	2+1=3	ΝΑΙ ($3 \leq 5$)	ΝΑΙ
5η	3+1=4	ΝΑΙ ($4 \leq 5$)	ΝΑΙ
6η	4+1=5	ΝΑΙ ($5 \leq 5$)	ΝΑΙ
7η	5+1=6	ΟΧΙ ($6 \leq 5$)	ΟΧΙ (τέλος δομής επανάληψης)

Από το παραπάνω παράδειγμα, παρατηρούμε ότι κάθε δομή επανάληψης:

1. Έχει μια **αρχική τιμή** (αρχικοποίηση) για την μεταβλητή i, καθώς και ότι είναι *πάντα* ακέραιου τύπου.

2. Κάθε δομή επανάληψης απαιτεί ένα **βήμα αύξησης** (i=i+1), αλλιώς στο πίνακα δεν θα μπορούσε να προχωρήσει η επαναληπτική διαδικασία περαιτέρω της 1ης επανάληψης (θα ίσχυε πάντα $(0 \leq 5)$.
3. Έχει μια **λογική συνθήκη** που ελέγχει και καθορίζει το πλήθος των επαναλήψεων, συνεπώς και την συνθήκη τερματισμού της επαναληπτικής διαδικασίας.

Σε αυτό το σημείο, αξίζει να τονιστεί ότι τόσο στην *for,* όσο και στην *while*, η λογική συνθήκη ελέγχεται στην αρχή της υλοποίησης, συνεπώς δεν είναι απαραίτητο να επιτελεστεί έστω και μια επανάληψη. Αυτό δεν ισχύει για την εντολή ***DoWhile,*** επειδή, αφού η λογική συνθήκη ελέγχεται στο τέλος της υλοποίησης, ακόμα και αν δεν ισχύει η συνθήκη, *πάντα θα εκτελείται τουλάχιστον 1 φορά.*

Τέλος, σε περίπτωση που κατά το εργαστήριο δεν ορίσαμε ορθώς το βήμα αύξησης ή γενικότερα αντιμετωπίσαμε κάποια δυσχέρεια που οδηγεί στην μη παύση της επαναληπτικής διαδικασίας, το φαινόμενο ονομάζεται «ατέρμων βρόγχος». Το πρόγραμμά θα τρέχει διαρκώς ‘’κολλημένο’’ σε μια επανάληψη, μέχρις ότου να γεμίσει η μνήμη του υπολογιστή μας, όπου ακολουθεί αυτόματη επανεκκίνησή του.

Παράδειγμα 1

Να κατασκευάσετε ένα πρόγραμμα, το οποίο θα εμφανίζει, σε μια γραμμή της οθόνης, τους ακεραίους αριθμούς -5, -4, -3, -2,- 1, 0, 1, 2, 3, 4, 5.

Το πρόγραμμά σας επιβάλλεται:

- Να υλοποιεί τις παραπάνω διαδικασία με την βοήθεια μιας εντολής ανακύκλωσης (δομή επανάληψης).
- Να υλοποιεί τις παραπάνω διαδικασία με ΔΙΑΦΟΡΕΤΙΚΟΥΣ τρόπους (for, while, dowhile)
- Να μεταγλωττίζεται και να εκτελείται επιτυχώς.

Για την επίλυση της άσκησης θα χρησιμοποιηθεί η εντολή scanf η οποία δέχεται («σκανάρει») από την γραμμή εντολών (cmd) την είσοδο του χρήστη. Ενδεικτικά:

```
//αν θελω να εχω εισοδο integer (πχ.2,3,4,10,...)
int metavliti;
scanf("%d",&metavliti); //decimal

//αν θελω να εχω εισοδο float (πχ.2.3,3.5,4.55,10.2,...)
float metavliti2;
scanf("%f",&metavliti2); //float
```

```
#include <stdio.h>

main()
{
        int i;
```

```
        for(i=-5; i<=5; i=i+1)
        {
                printf("%d\t",i);
        }

        printf("\nAllos tropos(While)\n");
        i=-5;
        while(i<6)
        {
                printf("%d\t",i);
                i+=1;
        }
        printf("\nAllos tropos(DoWhile)\n");

        i=-5;

        do
        {
                printf("%d\t",i);
                i++;

        } while( i<=5) ;
}
```

Αποτελέσματα στην οθόνη του χρήστη:

```
-5      -4      -3      -2      -1      0       1       2       3       4       5
Allos tropos(While)
-5      -4      -3      -2      -1      0       1       2       3       4       5
Allos tropos(DoWhile)
-5      -4      -3      -2      -1      0       1       2       3       4       5
```

Παράδειγμα 2

Να κατασκευάσετε ένα πρόγραμμα, το οποίο :

- Να διαβάζει από το πληκτρολόγιο του χρήστη έναν ακέραιο ΘΕΤΙΚΟ αριθμό Χ, μικρότερο του 10.
- Να εμφανίζει στην οθόνη του χρήστη Χ φορές το μήνυμα "Hello Student!" .
- να ειδοποιεί τον χρήστη για την εκ νέου εισαγωγή στοιχείων, σε περίπτωση εισαγωγής εσφαλμένου αριθμού.
- Να μεταγλωττίζεται και να εκτελείται επιτυχώς.

```c
#include <stdio.h>

int main()
{
        int input_Xristi,flag,k;
        flag=0;
        printf("Parakalw eisagete mia timi\n");

        do{
           scanf("%d",&input_Xristi);

                if(input_Xristi>0 && input_Xristi<10)
                {
                        flag=1;
                        printf("Dekti i timi eisagwgis\n");
                }

                else
                {
                        printf("Parakalw eisagete NEA timi\n");
                }

        }while(flag==0);

        for(k=1;k<=input_Xristi;k++)
                printf("Hello Student!\t");

        printf("\n");
        return 0;
}
```

Αποτελέσματα στην οθόνη του χρήστη:

```
Parakalw eisagete mia timi
2020
Parakalw eisagete NEA timi
-1
Parakalw eisagete NEA timi
0
Parakalw eisagete NEA timi
10
Parakalw eisagete NEA timi
5
Dekti i timi eisagwgis
Hello Student!  Hello Student!  Hello Student!  Hello Student!  Hello Student!
```

Εργαστήριο 3 - Δομές Επαναλήψεων

Σε συνέχεια του προηγούμενου εργαστηρίου, θα εξεταστούν αναλυτικότερα έννοιες που αφορούν τις δομές επαναλήψεων. Τονίζεται ότι, όπως αναφέρθηκε εκτενώς στο εργαστήριο, τόσο στις εντολές ελέγχου όσο και επανάληψης, επειδή αυτές καθορίζουν πολλαπλές εντολές, με βάση μια συνθήκη, ορίζουμε την εντολή if,for,while ή dowhile και στην συνέχεια τοποθετούμε εντός δυο αγκυλών ({…}) τον προγραμματιστικό κώδικα. Σε περίπτωση, όμως, που απαιτείται μόνο η χρήση μιας εντολής, αυτό δεν είναι αναγκαίο. Αν οριστεί ορθώς μια από τις παραπάνω εντολές και δεν τοποθετηθούν οι απαραίτητες αγκύλες, τότε εκτελείται η αμέσως επόμενη γραμμή προγραμματιστικού κώδικα, ασχέτως της διαδικασίας που υλοποιεί (εκτύπωση τιμής, εισαγωγή τιμής από το χρήστη, ανάθεση μεταβλητής κτλ.).

Ακολουθεί παράδειγμα προγραμματιστικού κώδικα, για μια απλή δομή ελέγχου:

if (συνθηκη**)** **printf("X");** **printf("Y");**	**if (**συνθηκη**)** **{** **printf("X");** **}** **printf("Y");**

Στους παραπάνω προγραμματιστικούς κώδικες, ΚΑΙ ΣΤΙΣ 2 ΠΕΡΙΠΤΩΣΕΙΣ, εξάγεται το ίδιο αποτέλεσμα. Ειδικότερα:

- Το Χ εκτυπώνεται ΜΟΝΟ όταν ισχύει η συνθήκη.
- Το Υ εκτυπώνεται ανεξαρτήτως της συνθήκης.

Ακολουθεί παράδειγμα προγραμματιστικού κώδικα, για μια απλή δομή επανάληψης:

for(i=1;συνθηκη;i++) **printf("X");** **printf("Y");**	**for(i=1;συνθηκη;i++)** **{** **printf("X");** **}** **printf("Y");**

Στους παραπάνω προγραμματιστικούς κώδικες, ΚΑΙ ΣΤΙΣ 2 ΠΕΡΙΠΤΩΣΕΙΣ, εξάγεται το ίδιο αποτέλεσμα. Ειδικότερα:

- Το Χ εκτυπώνεται ΜΟΝΟ όταν ισχύει η συνθήκη, όσες φορές ορίζει η συνθήκη στην επανάληψη.
- Το Υ εκτυπώνεται, ανεξαρτήτως της επανάληψης, 1 μόνο φορά.

Παράδειγμα 1

Να κατασκευάσετε ένα πρόγραμμα, το οποίο :

1. Θα εμφανίζει σε μια γραμμή της οθόνης τους ακεραίους αριθμούς:
   5,8,…,299
2. Θα υλοποιεί τις παραπάνω διαδικασία με την βοήθεια μιας εντολής ανακύκλωσης (δομή επανάληψης)
3. Θα υλοποιεί τις παραπάνω διαδικασίες με τουλάχιστον 2 ΔΙΑΦΟΡΕΤΙΚΟΥΣ τρόπους (for / while / dowhile).

```c
#include <stdio.h>

int main(){
    system("chcp 1253"); //cmd εμφάνιση ελληνικών χαρακτήρων
    //5->300==>5,8,...
    int i;
    i=5;
    printf("\n===1ος τροπος(dowhile)====\n");
        do{
        printf("%d ",i);
        i=i+3;
    }while(i<=300);
    printf("\n===2ος τροπος(for)====\n");
    for (i=5;i<=300;i=i+3){
        printf("%d ",i);
    }
    printf("\n===3ος τροπος(while)====\n");
    i=5;
    while(i<=300){
        printf("%d ",i);

        i=i+3;
    }
    return 0;
//  system("pause");
}
```

Αποτελέσματα στην οθόνη του χρήστη:

```
Active code page: 1253

===1ος τροπος(dowhile)====
5 8 11 14 17 20 23 26 29 32 35 38 41 44 47 50 53 56 59 62 65 68 71 74 77
80 83 86 89 92 95 98 101 104 107 110 113 116 119 122 125 128 131 134 137
140 143 146 149 152 155 158 161 164 167 170 173 176 179 182 185 188 191 1
94 197 200 203 206 209 212 215 218 221 224 227 230 233 236 239 242 245 24
8 251 254 257 260 263 266 269 272 275 278 281 284 287 290 293 296 299
===2ος τροπος(for)====
5 8 11 14 17 20 23 26 29 32 35 38 41 44 47 50 53 56 59 62 65 68 71 74 77
80 83 86 89 92 95 98 101 104 107 110 113 116 119 122 125 128 131 134 137
140 143 146 149 152 155 158 161 164 167 170 173 176 179 182 185 188 191 1
94 197 200 203 206 209 212 215 218 221 224 227 230 233 236 239 242 245 24
8 251 254 257 260 263 266 269 272 275 278 281 284 287 290 293 296 299
===3ος τροπος(while)====
5 8 11 14 17 20 23 26 29 32 35 38 41 44 47 50 53 56 59 62 65 68 71 74 77
80 83 86 89 92 95 98 101 104 107 110 113 116 119 122 125 128 131 134 137
140 143 146 149 152 155 158 161 164 167 170 173 176 179 182 185 188 191 1
94 197 200 203 206 209 212 215 218 221 224 227 230 233 236 239 242 245 24
8 251 254 257 260 263 266 269 272 275 278 281 284 287 290 293 296 299
```

Παράδειγμα 2

Να κατασκευάσετε ένα πρόγραμμα, το οποίο :

1. Θα εμφανίζει σε μια γραμμή της οθόνης τους ακεραίους αριθμούς:
   -32,-31,…,34,35,36
2. Θα υλοποιεί τις παραπάνω διαδικασία με την βοήθεια μιας εντολής ανακύκλωσης (δομή επανάληψης)
3. Θα υλοποιεί τις παραπάνω διαδικασίες με τουλάχιστον 2 ΔΙΑΦΟΡΕΤΙΚΟΥΣ τρόπους (for / while / dowhile).
4. Θα παραθέτει τα αποτελέσματα στην οθόνη του χρήστη με 3 ή 6 αριθμούς ανά γραμμή.

```
#include <stdio.h>

int main(){
    system("chcp 1253"); //cmd εμφάνιση ελληνικών χαρακτήρων
    int i;
    int c=0; //μετρητης επαναληψεων
    printf("\nA τροπος υλοποιηση με χρηση for(για)\n");
    for (i=-32;i<=36;i++)
    {
        printf("%d  ",i); // εκτυπωση αποτελεσματων στην οθονη του
χρηστη
        if (i%6==0){
            printf("\n"); //ανα 6 εμφανισεις αριθμων η εκτυπωση-
printf- θα γίνεται στην επομενη γραμμη
        }
    }

    printf("\nB τροπος υλοποιηση με χρηση dowhile(εφοσον)\n");
    i=-32;
    do{
        c++;
        printf("%d  ",i); // εκτυπωση αποτελεσματων στην οθονη του
χρηστη
        if (c%6==0 && c>=6){
            printf("\n"); //ανα 6 εμφανισεις αριθμων η εκτυπωση-
printf- θα γίνεται στην επομενη γραμμη

        }
        i++;
    }while(i<=36); // προσοχη τρεχει ΤΟΥΛΑΧΙΣΤΟΝ 1 φορα καθως ο
ελεγχος της συνθηκης πραγματοποιειται στο τελος!

    printf("\nΓ τροπος υλοποιηση με χρηση while(οσο)\n");
    i=-32;
    while(i<=36){
        printf("%d  ",i); // εκτυπωση αποτελεσματων στην οθονη του
χρηστη
        if (i%6==0){
            printf("\n"); //ανα 6 εμφανισεις αριθμων η εκτυπωση-
printf- θα γίνεται στην επομενη γραμμη
        }
        i++;
```

```
    }


    return 0;
    //system("pause");

}
```

Αποτελέσματα στην οθόνη του χρήστη:

```
Active code page: 1253

Α τροπος υλοποιηση με χρηση for(για)
-32  -31  -30  -29  -28  -27
-26  -25  -24  -23  -22  -21
-20  -19  -18  -17  -16  -15
-14  -13  -12  -11  -10  -9
-8  -7  -6  -5  -4  -3
-2  -1  0  1  2  3
4  5  6  7  8  9
10  11  12  13  14  15
16  17  18  19  20  21
22  23  24  25  26  27
28  29  30  31  32  33
34  35  36
Β τροπος υλοποιηση με χρηση dowhile(εφοσον)
-32  -31  -30  -29  -28  -27
-26  -25  -24  -23  -22  -21
-20  -19  -18  -17  -16  -15
-14  -13  -12  -11  -10  -9
-8  -7  -6  -5  -4  -3
-2  -1  0  1  2  3
4  5  6  7  8  9
10  11  12  13  14  15
16  17  18  19  20  21
22  23  24  25  26  27
28  29  30  31  32  33
34  35  36
Γ τροπος υλοποιηση με χρηση while(οσο)
-32  -31  -30  -29  -28  -27
-26  -25  -24  -23  -22  -21
-20  -19  -18  -17  -16  -15
-14  -13  -12  -11  -10  -9
-8  -7  -6  -5  -4  -3
-2  -1  0  1  2  3
4  5  6  7  8  9
10  11  12  13  14  15
16  17  18  19  20  21
22  23  24  25  26  27
28  29  30  31  32  33
34  35  36
--------------------------------
Process exited after 0.1698 seconds with return value 0
Press any key to continue . . .
```

Παράδειγμα 3

Να κατασκευάσετε ένα πρόγραμμα, το οποίο :

1. θα δέχεται έναν ακέραιο αριθμό από τον χρήστη και θα ελέγχει αν βρίσκεται μεταξύ 0 και 10.
   Θα υλοποιεί τις παραπάνω διαδικασία με την βοήθεια μιας εντολής ανακύκλωσης (δομή επανάληψης)
2. Θα αναγράφει στην οθόνη του χρήστη τα απαραίτητα μηνύματα :
   a. Εισαγωγής δεδομένων από το πληκτρολόγιο
   b. Ειδοποίηση εισαγωγής ορθής τη εσφαλμένης τιμής

```c
#include <stdio.h>

int main(){
    system("chcp 1253"); //cmd εμφάνιση ελληνικών χαρακτήρων
    int password;
    int flag=1;
    do{
    printf("Δωσε μια ακέραια τιμη:\n");
    scanf("%d",&password);
    if (password>=0 && password<=10){
        printf("Σωστη τιμη\n");
        flag=0;
    }
    else{
        printf("Εσφαλμενη η τιμη\n");
    }
    }while(flag==1);

    return 0;
}
```

Αποτελέσματα στην οθόνη του χρήστη:

```
Active code page: 1253
Δωσε μια ακέραια τιμη:
-1
Εσφαλμενη η τιμη
Δωσε μια ακέραια τιμη:
111
Εσφαλμενη η τιμη
Δωσε μια ακέραια τιμη:
20
Εσφαλμενη η τιμη
Δωσε μια ακέραια τιμη:
11
Εσφαλμενη η τιμη
Δωσε μια ακέραια τιμη:
10
Σωστη τιμη
```

Παράδειγμα 4

Να κατασκευάσετε ένα πρόγραμμα, το οποίο :

1. Θα διαβάζει από το πληκτρολόγιο του χρήστη ένα αριθμό σε δυαδική μορφή, με ακριβώς δέκα (10) δυαδικά ψηφία (0 ή 1).
2. Θα υπολογίζει και θα εμφανίζει στην οθόνη τον ισοδύναμο αριθμό στο δεκαδικό σύστημα αρίθμησης.
3. Θα ειδοποιεί τον χρήστη για την εκ νέου εισαγωγή στοιχείων, σε περίπτωση εισαγωγής εσφαλμένου αριθμού.
4. Θα μεταγλωττίζεται και θα εκτελείται επιτυχώς.

Προσοχή: Δεν μπορείτε να χρησιμοποιήσετε την έτοιμη συνάρτηση βιβλιοθήκης strtol( ) για την μετατροπή αυτή.

Για τον έλεγχο του προγράμματος, η τιμή $(1010101011)_2$ στο δυαδικό σύστημα, θα πρέπει να μετατραπεί στην τιμή $(683)_{10}$ στο δεκαδικό σύστημα αρίθμησης.

```
#include <stdio.h>
main()
{
        int i=0,digits = 0;
        char c;

        printf("Eisigage ton arithmo se diadiki morfi:");

        while((c=getchar()) =='1'|| c == '0'){
                i= i*2;
                if (c=='1')
                i+=1;
                digits++;
        }

        if (digits !=10)
                printf("Prepei na eisageis akribws 10 psifia !\n");
        else
                printf("O arithmos einai: %d\n",i);
}
```

Αποτελέσματα στην οθόνη του χρήστη:

```
Eisigage ton arithmo se diadiki morfi:21
Prepei na eisageis akribws 10 psifia !
```

Αποτελέσματα στην οθόνη του χρήστη:

```
Eisigage ton arithmo se diadiki morfi:1010101011
O arithmos einai: 683
```

Παράδειγμα 5

Να κατασκευάσετε ένα πρόγραμμα, το οποίο θα ελέγχει αν ένας αριθμός είναι ΤΕΛΕΙΟΣ:

1. Θα διαβάζει από το πληκτρολόγιο του χρήστη έναν ΑΚΕΡΑΙΟ ΘΕΤΙΚΟ αριθμό, μικρότερο του 1000.
2. Θα υπολογίζει και θα εμφανίζει στην οθόνη μήνυμα, σχετικά με το αν είναι τέλειος αριθμός ή όχι.
3. Θα τερματίζεται η υλοποίηση του προγράμματος, σε περίπτωση εισαγωγής εσφαλμένου αριθμού.
4. Θα μεταγλωττίζεται και θα εκτελείται επιτυχώς.

Προσοχή: Ένας ακέραιος αριθμός λέγεται τέλειος αριθμός, όταν οι παράγοντές του συμπεριλαμβανομένης και της μονάδας (αλλά όχι και του ίδιου του αριθμού), δίνουν ως άθροισμα τον ίδιο αριθμό.

Για παράδειγμα, το 6 είναι ένας τέλειος αριθμός, επειδή 6=1+2+3 και άλλοι τέλειοι αριθμοί είναι: 28, 496, 8128, …

```c
#include <stdio.h>
main()
{
	int n,i=1,sum=0;
	do{
		printf("Eisagete THETIKI timi mikroteri tou 1000:");
		scanf("%d",&n);
	}while(n<0||n>1000);

	while(i<n)
	{
		if(n%i==0)
			sum=sum+i;
		i++;
	}

	if(sum==n)
		printf("To %d einai telios arithmos\n",i);
	else
		printf("To %d den einai telios arithmos\n",i);

	return 0;}
```

Αποτελέσματα στην οθόνη του χρήστη:

```
Eisagete THETIKI timi mikroteri tou 1000: 21
To 21 den einai telios arithmos
```

Αποτελέσματα στην οθόνη του χρήστη:

```
Eisagete THETIKI timi mikroteri tou 1000:6
To 6 einai telios arithmos
```

Εργαστήριο 4 - Συναρτήσεις

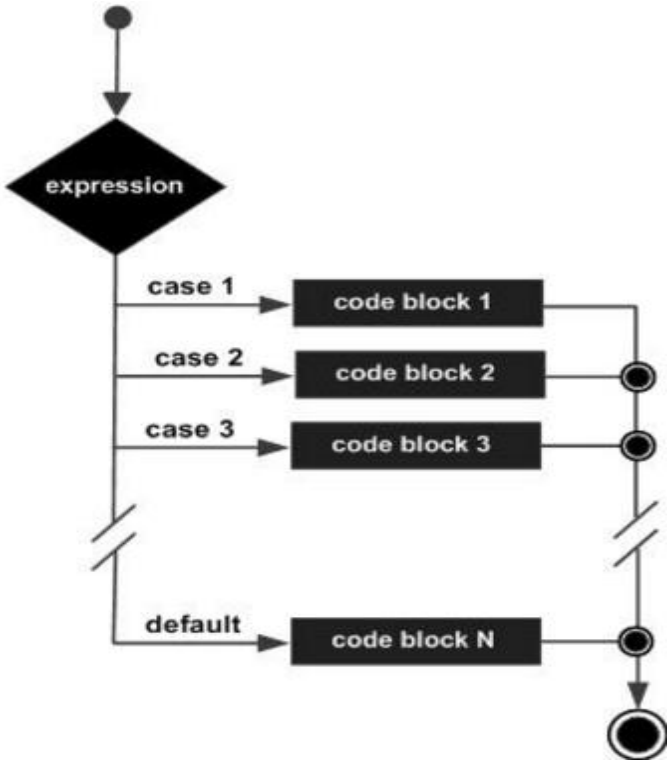
Στόχος του 4ου εργαστήριου είναι η περαιτέρω εξοικείωση με βασικές έννοιες του προγραμματισμού και ειδικότερα των συναρτήσεων.

Μια από τις εντολές που θα χρησιμοποιηθούν στο εργαστήριο αποτελεί η «Δομή Πολλαπλής επιλογής (swich)» για τον έλεγχο της ροής του προγράμματος.

Ο κώδικας υλοποίησης έχει την εξής μορφή:

```
switch(μεταβλητή ελέγχου)
{
case: (Τιμή_μεταβλητής_ελέγχου_1) :
//εκτέλεσε τον κώδικα
break;
case: (Τιμή_μεταβλητής_ελέγχου_2) :
//εκτέλεσε τον κώδικα
break;
default:
//εκτέλεσε τον κώδικα
break;
}
```

Το διάγραμμα λειτουργίας της είναι το εξής:



Εικόνα 14 *Διάγραμμα ροής δομής πολλαπλής επανάληψης (switch)* [4]

[4] Πηγή: https://www.tutorialspoint.com/cprogramming/switch_statement_in_c.htm

Παράδειγμα 1

Να κατασκευάσετε ένα πρόγραμμα, το οποίο :

1. Θα υπολογίζει το άθροισμα των αριθμών: 75+45
2. Θα υπολογίζει την αφαίρεση των αριθμών: 1234+45.986

Οι παραπάνω διαδικασίες θα επιτελούνται μέσω της κλήσης συναρτήσεων (μια για κάθε λειτουργία) οι οποίες θα έχουν την παρακάτω μορφή:

- Συναρτηση1: [εισοδος→εξοδος]=[int,int→int]
- Συναρτηση2: [εισοδος→εξοδος]=[int,float→float]

```
#include <stdio.h>

int add (arithmos1, arithmos2)
int arithmos1;
int arithmos2;
{
      return(arithmos1+arithmos2);
}


float sub (arithmos1,arithmos2)
int arithmos1;
float arithmos2;
{
      return(arithmos1-arithmos2);
}

main()
{
//----Ερωτημα 1
      int x,y,prosthesi;
      x=75;
      y=45;
      prosthesi=x+y;
      printf("Arithmos a= %d ",x);
      printf("\n"); // επομενη γραμμη
      printf("Arithmos b= %d ",y);
      printf("\n"); // επομενη γραμμη
      printf("Prothesi= %d ",prosthesi); //%d:decimal-->int
      printf("\n"); // επομενη γραμμη
      printf("Ara %d+%d=  %d \n",x,y,prosthesi);

//----Μερος 2
      int z=1234;//δηλωση μεταβλητης και αναθεση τιμης στην μεταβλητη
      float k=45.986;
      float praxi;
      praxi=z-k;

      printf("\n Afairesi integer(z)-float(k)= %f ",praxi);

//----Μερος 1-συναρτηση
       int apotelesmasinartisis;
       apotelesmasinartisis= add(x,y);
```

```
       printf("\n\n\n\nApotelesma sinartisis add= %d
",apotelesmasinartisis);

//----Μερος 2-συναρτηση
      float apotelesmasinartisis2;
      apotelesmasinartisis2= sub(z,k);
      printf("\nApotelesma sinartisis sub= %f
",apotelesmasinartisis2);}
```

Αποτελέσματα στην οθόνη του χρήστη:

```
Arithmos a= 75
Arithmos b= 45
Prothesi= 120
Ara 75+45=  120

 Afairesi integer(z)-float(k)= 1188.014038
 Afairesi integer(z)-float(k)= 1188



Apotelesma sinartisis add= 120
Apotelesma sinartisis sub= 1188.014038
```

Αναφορικά με το 2ο μέρος τονίζεται η παρακάτω απορία φοιτητή:

«Τι θα συμβεί αν αποθηκεύσουμε το αποτέλεσμα της αφαίρεσης int-float σε float μεταβλητή ;»

→Η απάντηση είναι ότι το πρόγραμμα θα γίνει επιτυχώς compile αλλά, το αποτέλεσμα δεν θα είναι ορθό καθώς θα έχει χαθεί πληροφορία (οι int δεν έχουμε δεκαδικά ψηφία είναι ακέραιοι αριθμοί).

Ειδικότερα, για να γίνει η σύγκριση με την τιμή της μεταβλητής: float praxi, εκτελέστε το παρακάτω αντίστοιχο κώδικα:

```
#include <stdio.h>
main()
{
//----Μερος 2 - Β τρόπος
int z=1234;//δηλωση μεταβλητης και αναθεση τιμης στην μεταβλητη
float k=45.986;
float praxi;
int erwtisifoiti;
erwtisifoiti=z-k;
printf("\n Afairesi integer(z)-float(k)= %d ",erwtisifoiti);
}
```

Τα αποτελέσματα της εκτέλεσης στη γραμμή εντολών είναι τα παρακάτω:

```
Afairesi integer(z)-float(k)= 1188
```

Παράδειγμα 2

Να κατασκευάσετε ένα πρόγραμμα, το οποίο :

1. Θα δέχεται από τον χρήστη 3 (έστω μη μηδενικές) τιμές.
2. Θα επιτελεί τις εξής λειτουργείες και θα εμφανίζει το αποτέλεσμα των πράξεων στην οθόνη του χρήστη:
    a. πρόσθεση 3ων αριθμών
    b. αφαίρεση 3ων αριθμών
    c. πολ/σμό 3ων αριθμών
    d. διαίρεση 3ων αριθμών

Οι παραπάνω διαδικασίες θα επιτελούνται μέσω της κλήσης συναρτήσεων (μια για κάθε λειτουργία).

```
#include <stdio.h>

int athroisma(int x,int y,int z)
{
    return (x+y+z);
}
int polsmo(int x1,int x2,int x3)
{
    return (x1*x2*x3);
}

int afairesi(int a,int b,int c)
{
    return (a-b-c);
}

float div(float a,float b, float c)
{
    return(a/b/c);
}
int main()
{
    system("chcp 1253"); //cmd εμφάνιση ελληνικών χαρακτήρων
    float function2;
    int a,b,c,function; //α τροπος δηλωση μεταβλητων
    //int a; //β τροπος δηλωσης μεταβλητων
    //int b; //β τροπος δηλωσης μεταβλητων
    //int c; //β τροπος δηλωσης μεταβλητων

    printf("Εισηγαγε τον 1ο αριθμο\n");
    scanf("%d",&a);// προοσοχη, κατα την δηλωση int %d εντός "" της
scanf (όπως και κατά την εκτύπωση-printf)

    printf("Eisigage ton 2o arithmo\n");
    scanf("%d",&b);

    printf("Εισηγαγε τον 3ο αριθμο\n");
    scanf("%d",&c);

    function =athroisma(a,b,c);
    printf("\nΑθροισμα=%d",function);

    function =afairesi(a,b,c);
```

```
    printf("\nΑφαιρεση=%d",function);

    function =polsmo(a,b,c);
    printf("\nΠολ_σμος=%d\n",function);
//  printf("Πολ/σμος=%d",polsmo(a,b,c));

    printf("Εισηγαγε 3 ακεραίους!\n");
    float q,w,e;

    printf("Εισηγαγε τον 1ο (δεκαδικό) αριθμο\n");
    scanf("%f",&q);
// προοσοχη, κατα την δηλωση float %f εντός ""
//της scanf(όπως και κατά την εκτύπωση-printf)

    printf("Εισηγαγε τον 2ο (δεκαδικό) αριθμο\n");
    scanf("%f",&w);

    printf("Εισηγαγε τον 3ο (δεκαδικό) αριθμο\n");
    scanf("%f",&e);

    function2 =div(a,b,c);

    printf("\nΔιαιρεση =%.2f\n",function2);
// εμφανιση αποτελεσματων για float με 2 δεκαδικα ψηφια



    return 0;
    system("pause");
}
```

Αποτελέσματα στην οθόνη του χρήστη:

```
Active code page: 1253
Εισηγαγε τον 1ο αριθμο
1
Eisigage ton 2o arithmo
2
Εισηγαγε τον 3ο αριθμο
3

Αθροισμα=6
Αφαιρεση=-4
Πολ_σμος=6
Εισηγαγε 3 ακεραίους!
Εισηγαγε τον 1ο (δεκαδικό) αριθμο
32
Εισηγαγε τον 2ο (δεκαδικό) αριθμο
55
Εισηγαγε τον 3ο (δεκαδικό) αριθμο
689

Διαιρεση =0.17
```

Παράδειγμα 3

Να κατασκευάσετε ένα πρόγραμμα, το οποίο :

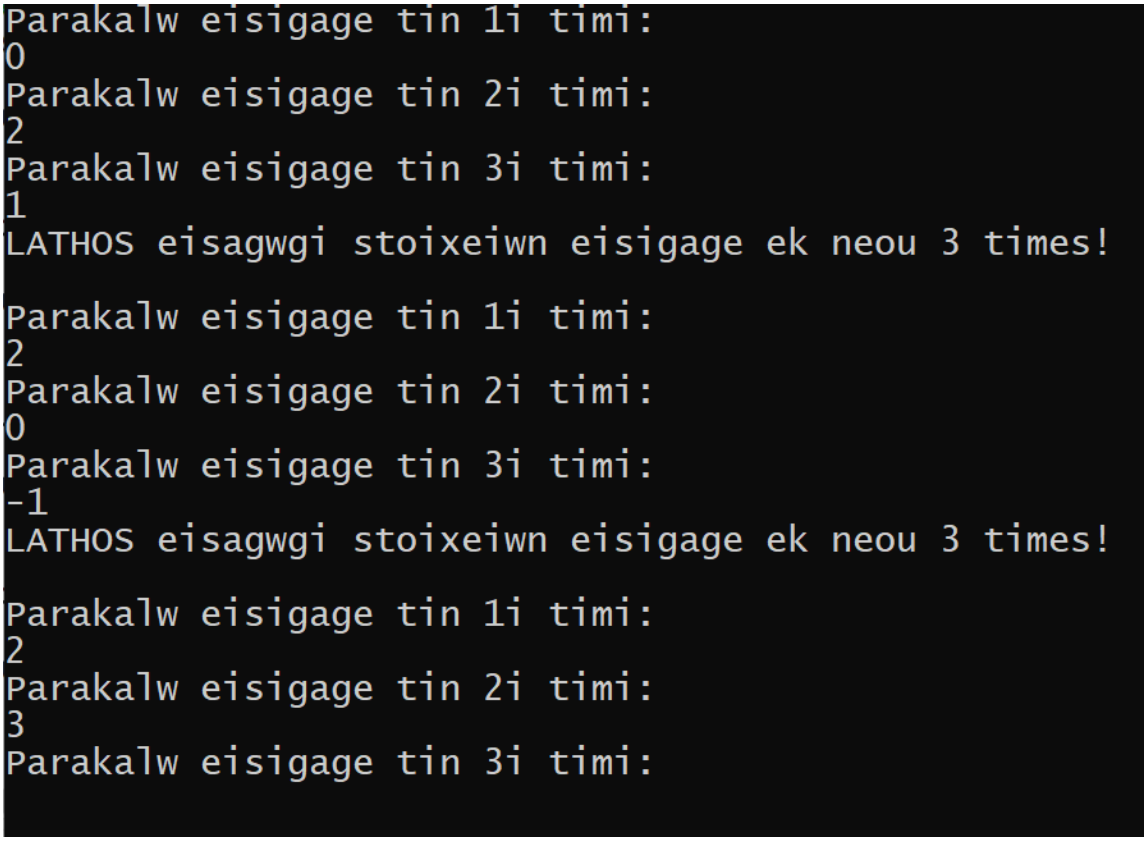
1. Θα δέχεται από τον χρήστη 3 ΘΕΤΙΚΕΣ ΑΚΕΡΑΙΕΣ τιμές
2. Θα εμφανίζει σχετικό μήνυμα και θα ειδοποιεί τον χρήστη για την εκ νέου εισαγωγή στοιχείων, σε περίπτωση εισαγωγής εσφαλμένου αριθμού (0 ή κάποιον αρνητικό)
3. Θα ρωτάει τον χρήστη ποια πράξη να επιτελέσει, μέσω της εισαγωγής ενός ακεραίου από το 1 έως το 4 και θα επιτελεί:
   A. πρόσθεση, εμφανίζοντας το αποτέλεσμα, αν επιλεγεί το 1,
   B. αφαίρεση, εμφανίζοντας το αποτέλεσμα, αν επιλεγεί το 2,
   C. πολλαπλασιασμό, εμφανίζοντας το αποτέλεσμα, αν επιλεγεί το 3,
   D. διαίρεση , εμφανίζοντας το αποτέλεσμα, αν επιλεγεί το 4.

Οι παραπάνω διαδικασίες θα επιτελούνται μέσω της κλήσης συναρτήσεων (μια για κάθε λειτουργία).

```
#include <stdio.h>
#include <math.h>
int prosthesi(int x,int y,int z)
{
      return(x+y+z);
}

int afairesi(int x,int y,int z)
{
      return(x-y-z);
}

int pol_smos(int x,int y,int z)
{
      return(x*y*z);

}
float diairesi(float x,float y,float z)
{
      return((float)(x/y/z));
}

main()
{
      int flag=0, flag2=0;
      int input1,input2,input3,input4,K;

      while(flag==0){
            printf("Parakalw eisigage tin 1i timi: \n");
            scanf("%d",&input1);
            printf("Parakalw eisigage tin 2i timi: \n");
            scanf("%d",&input2);
            printf("Parakalw eisigage tin 3i timi: \n");
            scanf("%d",&input3);

            if (input1<=0 ||input2<=0 ||input3<=0){
```

```
                        printf("LATHOS eisagwgi stoixeiwn eisigage ek
neou 3 times! \n\n");
            }
            else{
                    printf("ORTHI eisagwgi stoixeiwn !\n\n");
                    flag=1;
            }
      }

      do{
            printf("Epelexe praxi:\n 1.\t Gia Prosthesi\n 2.\t Gia
Afairesi\n 3.\t Gia Pol/smo\n 4.\t Gia Diairesi\n 5.\t Gia termatismo
\n");
            // \n: epomeni grammi, \t: stin idia grammi perissoero
keno metaxi twn xaraktirwn
             scanf("%d",&input4);
            switch(input4){
                  case 1:
                        printf ("Epelexes prosthesi twn 3wn
arithmwn\n");
                        K=prosthesi(input1,input2,input3);
                        printf ("I prosthesi twn
%d+%d+%d=%d\n",input1,input2,input3,K);

                  break;

                  case 2:
                        printf ("Epelexes afairesi twn 3wn
arithmwn\n");
                        K=afairesi(input1,input2,input3);
                        printf ("I afairesi twn %d-%d-
%d=%d\n",input1,input2,input3,K);


                  break;

                  case 3:
                        printf ("Epelexes polaplasiasmo twn 3wn
arithmwn\n");
                        K=pol_smos(input1,input2,input3);
                        printf ("O polaplasiasmos twn
%d*%d*%d=%d\n",input1,input2,input3,K);

                  break;

                  case 4:
                        printf ("Epelexes diairesi twn 3wn
arithmwn\n");
                        float K1=0; //dilwsi neou K kathws diairesi
paragei FLOAT arithmus
                        K1=diairesi(input1,input2,input3);
                        printf ("I diairesi twn
%d/%d/%d=%f\n",input1,input2,input3,K1);
                  break;
                  case 5:
                        printf("Telos programmatos \n");
                        flag2=1;
                  break;

                  default:   // gia kathe alli eisagwgi
                  printf("Lathos Epilogi \n");
```

```
                    break;
                }
        }while(flag2==0);
}
```

Εξεταζόμενη Περίπτωση	Αποτελέσματα στην οθόνη του χρήστη
Εφαλμένη εισαγωγή στοιχείων εισόδου	

Επιλογής Πρόσθεσης(1) (ορθή εισαγωγή στοιχείων)	Parakalw eisigage tin 1i timi: 10 Parakalw eisigage tin 2i timi: 20 Parakalw eisigage tin 3i timi: 70 ORTHI eisagwgi stoixeiwn ! Epelexe praxi: 1. Gia Prosthesi 2. Gia Afairesi 3. Gia Pol/smo 4. Gia Diairesi 5. Gia termatismo 1 Epelexes prosthesi twn 3wn arithmwn I prosthesi twn 10+20+70=100
Επιλογής Αφαίρεσης (2) (ορθή εισαγωγή στοιχείων)	Parakalw eisigage tin 1i timi: 12 Parakalw eisigage tin 2i timi: 2312 Parakalw eisigage tin 3i timi: 12354 ORTHI eisagwgi stoixeiwn ! Epelexe praxi: 1. Gia Prosthesi 2. Gia Afairesi 3. Gia Pol/smo 4. Gia Diairesi 5. Gia termatismo 2 Epelexes afairesi twn 3wn arithmwn I afairesi twn 12-2312-12354=-14654

Επιλογής Πολ/σμου (3) (ορθή εισαγωγή στοιχείων)	Parakalw eisigage tin 1i timi: 1 Parakalw eisigage tin 2i timi: 5 Parakalw eisigage tin 3i timi: 7 ORTHI eisagwgi stoixeiwn ! Epelexe praxi: 1. Gia Prosthesi 2. Gia Afairesi 3. Gia Pol/smo 4. Gia Diairesi 5. Gia termatismo 3 Epelexes polaplasiasmo twn 3wn arithmwn O polaplasiasmos twn 1*5*7=35
Επιλογής Διαίρεσης (4) (ορθή εισαγωγή στοιχείων)	Parakalw eisigage tin 1i timi: 3 Parakalw eisigage tin 2i timi: 2 Parakalw eisigage tin 3i timi: 12 ORTHI eisagwgi stoixeiwn ! Epelexe praxi: 1. Gia Prosthesi 2. Gia Afairesi 3. Gia Pol/smo 4. Gia Diairesi 5. Gia termatismo 4 Epelexes diairesi twn 3wn arithmwn I diairesi twn 3/2/12=0.125000

Παράδειγμα 4

Να κατασκευάσετε ένα πρόγραμμα το οποίο θα προσομοιώνει την χρήση ενός ATM. Ειδικότερα, θα εκτελεί τα παρακάτω :

1. Θα ζητάει συνεχώς από το χρήστη την εισαγωγή του κωδικού του μέχρι να εισαχθεί η ορθή τιμή (συνάρτηση elegxos)
2. Ο χρήστης θα επιλέγει αν θα κάνει κατάθεση ή ανάληψη χρηματικού ποσού (συνάρτηση katathesi,analipsi)
3. Θα δέχεται από το χρήστη έναν δεκαδικό αριθμό (float) που θα αντιστοιχεί στο συνολικό ποσό ανάληψης/κατάθεσης.

- Θεωρήστε ότι ο σωστός κωδικός είναι ο αριθμός: 1234
- Θεωρήστε ότι το συνολικό ποσό στο λογαριασμό του χρήστη είναι: 1000 $
- Θεωρήστε ότι ο χρήστης εισάγει από το πληκτρολόγιο μόνο αριθμούς .

```c
#include <stdio.h>
void elegxos();
float katathesi(float euros);
float analipsi(float euros);
void epilogi();

int main(){
    system("chcp 1253");//cmd εμφάνιση ελληνικών χαρακτήρων
    elegxos();
    epilogi();
    return 0; //system("pause");
}

void elegxos(){
    int password=1234;
    int input,flag=1;
    do{
     printf("Εισηγαγε το κωδικο\n");
     scanf("%d",&input);
     if(input==password){
        printf("Σωστος κωδικος\n");
        flag=0;}
     else{printf("Ξαναπροσπαθησε\n");}
      }while(flag==1);
}

float katathesi(float euros){
    float euroskat;
    printf("Δωσε ποσο καταθεσης:\n");
    scanf("%f",&euroskat);
    if(euroskat<0){
      printf("Λαθος τιμή:\n");
      printf("Υπολοιπο λογαριασμου: %f: \n",euros);
      return euros;
    }else{
      printf("Υπολοιπο λογαριασμου: %f \n",euros+euroskat);
      return (euros+euroskat);
    }
}
```

```
float analipsi(float euros){
    float eurosan;
    printf("Δωσε ποσο αναληψης:\n");
    scanf("%f",&eurosan);
    if(eurosan<0 || eurosan>=euros){
      printf("Λαθος τιμή!\n");
      printf("Υπολοιπο λογαριασμου: %f: \n",euros);
      return euros;
    }else{
      printf("Υπολοιπο λογαριασμου: %f \n",euros-eurosan);
      return (euros-eurosan);
    }
}

void epilogi(){
        int userinput,flag=1;
        //float euros=1000,telikoslogariasmos=0;
        float telikoslogariasmos=1000;

    do{

     printf("Πατησε: 1.Αναληψη: 2:Καταθεση 3:Τερματισμος\n");
     scanf("%d",&userinput);
     switch(userinput){
        case 1:
          telikoslogariasmos =  analipsi(telikoslogariasmos);
          break;
        case 2:
        telikoslogariasmos =    katathesi(telikoslogariasmos);
            break;
        case 3:
            flag=0 ; // δλδ: break;
            break;
        default:
        printf("Δωσε εκ νεου τιμη\n");

                    }
        }while(flag==1); // μέχρι …
}
```

Αποτελέσματα στην οθόνη του χρήστη:

```
Active code page: 1253
Εισηγαγε το κωδικο
12
Ξαναπροσπαθησε
Εισηγαγε το κωδικο
1234
Σωστος κωδικος
Πατησε: 1.Αναληψη: 2:Καταθεση 3:Τερματισμος
1
Δωσε ποσο αναληψης:
123
Υπολοιπο λογαριασμου: 877.000000
Πατησε: 1.Αναληψη: 2:Καταθεση 3:Τερματισμος
1
Δωσε ποσο αναληψης:
-14
Λαθος τιμή!
Υπολοιπο λογαριασμου: 877.000000:
Πατησε: 1.Αναληψη: 2:Καταθεση 3:Τερματισμος
2
Δωσε ποσο καταθεσης:
55.4
Υπολοιπο λογαριασμου: 932.400024
Πατησε: 1.Αναληψη: 2:Καταθεση 3:Τερματισμος
-933
Δωσε εκ νεου τιμη
Πατησε: 1.Αναληψη: 2:Καταθεση 3:Τερματισμος
-932
Δωσε εκ νεου τιμη
Πατησε: 1.Αναληψη: 2:Καταθεση 3:Τερματισμος
3
```

Στα παραπάνω τονίζεται ότι για την συνάρτηση elegxos, κατόπιν ερωτημάτων φοιτητών υπάρχει το εξής σύνηθες σφάλμα στην λογική συνθήκη ελέγχου της if:

```
void elegxos(){
    int password=1234;
    int input,flag=1;
    do{
     printf("Εισηγαγε το κωδικο\n");
     scanf("%d",&input);
     if(input==password){
        printf("Σωστος κωδικος\n");
        flag=0;}
     else{printf("Ξαναπροσπαθησε\n");}
      }while(flag==1);
}
//ΑΝΑΘΕΣΗ:      input =  password
//ΙΣΟΤΗΤΑ:      input == password
//ΟΧΙ ΙΣΟΤΗΤΑ: input != password
```

Αντίστοιχα, η εντολή switch που χρησιμοποιήθηκε:

```
printf("Πατησε: 1.Αναληψη: 2:Καταθεση 3:Τερματισμος\n");
scanf("%d",&userinput);
switch(userinput){
   case 1:
     telikoslogariasmos =  analipsi(telikoslogariasmos);
     break;
   case 2:
   telikoslogariasmos =    katathesi(telikoslogariasmos);
       break;
   case 3:
       flag=0 ; // δλδ: break;
       break;
   default:
   printf("Δωσε εκ νεου τιμη\n");

}
```

Θα γραφόταν με χρήση εντολών if ως εξής:

```
printf("Πατησε: 1.Αναληψη: 2:Καταθεση 3:Τερματισμος\n");
scanf("%d",&userinput);
if(userinput==1){
     telikoslogariasmos =  analipsi(telikoslogariasmos);
}else if(userinput==2){
   telikoslogariasmos =    katathesi(telikoslogariasmos);
}else if(userinput==3){
   flag=0 ; // δλδ: break;
}else{
   printf("Δωσε εκ νεου τιμη\n");
}
```

Εργαστήριο 5 - Αναδρομή Συναρτήσεων και Επαναληπτική Λειτουργία

Παράδειγμα 1

Να κατασκευάσετε ένα πρόγραμμα, το οποίο θα υπολογίζει το ΠΑΡΑΓΟΝΤΙΚΟ ενός αριθμού. Το πρόγραμμα θα επιτελεί τις εξής διαδικασίες:

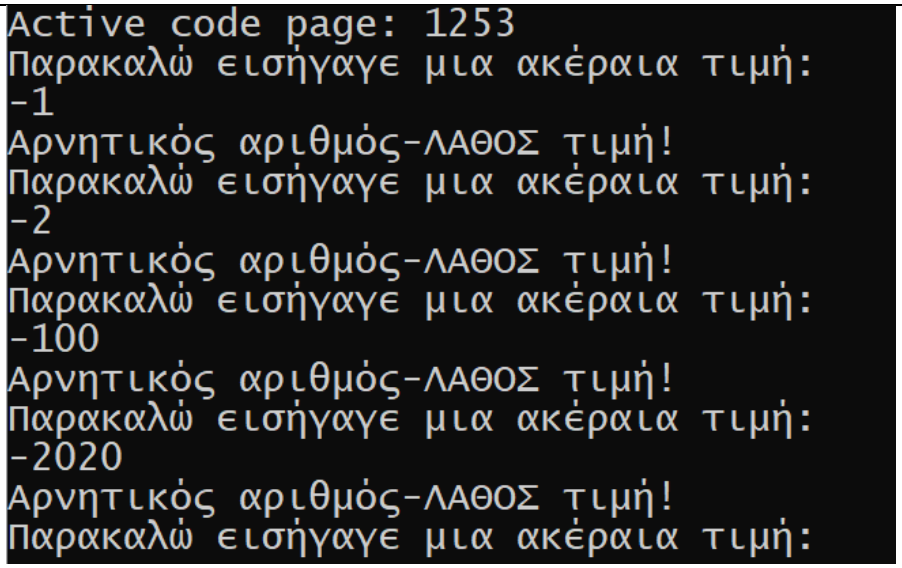
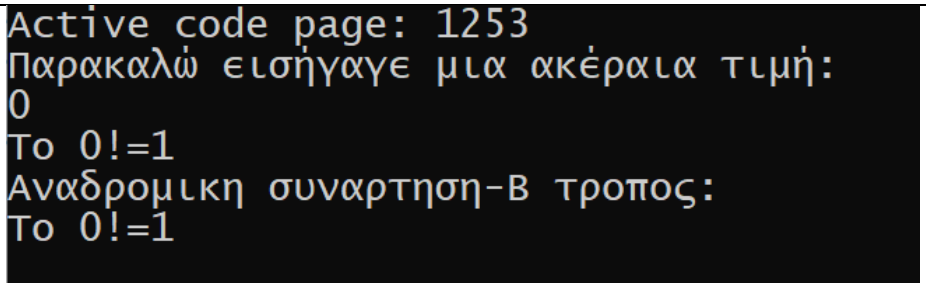
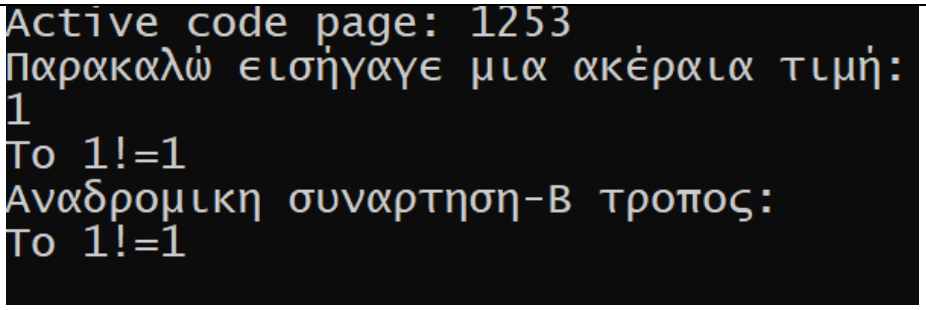
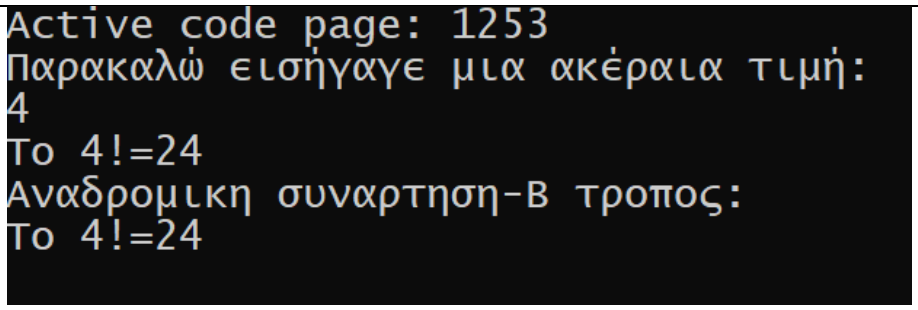
i. Θα δέχεται από τον χρήστη μια ΑΚΕΡΑΙΑ τιμή.
ii. Σε περίπτωση εισαγωγής αρνητικού αριθμού, θα εμφανίζει σχετικό μήνυμα και θα ειδοποιεί τον χρήστη για την εκ νέου εισαγωγή στοιχείων.
iii. Σε περίπτωση εισαγωγής μηδενικής τιμής, θα παράγει την έξοδο 1 (0!=1)

Οι παραπάνω διαδικασίες θα επιτελούνται μέσω της κλήσης της ΙΔΙΑΣ συνάρτησης (αναδρομή-recursion).

Τέλος, το πρόγραμμα απαιτείται να μεταγλωττίζεται και να εκτελείται επιτυχώς.

Παραδείγματα Διαδικασίας Υπολογισμού Παραγοντικού:

- **2!**=1·2= 2
- **3!**=1·2·3= 6
- **4!**=1·2·3·4= 24
- **5!**=1·2·3·4·5= 120
- **8!**=1·2·3·4·5·6·7·8= 40.320
- **10!**=1·2·3·4·5·6·7·8·9·10= 3.628.800
- **12!**=1·2·3·4·5·6·7·8·9·10·11·12= 479.001.600

Εξεταζόμενη Περίπτωση	Αποτελέσματα στην οθόνη του χρήστη
Εφαλμένη εισαγωγή στοιχείων εισόδου	
Ορθή εισαγωγή στοχείων (ορθή εισαγωγή στοιχείων)	
Ορθή εισαγωγή στοχείων (ορθή εισαγωγή στοιχείων)	
Ορθή εισαγωγή στοχείων (ορθή εισαγωγή στοιχείων)	

```c
#include <stdio.h>


long int parag(int arithmos){
	if (arithmos==0){
		return 1;
	}else{
		return (arithmos*(parag(arithmos-1)));
	}
}

int main(){
	int x;//δήλωση τύπου μεταβλητής
	system("chcp 1253");//cmd εμφάνιση ελληνικών χαρακτήρων
	do{
		printf("Παρακαλώ εισήγαγε μια ακέραια τιμή:\n");
		scanf("%d",&x);
		if (x<0){
			printf("Αρνητικός αριθμός-ΛΑΘΟΣ τιμή!\n");
		}
	}while(x<0);//Οσο (συνθηκη) κανε

	int paragontiko=1;
	int i;

	if (x==0){
		printf("Το %d!=1\n",x);
	}else{
	  //for(μεταβλητη_επαναληψης;συνθηκη;βημα αυξησης/μειωσης)
		for(i=1;i<=x;i=i+1){//ακριβως το ιδιο:for(i=1;i<=x;i++)
			paragontiko=paragontiko*i;
		}
		printf("Το %d!=%d\n",x,paragontiko);
	}
	//Β τροπος, κληση αναδρομικης συναρτησης
	long int paragontiko2=1;
	printf("Αναδρομικη συναρτηση-Β τροπος:\n");
	paragontiko2=parag(x);
	printf("Το %d!=%ld\n",x,paragontiko2);



	return 0;
}
```

Αναφορικά με την συνάρτηση `parag` :

```c
long int parag(int arithmos){
	if (arithmos==0){
		return 1;
	}else{
		return (arithmos*(parag(arithmos-1)));
	}
}
```

Για παράδειγμα αν εκτελεστεί η συνάρτηση παραγοντικό(parag) 3 φορές θα έχει τις παρακάτω τιμές:

```
//1η εκτέλεση
τιμή arithmos | τιμή paragontiko((arithmos-1))
  3           |  paragontiko(3-1)=2
```

```
//2η εκτέλεση
τιμή arithmos | τιμή paragontiko((arithmos-1))
  2           |  paragontiko(2-1)=1

//3η εκτέλεση
τιμή arithmos | τιμή paragontiko((arithmos-1))
1             | paragontiko(1-1)=0
```

Αναφορικά με τον Α τρόπο υπολογισμού του παραγοντικού με δομή επανάληψης:

```
  //for(μεταβλητη_επαναληψης;συνθηκη;βημα αυξησης/μειωσης)
            for(i=1;i<=x;i=i+1){//ακριβως το ιδιο:for(i=1;i<=x;i++)
                  paragontiko=paragontiko*i;
            }
```

Για παράδειγμα αν εκτελεστεί η δομή επανάληψης 3 φορές θα έχει τις παρακάτω τιμές:

```
//δομή επανάληψης για i<=3

τιμη i | συνθηκη(i<=3) |τιμη paragontiko| Επαναληψη νουμερο

-----------------------------------------------------

1       |  1<=3          | 1*1=1           |  επαναληψη νουμερο 1
2       |  2<=3          | 1*2=2           |  επαναληψη νουμερο 2
3       |  3<=3          | 2*3=6           |  επαναληψη νουμερο 3
4       |  4<=3          | -               |  -
```

Τέλος, παρατηρείται ότι κάθε τύπος μεταβλητής έχει συγκεκριμένο εύρος τιμών (δηλαδή έχει αντίστοιχη χωρητικότητα).
Ενδεικτικά, μερικοί από τους συνήθης τύπους μεταβλητών είναι οι παρακάτω:

Type	Bytes	Range (at least)
short int	2	-32,768 -> +32,767
unsigned short int	2	0 -> +65,535
unsigned int	4	0 -> +4,294,967,295
int	4	-2,147,483,648 -> +2,147,483,647
long int	4	-2,147,483,648 -> +2,147,483,647
signed char	1	-128 -> +127
unsigned char	1	0 -> +255
float	4	1.2E-38 -> 3.4E+38
double	8	2.3E-308 -> 1.7E+308
long double	12	3.4E-4932 -> 1.1E+4932

Παράδειγμα 2

Σε συνέχεια του π.χ.1, παρουσιάζεται άσκηση αναφορικά με τα μεγέθη των τύπων μεταβλητών. Ειδικότερα, χρησιμοποείται η εντολή sizeof(τυπος_μεταβλητης) η οποία υπολογίζει το μέγεθος σε bytes.

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

        printf("sizeof(char) == %d\n", sizeof(char));
        printf("sizeof(short) == %d\n", sizeof(short));
        printf("sizeof(int) == %d\n", sizeof(int));
        printf("sizeof(long) == %d\n", sizeof(long));
        printf("sizeof(float) == %d\n", sizeof(float));
        printf("sizeof(double) == %d\n", sizeof(double));
        printf("sizeof(long double) == %d\n", sizeof(long double));
        printf("sizeof(long long) == %d\n", sizeof(long long));

        return 0;}
```

Αποτελέσματα στην οθόνη του χρήστη:

```
sizeof(char) == 1
sizeof(short) == 2
sizeof(int) == 4
sizeof(long) == 4
sizeof(float) == 4
sizeof(double) == 8
sizeof(long double) == 16
sizeof(long long) == 8
```

Παράδειγμα 3

Να κατασκευάσετε ένα πρόγραμμα, το οποίο:

- Θα επιτελεί, μέσω μιας αναδρομικής συνάρτησης, τον υπολογισμό της δύναμης ενός αριθμού (π.χ. 20 =1, 23 =8, (-3)2 =9, …)
- Το κυρίως πρόγραμμα επιβάλλεται:
  1. Να ζητά από τον χρήστη την εισαγωγή ενός αριθμού (θετικού, αρνητικού ή μηδέν).
  2. Να ζητά από τον χρήστη την εισαγωγή της δύναμης που θα υπολογιστεί.

3. Να καλεί μια αναδρομική συνάρτηση υπολογισμού του τελικού αποτελέσματος.
4. Να εκτελείται συνεχώς (μέσω μιας δομής επανάληψης) μέχρι ο χρήστης να πληκτρολογήσει την τιμή (αγγλικού χαρακτήρα) → y ή Y
5. Να αποτελείται από τις εξής συναρτήσεις: εισαγωγή αριθμών από το χρήστη (ερ.1,ερ.2), ο έλεγχος της τιμής του τερματισμού του προγράμματος (ερ.4), η αναδρομική συνάρτηση υπολογισμού (ερ.3).

- Το πρόγραμμα απαιτείται να μεταγλωττίζεται και να εκτελείται επιτυχώς.

```
#include <stdio.h>
#include <math.h>
#include <conio.h>
int inputXristi();
char check();
int VresDinami(int x, int n );

main ()
{
    //system("chcp 1253"); //cmd εμφάνιση ελληνικών χαρακτήρων

    int x,n,apotelesma;
    char flag='n';
    while(flag=='n'){
    printf("Eisigage arithmo: \n");
    x=inputXristi();
    printf("Eisigage ti dynami pou thes na ypologiseis: \n");
    n=inputXristi();
    apotelesma=VresDinami(x,n);
    printf("Apotelesma: %d \n",apotelesma);
    //system("pause");
    flag=check();
}
}

int inputXristi(){
        int number;
        scanf("%d",&number);
        return number;
        }

char check(){
    char ch;
    printf(" To terminate execution press: y \n Press anything
else to continue\n\n ");
    ch=getche();
    printf("\n");
    if (ch=='y' || ch=='Y'){
     return ch;
    }else
    {return 'n';
    }
}

int VresDinami(int x, int n )
{
    if (n==0) return 1;
    if (n%2 == 0) {
        if (n==2) return x*x;
```

```
            return VresDinami(VresDinami(x,n/2),2);
        }
        else return x*VresDinami(x, n-1);
    }
```

Αποτελέσματα στην οθόνη του χρήστη:

```
Eisigage arithmo:
0
Eisigage ti dynami pou thes na ypologiseis:
1
Apotelesma: 0
 To terminate execution press: y
 Press anything else to continue


Eisigage arithmo:
2
Eisigage ti dynami pou thes na ypologiseis:
3
Apotelesma: 8
 To terminate execution press: y
 Press anything else to continue


Eisigage arithmo:
5
Eisigage ti dynami pou thes na ypologiseis:
3
Apotelesma: 125
 To terminate execution press: y
 Press anything else to continue

 t
Eisigage arithmo:
3
Eisigage ti dynami pou thes na ypologiseis:
2
Apotelesma: 9
 To terminate execution press: y
 Press anything else to continue

 y
```

Παράδειγμα 4

Να κατασκευάσετε ένα πρόγραμμα, όπου:

- Θα επιτελείται μια αναδρομική συνάρτηση, η οποία θα υπολογίζει το n-ιοστο όρο της ακολουθίας FIBONACCI, ο οποίος δίνεται από την επαναληπτική σχέση:

$$U_1 = 1 \quad U_2 = 1 \quad U_n = U_{n-1} + U_{n-2} \quad (\text{για } n>2)$$

- Το κυρίως πρόγραμμα θα υπολογίζει το μέγιστο όρο της ακολουθίας, ο οποίος μπορεί να υπολογιστεί όταν η μεταβλητή U δηλωθεί με τους 4 διαφορετικούς τρόπους: int, long, double και long double
- Το πρόγραμμα απαιτείται να μεταγλωττίζεται και να εκτελείται επιτυχώς.

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

long fib (int k)
{
      if (k==0) {
      return 1; //proti arxiki sytnthiki
      }
      else{
            if (k==1){
                  return 1; //deuteri arxiki sytnthiki
            }
            else{
                  return (fib(k-2)+fib(k-1)); //anadromikos typos
            }
      }
}

main()
{
      int n;
      long p;
      char ch;
      do
      {
         do{
            printf("Dvse ena thetiko akeraio mexri 20: \n");
            scanf("%d",&n);
         }while(n<0||n>20);
       p=fib(n);
       printf("H timi tou zitoumenou arithmou fibonacci einai: %ld\n",p);
       printf("Thes na epanalaveis; y/n \n");
       ch=getche();//αντίστοιχη εντολή η εξής: scanf("%c",&ch);
       printf("\n");
      }while(ch!='n'&& ch!='N');
}
```

Αποτελέσματα στην οθόνη του χρήστη:

```
Dvse ena thetiko akeraio mexri 20:
-3
Dvse ena thetiko akeraio mexri 20:
0
H timi tou zitoumenou arithmou fibonacci einai: 1
Thes na epanalaveis; y/n
y
Dvse ena thetiko akeraio mexri 20:
3
H timi tou zitoumenou arithmou fibonacci einai: 3
Thes na epanalaveis; y/n
y
Dvse ena thetiko akeraio mexri 20:
123
Dvse ena thetiko akeraio mexri 20:
13
H timi tou zitoumenou arithmou fibonacci einai: 377
Thes na epanalaveis; y/n
n
```

Εργαστήριο 6 - Πίνακες

Παράδειγμα 1

Να κατασκευάσετε ένα πρόγραμμα, το οποίο :

1. Θα δημιουργεί έναν πίνακα Ν διαστάσεων ακεραίων τιμών.
2. Θα υπολογίζει το άθροισμα όλων των στοιχείων(sum), τον ελάχιστο(min) και τον μέγιστο(max) αριθμό.
3. Θα υπολογίζει το άθροισμα όλων των στοιχείων(sum), τον ελάχιστο(min) και τον μέγιστο(max) αριθμό.

- Όπου Ν θετικός ακέραιος αριθμός που εισάγεται από το χρήστη
- Η ανάθεση τιμών στους πίνακες θα γίνεται ΜΕ ΤΥΧΑΙΟ ΤΡΟΠΟ (π.χ. χρήση rand συνάρτησης).

Το πρόγραμμα απαιτείται να μεταγλωττίζεται και να εκτελείται επιτυχώς .

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

int main(){
    system("chcp 1253");//cmd εμφάνιση ελληνικών χαρακτήρων
    int n ;//ο τύπος της μεταβλητής δηλώνεται ΠΑΝΤΑ μια και μόνο μια φορά!
    do{
      printf("Εισήγαγε την τιμή του N:\n");
      scanf("%d",&n);
      printf("Ο χρήστης εισήγαγε N=%d\n",n);
      if (n<=0){
            printf("\tΛαθος (αρνητική) τιμή, N=%d\n\n\n",n);
       }else{
```

```
            printf("\tΣωστη (θετική) τιμή, N=%d\n\n\n\n",n);
        }
    }while(n<=0);
    int pinakas[n];//πινακας δηλωση: ονομα_πινακα[]
    int max,min,sum,i;
    sum=0;//αθροισμα
    srand(time(NULL));//τυχαιες τιμές! Συνάρτηση rand--> rand();
/*     RAND_MAX is a constant whose default value is granted to be at least
32767-->
   https://www.tutorialspoint.com/c_standard_library/c_function_rand.htm
*/

    for (i=0;i<n;i=i+1){//for(αρχικο_βημα;συνθηκη_τερματισμου;βημα_αυξησης){}
      pinakas[i] = rand(); //Length πίνακα=N-1 (1ο στοιχείο==pinakas[0])

      sum=sum+pinakas[i];
       printf("\nΤο στοιχειο i=%d και ο pinakas[%d]=%d \n", i, i,
              pinakas[i]);
      }
      printf("Το SUM=%d \n",sum);

      min=pinakas[0]; // έστω ότι το 1ο στοχείο του πίνακα=minimum
      max=pinakas[0]; // έστω ότι το 1ο στοχείο του πίνακα=maximum

/*Επιλέχθηκε αυτή η for και όχι: for(i=0;i<n;i=i+1) καθώς στη 1η
επανάληψη δλδ για i=0 θα επιτελεί  σύγκριση με τον εαυτό του... */
      for (i=1;i<n;i=i+1){
        if (min>pinakas[i]){
            min=pinakas[i];
        }

        if (max<pinakas[i]){
            max=pinakas[i];
        }
      }
      printf("Το MIN=%d και το MAX=%d",min,max);
      return 0;
}
```

Αναφορικά με την δομή επανάληψης do while:

```
do{
            printf("Εισήγαγε την τιμή του N:\n");
            scanf("%d",&n);
      printf("Ο χρήστης εισήγαγε N=%d\n",n);
            if (n<=0){
                  printf("\tΛαθος (αρνητική) τιμή, N=%d\n\n\n",n);
            }else{
                  printf("\tΣωστη (θετική) τιμή, N=%d\n\n\n\n",n);
            }
      }while(n<=0);
```

Τονίζεται ότι ισοδύναμα θα μπορούσε να είχε χρησιμοποιηθεί ο παρακάτω κώδικας:

```
// Β τρόπος δομή επανάληψης while…
int n;//ο τύπος της μεταβλητής δηλώνεται ΠΑΝΤΑ μια και μόνο μια φορά!
printf("Εισήγαγε την τιμή του N:\n");
scanf("%d",&n);
while(n<=0){
  if (n<=0){
      printf("\tΛαθος (αρνητική) τιμή, N=%d\n\n\n",n);
      printf("Εισήγαγε την τιμή του N:\n");
    scanf("%d",&n);
  }else{
```

```
        printf("\tΣωστη (θετική) τιμή, N=%d\n\n\n\n",n);
    }
}
```

Αποτελέσματα στην οθόνη του χρήστη για λάθος τιμή εισόδου από το χρήστη:

```
Active code page: 1253
Εισήγαγε την τιμή του N:
0
Ο χρήστης εισήγαγε N=0
        Λαθος (αρνητική) τιμή, N=0


Εισήγαγε την τιμή του N:
-12
Ο χρήστης εισήγαγε N=-12
        Λαθος (αρνητική) τιμή, N=-12
```

Αποτελέσματα στην οθόνη του χρήστη(τονίζεται ότι πίνακας 12 θέσων==0,1,2,3,…11):

```
Active code page: 1253
Εισήγαγε την τιμή του N:
12
Ο χρήστης εισήγαγε N=12
        Σωστη (θετική) τιμή, N=12



Το στοιχειο i=0 και ο pinakas[0]=15522

Το στοιχειο i=1 και ο pinakas[1]=18271

Το στοιχειο i=2 και ο pinakas[2]=8730

Το στοιχειο i=3 και ο pinakas[3]=18239

Το στοιχειο i=4 και ο pinakas[4]=1347

Το στοιχειο i=5 και ο pinakas[5]=6799

Το στοιχειο i=6 και ο pinakas[6]=9616

Το στοιχειο i=7 και ο pinakas[7]=14610

Το στοιχειο i=8 και ο pinakas[8]=24941

Το στοιχειο i=9 και ο pinakas[9]=32363

Το στοιχειο i=10 και ο pinakas[10]=26928

Το στοιχειο i=11 και ο pinakas[11]=6098
Το SUM=183464
Το MIN=1347 και το MAX=32363
```

Αποτελέσματα στην οθόνη του χρήστη(τονίζεται ότι πίνκας 13 θέσων==0,1,2,3,…12):

```
Active code page: 1253
Εισήγαγε την τιμή του N:
13
Ο χρήστης εισήγαγε N=13
        Σωστη (θετική) τιμή, N=13



Το στοιχειο i=0 και ο pinakas[0]=16403

Το στοιχειο i=1 και ο pinakas[1]=3989

Το στοιχειο i=2 και ο pinakas[2]=15160

Το στοιχειο i=3 και ο pinakas[3]=27269

Το στοιχειο i=4 και ο pinakas[4]=1002

Το στοιχειο i=5 και ο pinakas[5]=18440

Το στοιχειο i=6 και ο pinakas[6]=30778

Το στοιχειο i=7 και ο pinakas[7]=10677

Το στοιχειο i=8 και ο pinakas[8]=26622

Το στοιχειο i=9 και ο pinakas[9]=10785

Το στοιχειο i=10 και ο pinakas[10]=19762

Το στοιχειο i=11 και ο pinakas[11]=17991

Το στοιχειο i=12 και ο pinakas[12]=6175
Το SUM=205053
Το MIN=1002 και το MAX=30778
```

Εργαστήριο 7 & 8- Αρχεία

Τα αρχεία είναι από τα πιο σημαντικά κεφάλαια στην C. Ειδικότερα, όπως και στην πραγματική ζωή, οι πληροφορίες που παράγονται από τα υπολογιστικά συστήματα από τις πιο απλές έως και τις σύνθετες διαδικασίες αποθηκεύονται είτε τοπικά στον υπολογιστή μας είτε σε κάποια νεφοϋπολογιστική δομή (cloud). Ο πιο συνήθης όμως τρόπος αποθήκευσης και αποστολής πληροφοριών είναι μέσω της χρήσης αρχείων.

Σε μια επιχείρηση, όπως για παράδειγμα μια τράπεζα, συχνά απαιτείται από το προγραμματιστή όχι απλώς να αποστείλει κάποιες πληροφορίες αλλά, να δημιουργήσει ένα αρχείο με ένα συγκεκριμένο όνομα και να αποθηκεύσει εντός του διάφορα πληροφορίες. Όταν ζητείται να αποθηκεύσουμε τις πληροφορίες εντός ενός αρχείου (π.χ. κειμένου) με συγκεκριμένη μορφοποίηση τότε αυτό το αρχείο ονομάζεται «αρχείο γραμμογράφησης».

Για έναν προγραμματιστή κάθε αλληλεπίδραση με δομές αρχείων αποτελείται από 3 διαδικασίες: άνοιγμα, επεξεργασία/ανάγνωση και κλείσιμο.

Στη συνέχεια θα δούμε αναλυτικά τις εντολές που χρησιμοποιούνται για κάθε μια από αυτές τις διαδικασίες:

- Για το **άνοιγμα και την επεξεργασία** του αρχείου χρησιμοποιείται η συνάρτηση:

```
FILE *fopen(const char *filename, const char *mode)
```

Για παράδειγμα, η εντολή για την δημιουργία (και εγγραφή) ενός txt αρχείου με ονομασία «C_erg.txt» στον τοπικό δίσκο «C:/» θα ήταν η ακόλουθη:

```
fp=fopen("C://C_erg.txt","w");//w:write
```

Ειδικότερα, η παραπάνω συνάρτηση επιτελεί τα παρακάτω:

1. Το 1ο όρισμα (`const char *filename`) αφορά την ονομασία (και τοποθεσία) του αρχείου.
   (αν δεν συμπληρωθεί η τοποθεσία π.χ. C:/ τότε αυτή είναι ίδια με την τοποθεσία αποθήκευσης του εκτελέσιμου (.exe) αρχείου στο δίσκο)

2. Το 2ο όρισμα αφορά τον τρόπο ανοίγματος του αρχείου (r,w,a,r+,w+,a+)

Η συνάρτηση επιστρέφεται δείκτη στη δομή FILE που εμπεριέχει όλες τις απαραίτητες πληροφορίες για έναν προγραμματιστή (π.χ. σε περίπτωση λάθους επιστρέφει null).

Ακολουθεί αναλυτικός πίνακας των τιμών του 2ου ορίσματος αναφορικά με τον τρόπο ανοίγματος-ενέργειας στο αρχείο:

Πίνακας 1 Επεξήγηση ενεργειών τρόπου ανοίγματος και επεξεργασίας αρχείων

Τρόπος Ανοίγματος & Ενέργειας	**Περιγραφή Ενέργειας**	**Αν αρχείο==true (ύπαρξη αρχείου)**	**Αν αρχείο==false (μη ύπαρξη αρχείου)**
r	Διάβασμα (read)	Δείκτης στην αρχή αρχείου	Null
w	Εγγραφή (write)	Διαγραφή περιεχομένων (σβήσιμο)	Δημιουργία αρχείου
a	Πρόσθεση (append)	Δείκτης στο τέλος αρχείου	Δημιουργία αρχείου
r+	Διάβασμα Ενημέρωση	Δείκτης στη αρχή αρχείου Εγγραφή σε οποιαδήποτε θέση (γράψιμο)	Null
w+	Εγγραφή Ενημέρωση	Διαγραφή περιεχομένων (σβήσιμο)	Δημιουργία αρχείου
a+	Πρόσθεση Ενημέρωση	Δείκτης στη αρχή αρχείου Εγγραφή στο τέλος αρχείου (γράψιμο)	Δημιουργία αρχείου

- Για το **<u>κλείσιμο</u>** του αρχείου χρησιμοποιείται η συνάρτηση:

```
int fclose (FILE *fp) // ή int fcloseall(void)
```

Για παράδειγμα, η εντολή για το κλείσιμο ενός ανοιγμένου αρχείου (fp) θα ήταν η ακόλουθη:

```
fclose(fp);
```

Ειδικότερα, η παραπάνω συνάρτηση επιστρέφει μια από τις ακόλουθες ακέραιες τιμές:

1. 0 → για επιτυχές κλείσιμο του αρχείου
2. 1 → για μη επιτυχές κλείσιμο του αρχείου

- Για την **εκτύπωση** κάποιας τιμής σε ένα αρχείο χρησιμοποιείται η συνάρτηση:

```
int fclose (FILE *fp) // ή int fcloseall(void)
```

Για παράδειγμα, η εντολή για την εγγραφή του κειμένου «test keimeno» σε ένα ήδη υπάρχον ανοιγμένο αρχείο (fp) θα ήταν η ακόλουθη:

```
fprint(fp, "test keimeno"); //ίδια λειτουργία με printf
```

Αντίστοιχη εντολή για την **εκτύπωση** μιας τιμής ενός χαρακτήρα (char) σε ένα αρχείο είναι η χρήση της εξής συνάρτησης:

```
int putc(int ch, FILE *fp) // ή int fputc(int ch, FILE *fp)
```

Για παράδειγμα, η εντολή για την εγγραφή του χαρακτήρα «Α» σε ένα ήδη υπάρχον ανοιγμένο αρχείο (fp) θα ήταν η ακόλουθη:

```
putc('A',fp);
```

Τέλος, τονίζεται ότι η συνάρτηση αυτή επιστρέφει τον χαρακτήρα που έγινε εγγραφή ή EOF (end of file) αν προκύψει κάποιο σφάλμα.

ΠΡΟΣΟΧΗ: Για να εκτελεστούν τα παρακάτω παραδείγματα απαιτούνται δικαιώματα διαχειριστή συνεπώς, προτείνεται:

1. Να αποθηκεύσετε το .c αρχείο σας (άρα και το compiled αρχείο .exe) είτε στο σκληρό σας δίσκο C:/ είτε στον κοινόχρηστο φάκελο των Λήψεων (Downloads).
2. Αν δεν εκτελείται compilation και execution μέσω του cmd αλλά, μέσω κάποιου προγράμματος όπως το DevCpp ή το ChIDE, ή Codeblocks σε περιβάλλον Windows επιλέξτε κατά το άνοιγμα και εκτέλεση του προγράμματος «Εκτέλεση με Δικαιώματα Διαχειριστή» (Run as Administrator).

Παράδειγμα 1

Να κατασκευάσετε ένα πρόγραμμα, το οποίο:

- Να πραγματοποιεί όλες τις απαραίτητες διαδικασίες (open, write, close) σε ένα αρχείο κειμένου
- Κατόπιν ανοίγματος του αρχείου με την χρήση κατάλληλης εντολής, θα γράφει τα εξής κείμενα:
  - ✓ Γραμμή 1: "Ergastirio C"
  - ✓ Γραμμή 2: "Erg7-2020 "
- Κατόπιν της επεξεργασίας (εγγραφής γραμμής 1,2) να κλείνει το αρχείο το οποίο επεξεργάστηκε
- Εκτελέστε εκ νέου το αρχείο για άλλο κείμενο στη γραμμή 1,2. Τι παρατηρείτε ;

Το πρόγραμμα απαιτείται να μεταγλωττίζεται και να εκτελείται επιτυχώς.

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

main(){
	FILE *fp;//ορισμος FILE(για fopen/fclose...)
	fp = fopen("C://Ergastirio7C_ask1.txt","w");
//fp = fopen("C:\\Users\\alex\\Downloads\\ErgastirioCask1.txt","w");//w:write
	if (fp==NULL){
		printf("File can not OPEN!\n");//cmd
		exit(0);//διακοπη του προγραμματος
	}else{
		printf("File can OPEN!\n");//cmd
		//εκτυπωσε μεσα στο αρχειο:
		//A τροπος
		int arithmos1,arithmos2;
		arithmos1= 7;
		arithmos2= 2020;
		fprintf(fp,"B tropos: Ergastirio C\nErg%d-
%d",arithmos1,arithmos2);//file
		//B τροπος:fprintf(fp,"A tropos: Ergastirio C\nErg7-2020\n");
	}
	fclose(fp); }
```

Αναφορικά με την παραπάνω υλοποίηση, τονίζεται ότι οι υπολογιστές έχουν μια ροή εισόδου (stdin) και μια ροή εξόδου (stdout).
Ειδικότερα, όταν θέλουμε να ελέγξουμε τι υπάρχει στη ροή εξόδου(stdout, δηλαδή, τι εκτυπώνεται στη γραμμή εντολών (cmd) πριν την εγγραφή στο αρχείο μπορούμε να χρησιμοποιήσουμε την παρακάτω εντολή:

```
fprintf(stdout,"B tropos: Ergastirio C\nErg%d-%d",arithmos1,arithmos2);//file
```

Τέλος, παρατηρούμε ότι επειδή το αρχείο έχει ανοίξει με επιλογή *w* τη 1η φορά που εκτελείται ο κώδικας το αρχείο δημιουργείται στη διεύθυνση αποθήκευσης που έχει ορισθεί.
Τη 2η φορά που εκτελείται το αρχείο (και υπάρχει ήδη στη συγκεκριμένη διεύθυνση αρχείο με τη ίδια ονομασία), γίνεται αντικατάσταση (διαγραφή του παλιού αρχείου) με βάση το νέο αρχείο της τελευταίας εκτέλεσης.
→Για περισσότερες πληροφορίες δείτε το πίνακα 1 (Επεξήγηση ενεργειών τρόπου ανοίγματος και επεξεργασίας αρχείων)

Παράδειγμα 2 (επεξεργασία αρχείου Παραδείγματος 1)

Να κατασκευάσετε ένα πρόγραμμα, το οποίο:

- Να πραγματοποιεί όλες τις απαραίτητες διαδικασίες (open, write, close) σε ένα αρχείο κειμένου
- Κατόπιν ανοίγματος του αρχείου με την χρήση κατάλληλης εντολής, να επεξεργάζεται το ανοιγμένο αρχείο προσθέτοντας μια εγγραφή όπως το ονοματεπώνυμό σας.
- Κατόπιν της επεξεργασίας να κλείνει το αρχείο το οποίο επεξεργάστηκε
- Εκτελέστε εκ νέου το αρχείο για άλλο κείμενο στη γραμμή 1,2. Τι παρατηρείτε ;

Το πρόγραμμα απαιτείται να μεταγλωττίζεται και να εκτελείται επιτυχώς.

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

main(){
      FILE *fp;//ïñéóìïò FILE(ãéá fopen/fclose...)

      fp = fopen("C://Ergastirio7C_ask1.txt","a");
//fp = fopen("C:\\Users\\alex\\Downloads\\ErgastirioCask1.txt","a");//a:append
      if (fp==NULL){
            printf("File can not OPEN!\n");//cmd
            exit(0);//διακοπή προγράμματος
      }else{
            printf("File can OPEN!\n");//cmd
            fprintf(fp,"\n Alex Gazis");//file
      }
      fclose(fp); }
```

Τέλος, παρατηρούμε ότι επειδή το αρχείο έχει ανοίξει με επιλογή *a* κάθε φορά προστίθενται εγγραφές στο αρχείο του Παραδείγματος 1.
Αν δεν υπάρχει το αρχείο του Παραδείγματος 1 (δηλαδή δεν υπάρχει αρχείο στο διεύθυνση αποθήκευσης) τότε δημιουργείται αυτόματα το αντίστοιχο αρχείο.
→Για περισσότερες πληροφορίες δείτε το πίνακα 1 (Επεξήγηση ενεργειών τρόπου ανοίγματος και επεξεργασίας αρχείων)

Παράδειγμα 3

Να κατασκευάσετε ένα πρόγραμμα, το οποίο:

- Να εγγράφει σε ένα αρχείο, στην περιφερειακή μονάδα της επιλογής σας (π.χ. σε ένα flash στη μονάδα E:\) με όνομα ***test1.txt***, μια εγγραφή.
- Η εγγραφή, επιβάλλεται να περιέχει δύο πραγματικούς αριθμούς (ένα θετικό και ένα αρνητικό), οι οποίοι θα εισάγονται από το πληκτρολόγιο, με τη βοήθεια κατάλληλων μηνυμάτων.
- Ο ορισμός της περιφερειακής μονάδας να πραγματοποιείται από τη γραμμή εντολών (command mode).

Το πρόγραμμα απαιτείται να μεταγλωττίζεται και να εκτελείται επιτυχώς.

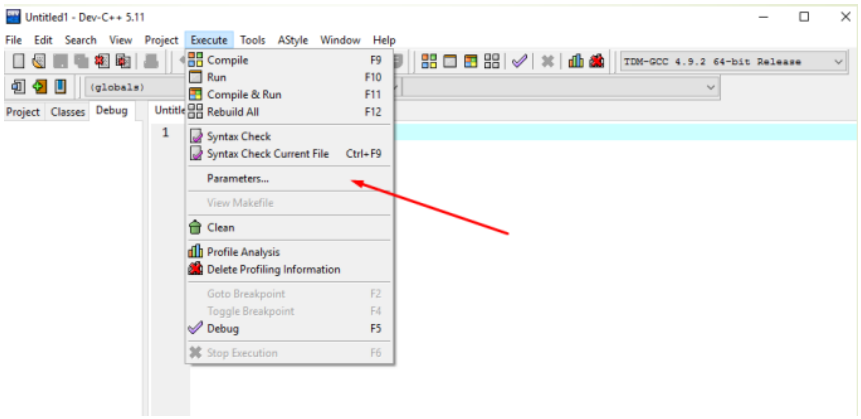
```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char* argv[]){
//argc: argument count {πλήθος στοιχειων}
//argv: argument vector{τιμες στοιχείων}
      system("chcp 1253");//cmd εμφάνιση ελληνικών χαρακτήρων
      FILE* fp;
      printf("argc=%d \n",argc);
      int i;
      for (i=0;i<argc;i=i+1){//β τροπος:  for (i=0;i<argc;i++){...}
            printf("argv[%d]=%s \n",i, argv[i]);
      }

      //Αρα π.χ. εκτυπωσης για πινακα 4 θεσεων{0,1,2,3}:
      // i   |     Εκτυπωση                |Επαναληψη
      // 0   |     arg[0]=κειμενο0         |1η επαναληψη
      // 1   |     arg[1]=κειμενο1         |2η επαναληψη
      // 2   |     arg[2]=κειμενο2         |3η επαναληψη
      // 3   |     arg[3]=κειμενο3         |4η επαναληψη

//Για παράδειγμα, στα πλαίσια του εργ., για συγκεκριμενο πληθος στο argc θα ισχύει:
      if (argc!=4){//Το συμβολο '!=' ειναι το ΟΧΙ ισο, ενω το ΙΣΟ ειναι '=='
       printf("You did not supply the correct number of arguments.");
       printf("\n Correct syntax: <program> file_name arg0 arg1 arg2 arg3 \n") ;
       //arg0:ονομα προγράμματος (αυτοματα απο το ιδιο το προγραμμα)
       //arg1:αρχειο που θα γίνει η εγγραφή
       //arg2:ορισμα 1 (αριθμός 1 που θα γραφτεί & προστεθεί)
       //arg3:ορισμα 2 (αριθμός 2 που θα γραφτεί & προστεθεί)
       exit(1); //exit(1)==εξοδος ΟΚ
      }

      fp=fopen(argv[1],"w");//w:write
      //fp=fopen("C://dokimastiko.txt","w");//w:write

      if(fp==NULL){
        printf("The file could not be opened for writing. Reasons? Maybe ");
        printf("there is not enough disk space or you have not granted ");
        printf("the access to perform an action(write) ");
        printf("to the specific location\n");
        exit(2);//exit(2)==εξοδος ΟΚ
```

```
        }

        //putc: εντολη εκτυπωση στο αρχειο δλδ PutCharacter
        for (i=0;argv[2][i]!=0;i++){ //for (i=0;i<2;i++){
              putc(argv[2][i],fp);
              printf(">>argv[2][%d]=%c\n",i,argv[2][i]);
        }

        putc(',',fp);

        for (i=0;argv[3][i]!=0;i++){//for (i=0;i<2;i++){
              putc(argv[3][i],fp);
              printf(">>argv[3][%d]=%c\n",i,argv[3][i]);
        }


        printf("The arguments 2 and 3 are: %s , %s \n",argv[2],argv[3] );
        printf("The arguments 1 (i.e. file name to be written):%s \n",argv[1]);

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

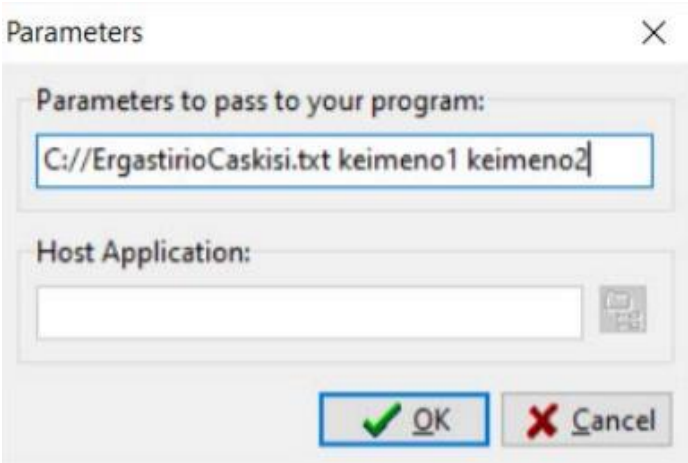
```
//B τρόπος υλοποίησης
#include <stdio.h>
#include <locale.h>
#include <stdlib.h>
#include <string.h>
int main(int argc, char **argv)
{
   system("chcp 1253"); //cmd εμφάνιση ελληνικών χαρακτήρων

   /* Δηλωση μεταβλητων */
   float r1,r2;
   FILE *fp; /* δεικτης αρχειου*/
   char *temp = ":\text1.t"; // αρχική τιμή ονόματος αρχείου
   if(argc !=2)
   {
     printf("Η χρηση της εντολης ειναι : %s Γραμμα_περιφερειακης_μοναδας (π.χ. D , E)\n",
argv[0]);
     exit(1);
   }
   /* Δημιουργια πλήρους θέσης αποθήκευσης αρχείου π.χ.  "E:\test1.dat " */
   char *file=argv[1];
   file=strcat(file, temp); // πρόσθεση της τιμής της μεταβλητής file και temp στη file
  /* Αναγνωση δεδομένων μέσω αλληλεπίδρασης με το χρήστη*/
   do
   {
     printf("Δωστε εναν θετικο πραγματικο αριθμο\n");
     fflush(stdin);
   }while(0!=(scanf("%f",&r1)) && r1<=0);

   do
   {
```

```
    printf("Δωστε εναν αρνητικο πραγματικο αριθμο\n");
    fflush(stdin);
  }while(0!=(scanf("%f",&r2)) && r2>=0);

  /* Ανοιγμα, εγγραφή στο αρχείο & κλείσιμο αρχείο με την εκτυπωση μηνύματος */
  fp=fopen(file, "w");
  fprintf(fp, "%f %f \n", r1, r2);
  fclose(fp);
  printf ("\nΕδωσες τους πραγματικούς %f και %f και αποθηκεύτηκαν στο αρχείο %s\n", r1,
r2, file);
  return 0;
}
```

Για την επιλογή της παραμέτρου/τοποθεσίας στο περιβάλλον του Dev-C++, ακολουθείται η εξής διαδικασία:



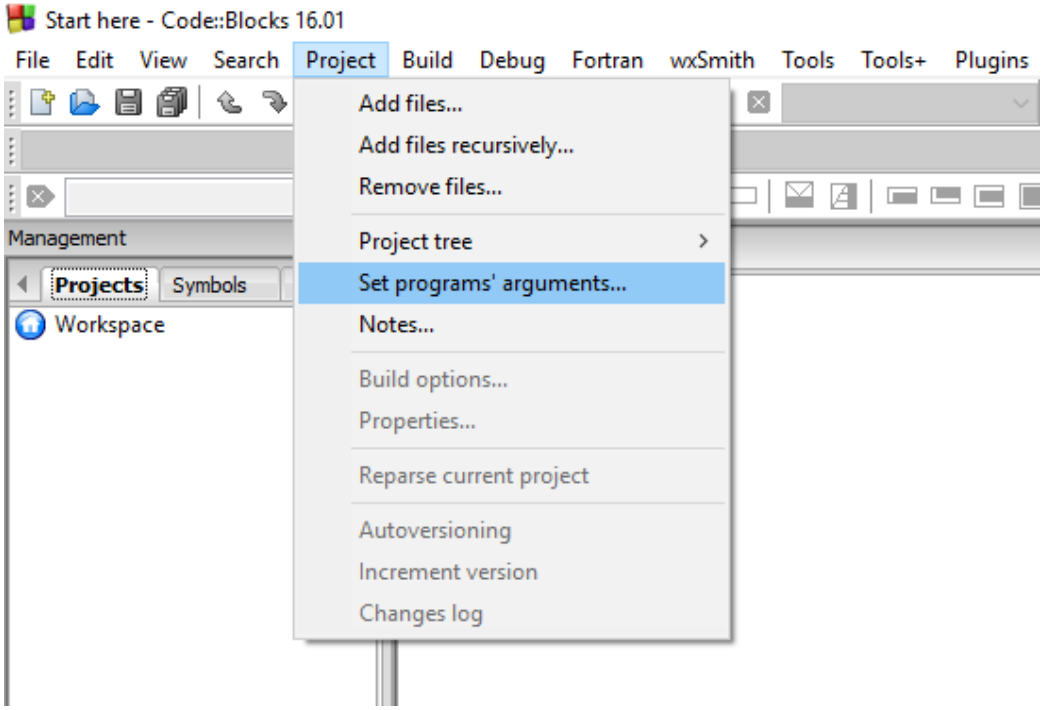
Εικόνα 15 *Βήμα 1 επιλέγω Parameters*



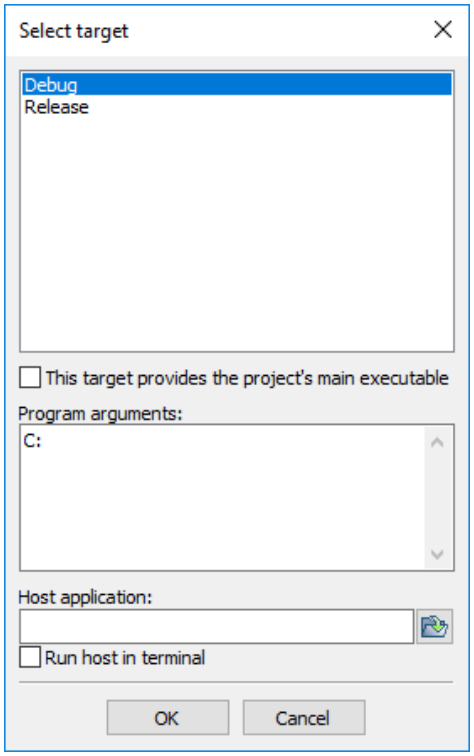
Εικόνα 16 *Βήμα 2 ορίζω την παράμετρο. Προσοχή στο γράμμα/διεύθυνση που θα ορίσετε στο πρόγραμμά σας*

κείμενο εικόνας (προσοχή στους κενούς χαρακτήρες κατά την αντιγραφή)→C://ErgastirioCaskisi.txt keimeno1 keimeno2

Η αντίστοιχη διαδικασία και για το Codeblocks:



Εικόνα 17 *Βήμα 1 επιλέγω arguments, για το πρόγραμμα που χρησιμοποιώ*



Εικόνα 18 *Βήμα 2 ορίζω την αποθηκευτική μονάδα. Προσοχή στο γράμμα/διεύθυνση που θα ορίσετε στο πρόγραμμά σας*

Παράδειγμα 4

Να κατασκευάσετε ένα πρόγραμμα το οποίο:

- Με τη βοήθεια μηνυμάτων στην οθόνη (στην ελληνική γλώσσα και με ελληνικά γράμματα) θα δέχεται από το πληκτρολόγιο δύο μιγαδικούς αριθμούς (πραγματικό και φανταστικό μέρος, ως μέλη μιας δομής).
- Θα εμφανίζει στην οθόνη ένα μενού επιλογών, προκειμένου να επιλέξει ο χρήστης μια από τις ενέργειες:
  A. πρόσθεση,
  B. αφαίρεση,
  C. πολλαπλασιασμό,
  D. διαίρεση,
  E. τέλος των πράξεων.
- Να εμφανίζει το αποτέλεσμα της κάθε πράξης στην οθόνη με το σχετικό μήνυμα στην ελληνική γλώσσα και με ελληνικά γράμματα,
- Προσοχή, να δημιουργηθεί, αποθηκευτεί και υλοποιηθεί το εκτελέσιμο αρχείο (.EXE) σε ένα βοηθητικό μέσο (π.χ. σε ένα USB flash).
- Το πρόγραμμα απαιτείται να μεταγλωττίζεται και να εκτελείται επιτυχώς.

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

struct complex2  // Ορισμός της δομής
{
      float re;
      float im;
}info1,info2;

void result(float k, float l)
{
    /* ελέγχουμε το φανταστικό μέρος για να εμφανίσουμε σωστά το
μιγαδικό */
    if(l>=0)
            printf(" %f+%fi\n",k,l);
      else
        printf(" %f%fi\n",k,l);
      printf("\n");
}
    //συνάρτηση πρόσθεσης
int add(float x, float y, float z, float w)
{
      int k,l; //δήλωση τοπικών μεταβλητών
      k=x+y;   //υπολογισμός πραγματικού μέρους
      l=z+w;   //υπολογισμός φανταστικού μέρους
    printf("\nΤο αποτέλεσμα της πρόσθεσης είναι: ");
      result(k,l);
      return(0);
}
    //συνάρτηση αφαίρεσης
int sub(float k, float l, float m, float n)
{
      int p,o;  //δήλωση τοπικών μεταβλητών
```

```
      p=k-l;     //υπολογισμός πραγματικού μέρους
      o=m-n;     //υπολογισμός φανταστικού μέρους
    printf("\nΤο αποτέλεσμα της αφαίρεσης είναι: ");
      result(p,o);
      return(0);
}
    //συνάρτηση πολλαπλασιασμού
  int mult(float u, float v, float j, float g)
{
      float y,x;  //δήλωση τοπικών μεταβλητών
      y=u*v-j*g;  //υπολογισμός πραγματικού μέρους
      x=u*g+j*v;  //υπολογισμός φανταστικού μέρους
    printf("\nΤο αποτέλεσμα του πολλαπλασιασμού είναι: ");
      result(y,x);
      return(0);
}
    //συνάρτηση διαίρεσης
int divis(float r, float t, float f, float d)
{
      float g,h,m,n,j;//δήλωση τοπικών μεταβλητών
      g=(r*t)+(f*d);//επειδή διαίρεση δυο integer δινει μόνο το
ακέραιο υπόλυπο για να βρούμε το πραγματρικό και
      h=(f*t)-(r*d);//το φανταστικό μέρος κάνουμε τις πράξεις σπαστά
και τις αποθηκεύουμε στις float μεταβλητεσ g,h,m
      m=(t*t)+(d*d);
      n=g/m;       //υπολογισμός πραγματικού μέρους
      j=h/m;       //υπολογισμός φανταστικού μέρους
    printf("\nΤο αποτέλεσμα της διαίρεσης είναι: ");
      result(n,j);
      return(0);
}

main()
{
    system("chcp 1253");//cmd εμφάνιση ελληνικών χαρακτήρων
    char choi;     //δήλωση βοηθητικής μεταβλητής
    printf("Το πρόγραμμα δέχεται δύο μιγαδικούς αριθμούς \n");
    printf("(πραγματικό και φανταστικό μέρος, ως μέλη μιας δομής)
\n");
    printf("και εμφανίζει στην οθόνη το αποτέλεσμα απλών πράξεων.
\n\n");

      printf("Δώστε το πραγματικό μέρος του πρώτου αριθμού: ");
      scanf("%f",&info1.re);

      printf("Δώστε το φανταστικό μέρος του πρώτου αριθμού: ");
      scanf("%f",&info1.im);


      printf("Δώστε το πραγματικό μέρος του δεύτερου αριθμού: ");
      scanf("%f",&info2.re);

      printf("Δώστε το φανταστικό μέρος του δεύτερου αριθμού: ");
      scanf("%f",&info2.im);

    fflush(stdin);

      if(info1.im>0) //ελέγχουμε το φανταστικό μέρος της info1 αν
είναι θετικό ή αρνητικό για να τυπώσουμε στην οθόνη σωστά τον 1ο
μιγαδικό
```

```
            printf("\nO πρώτος αριθμός είναι:
%f+%fi\n",info1.re,info1.im);
      else
            printf("\nO πρώτος αριθμός είναι:
%f%fi\n",info1.re,info1.im);

      if(info2.im>0) //ελέγχουμε το φανταστικό μέρος της info2 αν
είναι θετικό ή αρνητικό για να τυπώσουμε στην οθόνη σωστά τον 2ο
μιγαδικό
            printf("O δεύτερος αριθμός είναι:
%f+%fi\n\n",info2.re,info2.im);
      else
            printf("O δεύτερος αριθμός είναι:
%f%fi\n\n",info2.re,info2.im);

      do
    {
        printf(" Μενου επιλογής πράξης \n");
            printf(" 1  Για την πρόσθεση \n");
            printf(" 2  Για την αφαίρεση \n");
            printf(" 3  Για πολλαπλασιασμό \n");
            printf(" 4  Για τη διαίρεση \n");
            printf(" 5  Τέλος των πράξεων \n");
            printf(" Δώστε την επιλογή σας: ");
            fflush(stdin);
            choi=getchar();
            switch(choi)
            {
            case '1':  //συνάρτηση πρόσθεσης
                  add(info1.re,info2.re,info1.im,info2.im);
                  break;
            case '2':  //συνάρτηση αφαίρεσης
                  sub(info1.re,info2.re,info1.im,info2.im);
                  break;
            case '3':  //συνάρτηση πολλαπλασιασμού
                  mult(info1.re,info2.re,info1.im,info2.im);
                  break;
            case '4':  //συνάρτηση διαίρεσης
                  divis(info1.re,info2.re,info1.im,info2.im);
                  break;
            case '5':  //Τέλος των πράξεων
                   printf("\nΠατήστε Enter για έξοδο...");
                   getchar( );
                  exit(1);
            default:   // για κάθε άλλη περίπτωση
                  printf("Λάθος επιλογή \n");
                  break;
            }
   }
    while (choi != '5');
}
```

Αποτελέσματα στην οθόνη του χρήστη:

```
Active code page: 1253
Το πρόγραμμα δέχεται δύο μιγαδικούς αριθμούς
(πραγματικό και φανταστικό μέρος, ως μέλη μιας δομής)
και εμφανίζει στην οθόνη το αποτέλεσμα απλών πράξεων.

Δώστε το πραγματικό μέρος του πρώτου αριθμού: 12
Δώστε το φανταστικό μέρος του πρώτου αριθμού: 123
Δώστε το πραγματικό μέρος του δεύτερου αριθμού: -3
Δώστε το φανταστικό μέρος του δεύτερου αριθμού: 4

Ο πρώτος αριθμός είναι: 12.000000+123.000000i
Ο δεύτερος αριθμός είναι: -3.000000+4.000000i

 Μενου επιλογής πράξης
 1  Για την πρόσθεση
 2  Για την αφαίρεση
 3  Για πολλαπλασιασμό
 4  Για τη διαίρεση
 5  Τέλος των πράξεων
 Δώστε την επιλογή σας: 1

Το αποτέλεσμα της πρόσθεσης είναι:  9.000000+127.000000i

 Μενου επιλογής πράξης
 1  Για την πρόσθεση
 2  Για την αφαίρεση
 3  Για πολλαπλασιασμό
 4  Για τη διαίρεση
 5  Τέλος των πράξεων
 Δώστε την επιλογή σας: 2

Το αποτέλεσμα της αφαίρεσης είναι:  15.000000+119.000000i

 Μενου επιλογής πράξης
 1  Για την πρόσθεση
 2  Για την αφαίρεση
 3  Για πολλαπλασιασμό
 4  Για τη διαίρεση
 5  Τέλος των πράξεων
 Δώστε την επιλογή σας: 3

Το αποτέλεσμα του πολλαπλασιασμού είναι:  -528.000000-321.000000i

 Μενου επιλογής πράξης
 1  Για την πρόσθεση
 2  Για την αφαίρεση
 3  Για πολλαπλασιασμό
 4  Για τη διαίρεση
 5  Τέλος των πράξεων
 Δώστε την επιλογή σας: 4

Το αποτέλεσμα της διαίρεσης είναι:  18.240000-16.680000i

 Μενου επιλογής πράξης
 1  Για την πρόσθεση
 2  Για την αφαίρεση
 3  Για πολλαπλασιασμό
 4  Για τη διαίρεση
 5  Τέλος των πράξεων
 Δώστε την επιλογή σας: 5

Πατήστε Enter για έξοδο...
```

Ανατρέξτε στις εικόνες που βρίσκονται στο τέλος του προηγούμενου παραδείγματος, για την αποθήκευση στο υπολογιστή σας τοπικά, μέσω του προγράμματος Dev-C++ ή Codeblocks.

Εργαστήριο 9 - Δείκτες

Παράδειγμα 1

Να κατασκευάσετε ένα πρόγραμμα, το οποίο :

1. Θα ορίζει μια ακέραια τιμή σε μια μεταβλητή
2. Θα εκτυπώνει την ακέραια τιμή
3. Θα ορίζει και θα εκτυπώνει την διεύθυνση μνήμης αυτής της τιμής (pointer)
4. Θα ορίζει και θα εκτυπώνει την τιμή της διεύθυνσης μνήμης του ερ.3
5. Θα εκτυπώνει το περιεχόμενο της τιμής της διεύθυνσης του ερ.4

```c
#include <stdio.h>
//&, *
//&-->reference operator (π.χ scanf("%d",&metavliti))
//*-->deference operator (π.χ.FILE* fp;)
int main(){
	system("chcp 1253"); //cmd εμφάνιση ελληνικών χαρακτήρων
	int gazis;//int:integer --> 4 bytes
	gazis=21;
	printf("Η μεταβλητη gazis=%d \n",gazis);

	int *ptrgazis;//ενδεικτική ονομασία(π.χ. pgazis, κτ)
	//το ptr ΔΕΝ ειναι δεσμεθμένη λέξη
	printf("Αρχικά,pointer ptrgazis=%d \n",ptrgazis);

	ptrgazis = &gazis; //reference  operator-->διευθυνση μνημης(pointer)
	printf("Μετέπειτα,pointer ptrgazis=%d \n",ptrgazis);
	*ptrgazis =55;    //dereference operator-->περιεχομενο μνημης
	printf("Η μεταβλητη gazis=%d \n",gazis);
	printf("Η μεταβλητη &gazis=%d \n",&gazis);

	printf("A, ptrgazis=%d \n",ptrgazis);
	printf("B, *ptrgazis=%d \n",*ptrgazis);
	printf("Η μεταβλητη gazis=%d \n",gazis);



	return 0;}
```

Αποτελέσματα στην οθόνη του χρήστη:

```
Active code page: 1253
Η μεταβλητη gazis=21
Αρχικά,pointer ptrgazis=0
Μετέπειτα,pointer ptrgazis=6487572
Η μεταβλητη gazis=55
Η μεταβλητη &gazis=6487572
A, ptrgazis=6487572
B, *ptrgazis=55
Η μεταβλητη gazis=55
```

Αναφορικά με το παραπάνω παράδειγμα, ακολουθεί μια αυθέραιτη απεικόνιση του τρόπου αποθήκευσης στη μνήμη μερικών εντολών του Παραδείγματος 1:

<table>
<tr><th>Εντολή στη C</th><th>Σχηματική Απεικόνιση στη C</th></tr>
<tr><td>-

(γενική αναπαράσταση μνήμης)</td><td>
<table>
<tr><td>…</td><td>150</td><td>151</td><td>152</td><td>153</td><td>154</td><td>155</td><td>156</td><td>157</td><td>158</td><td>159</td><td>…</td></tr>
<tr><td>…</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>…</td></tr>
</table>
</td></tr>
<tr><td><code>int gazis</code>

<code>//int:integer-> 4 bytes</code></td><td>
<table>
<tr><td>…</td><td>150</td><td>151</td><td>152</td><td>153</td><td>154</td><td>155</td><td>156</td><td>157</td><td>158</td><td>159</td><td>…</td></tr>
<tr><td>…</td><td>////</td><td>////</td><td>////</td><td>////</td><td></td><td></td><td></td><td></td><td></td><td></td><td>…</td></tr>
</table>
Όπου //// αναπαραστά δεσμευμένο κελί μνήμης
</td></tr>
<tr><td><code>int gazis</code>

<code>//int:integer-> 4 bytes</code></td><td>
<table>
<tr><td>…</td><td>150</td><td>151</td><td>152</td><td>153</td><td>154</td><td>155</td><td>156</td><td>157</td><td>158</td><td>159</td><td>…</td></tr>
<tr><td>…</td><td><code>gazis</code></td><td>////</td><td>////</td><td>////</td><td></td><td></td><td></td><td></td><td></td><td></td><td>…</td></tr>
</table>
Όπου η μεταβλητή gazis καταλαμβάνει 4 θέσεις μνήμης από το 1ο κελί 150 έως και 153 (δηλαδή, 4 κελιά)
</td></tr>
<tr><td><code>gazis=21</code></td><td>
<table>
<tr><td>…</td><td>150</td><td>151</td><td>152</td><td>153</td><td>154</td><td>155</td><td>156</td><td>157</td><td>158</td><td>159</td><td>…</td></tr>
<tr><td>…</td><td><code>gazis=21</code></td><td>////</td><td>////</td><td>////</td><td></td><td></td><td></td><td></td><td></td><td></td><td>…</td></tr>
</table>
</td></tr>
<tr><td>Ισοδύναμα:</td><td>Ισοδύναμα:</td></tr>
</table>

`gazis=21`

...	150	151	152	153	154	155	156	157	158	159	...
...	21	////	////	////							...

`int *ptrgazis`

...	150	151	152	153	154	155	156	157	158	159	...
...	gazis=21	////	////	////	ptrgazis	////	////	////			...

`ptrgazis= &gazis`

...	150	151	152	153	154	155	156	157	158	159	...
...	gazis=21	////	////	////	ptrgazis=55	////	////	////			...

Ισοδύναμα:

`ptrgazis= &gazis`

Ισοδύναμα:

...	150	151	152	153	154	155	156	157	158	159	...
...	21	////	////	////	150	////	////	////			...

Δηλαδή, αποθηκεύεται η διεύθυνση της μεταβλητής gazis στο περιεχόμενο της διεύθυνσης ptrgazis !!!

`printf("%d",gazis);`	Εκτύπωση → 21
`printf("%d",&gazis);`	Εκτύπωση → 150 (reference operator)
`printf("%d",ptrgazis);`	Εκτύπωση → 150 (διεύθυνση μνήμης μεταβλητής gazis)
`printf("%d",*ptrgazis);`	Εκτύπωση → 21 (dereference operator, δηλαδή τιμή της διεύθυνσης μνήμης)

Τέλος, αναφορικά με το παραπάνω παράδειγμα τονίζεται ότι η αρίθμιση των κελιών μνήμης από το 150… κτλ είναι αυθέραιτη. Κατά την εκτέλεση του παραδείγματος στον υπολογιστή σας δύναται να αποθηκευτούν σε άλλα κελιά μνήμης οι μεταβλητές σας. Το πιο σημαντικό της παραπάνω αυθαίρετης σχηματικης απεικόνισης είναι ότι τα κελιά μνήμης είναι:

1. διαδοχικοί ακέραιοι αριθμοι (150,151,152)
2. ανεξαρτήτως της τιμής που θα εισαχθεί σε ένα τύπο μεταβλητής, στην μνήμη δεσμεύονται πάντα όλα τα bytes
   (υπενθύμιση: αν δεν γνωρίζουμε το πλήθος των bytes μιας μεταβλητής, χρησιμοποιούμε την εντολή sizeof(τυπος_μεταβλητης) η οποία υπολογίζει το μέγεθος σε bytes. Για περισσότερες πληροφιρίες αναφορικά με την εντολή δείτε: Εργαστήριο 5, Παράδειγμα 2)

Παράδειγμα 2

Να κατασκευάσετε ένα πρόγραμμα, το οποίο :

1. Θα δέχεται από τον χρήστη μια ΑΚΕΡΑΙΑ τιμή, θα την αποθηκεύει σε μια μεταβλητή και θα εκτυπώνει το κελί μνήμης (address) που έχει αποθηκευτεί.
2. Θα έχει αποθηκευμένο έναν πίνακα, με τις εξής τιμές: {-4,-3,-2,-1,0,1,2,3,4} και θα εκτυπώνει για το κάθε στοιχείο του πίνακα το κελί μνήμης καθώς και την τιμή του, στην οθόνη του χρήστη.
3. Θα εκτυπώνει για το 1ο στοιχείο του πίνακα την τιμή και την διεύθυνσή του (κελί μνήμης-pointer)

Το πρόγραμμα απαιτείται να μεταγλωττίζεται και να εκτελείται επιτυχώς .

```
#include <stdio.h>
main()
{
        int inputXristi; //erwtima 1
        int i; //erwtima 2
        int Array[9]={-4,-3,-2,-1,0,1,2,3,4}; //erwtima 2

        //=============ERWTIMA 1=============
        printf("Eisigage mia thetiki timi\n");
        scanf("%d", &inputXristi);

        //1os tropos
        printf("I timi tu xristi einai: %d \n",inputXristi);

        //2os tropos: dereference pointer (timi enos pointer)
        // int *pinputXristi=&inputXristi;
        // printf("I timi tu xristi einai: %d \n",*pinputXristi);

        //1os tropos
        printf("I eisodos tu xristi exei apothikeutei "
        "sti dieuthinsi mninmis:%p kai exei timi: %d \n",&inputXristi, inputXristi);
        printf("\n ======================================== \n");

        //2os tropos: dereference pointer (timi enos pointer)
        //  printf("I eisodos tu xristi exei apothikeutei "
        // "sti dieuthinsi mninmis:%p kai exei timi: %d \n",&inputXristi, *pinputXristi);

        //=============ERWTIMA 2=============
        printf("Stoixeio\t Timi\t Keli Mnimis\n");
        for (i=0; i<9; i=i+1)
        printf("Stoixeio[%d]\t %d \t %p\n",i,Array[i],&Array[i]);
        printf("\n ======================================== \n");

        //=============ERWTIMA 3=============
```

```
        ///kathe pointer exei to 1o stoixeio enos pinaka(array)
        printf("\n O pointer tu pinaka mas: %p \n",Array);
        printf("I timi tu 1ou stioixeiou tu pinaka: %d \n",*Array);
        //printf("To 3o stoixeio tou pinaka einai:%d \n",*(Array+2));
}
```

Αποτελέσματα στην οθόνη του χρήστη:

```
I eisodos tu xristi exei apothikeutei sti dieuthinsi mninmis:000000000062FE18
kai exei timi: 21

 ======================================
Stoixeio         Timi    Keli Mnimis
Stoixeio[0]      -4      000000000062FDF0
Stoixeio[1]      -3      000000000062FDF4
Stoixeio[2]      -2      000000000062FDF8
Stoixeio[3]      -1      000000000062FDFC
Stoixeio[4]      0       000000000062FE00
Stoixeio[5]      1       000000000062FE04
Stoixeio[6]      2       000000000062FE08
Stoixeio[7]      3       000000000062FE0C
Stoixeio[8]      4       000000000062FE10

 ======================================

 O pointer tu pinaka mas: 000000000062FDF0
I timi tu 1ou stioixeiou tu pinaka: -4
```

Παράδειγμα 3

Να κατασκευάσετε ένα πρόγραμμα, το οποίο :

1. Θα δέχεται από τον χρήστη μια ΑΚΕΡΑΙΑ τιμή, θα την αποθηκεύει σε μια μεταβλητή και θα εκτυπώνει το κελί μνήμης (address) που έχει αποθηκευτεί.
2. Θα έχει αποθηκευμένους δυο πίνακες, με τις εξής τιμές:
   {-3,-2,-1,0,1,2,3},{-3.3,2.9,3.5,1,11.4}
   και θα εκτυπώνει για το κάθε στοιχείο του πίνακα το κελί μνήμης καθώς και την τιμή του, στην οθόνη του χρήστη.
3. Θα εκτυπώνει για το 1ο στοιχείο του πίνακα την τιμή και την διεύθυνσή του (κελί μνήμης-pointer)

Το πρόγραμμα απαιτείται να μεταγλωττίζεται και να εκτελείται επιτυχώς .

```
 #include <stdio.h>
//&-->διευθυνση μνημης(pointer)-->reference operator (π.χ
scanf("%d",&metavliti))
//*-->περιεχομενο μνημης       -->deference opertor  (π.χ.FILE* fp;)
int main (){
 system("chcp 1253"); //cmd εμφάνιση ελληνικών χαρακτήρων //ερωτημα1
 int timiXristi ;
 printf("Παρακαλω να εισαχθει μια ακέραια τιμή:\n");
 scanf("%d",&timiXristi);
 printf("Τιμή χρήστη->  timiXristi=  %d \n" ,timiXristi);
 printf("Τιμή χρήστη->  &timiXristi=  %d  \n",&timiXristi);
 printf(">>Τιμή χρήστη-> &timiXristi= %p  \n",&timiXristi);//hex:δεκαεξαδικη αναπαράσταση
 //ερωτημα2
 int    pinakas1[7]={-3,-2,-1,0,1,2,3};
 //7 στοχεία δλδ 0,1,2,3,4,5,6
 float  pinakas2[5]={-3.3,2.9,3.5,1,11.4};
 //5 στοιχεία δλδ 0,1,2,3,4
 double pinakas3[5]={-3.3,2.9,3.5,1,11.4};
 //5 στοιχεία δλδ 0,1,2,3,4  //επιπρόσθετο πχ εργαστηριου για επεξήγηση double-float

 int i;

 printf("Πινακας 1(integer, 4bytes):\n");
 for (i=0;i<7;i=i+1){
  printf("(Στοιχείο,τιμή)= (i,pinakas1[i])=(%d,%d) \n",i,pinakas1[i]);
  printf("Κελί μνήμης= %d \n",&pinakas1[i]);
//printf("Κελί μνήμης= %p \n",&pinakas1[i]);
 }

 printf("\n================\n");

 printf("Πινακας 2(float, 4bytes):\n");
 for (i=0;i<5;i=i+1){
   printf("(Στοιχείο,τιμή)= (i,pinakas2[i])=(%d,%.2f) \n",i,pinakas2[i]);
   printf("Κελί μνήμης= %d \n",&pinakas2[i]);
 //printf("Κελί μνήμης= %p \n",&pinakas2[i]);
 }

 printf("\n================\n");
```

```c
 //επιπρόσθετο πχ εργαστηριου για επεξήση χωριτικότητας και μνήμης στην C
 //int       -->4 byte
 //float     -->4 byte
 //double    -->8 byte
 printf("Πινακας 3(double, 8bytes):\n");
 for (i=0;i<5;i=i+1){
   printf("(Στοιχείο,τιμή)= (i,pinakas3[i])=(%d,%.2f) \n",i,pinakas3[i]);
   printf("Κελί μνήμης= %d \n",&pinakas3[i]);
 //printf("Κελί μνήμης= %p \n",&pinakas3[i]);
 }

//ερωτημα3
 printf("Η τιμή του 1ου στοιχείο του πίνακα1 ειναι %d \n",*pinakas1);
 printf("Α τροπος,ο δείκτης(pointer) του 1ου στοιχείο του πίνακα1:%d\n",pinakas1); //%p
 printf("Β τροπος,ο δείκτης(pointer) του 1ου στοιχείο του πίνακα1: %d \n",&pinakas1[0]);//%p

 printf("Η τιμή του 3ου στοιχείο του πίνακα2 ειναι %f \n",  pinakas2[2]);
 printf("Η τιμή του 3ου στοιχείο του πίνακα2 ειναι %f \n",*(pinakas2+2));
 printf("Η διεύθυνση του 3ου στοιχείο του πίνακα2 ειναι %d \n", &pinakas2[2]);//%p
 return 0;
}
```

Αποτελέσματα στην οθόνη του χρήστη:

```
Active code page: 1253
Παρακαλω να εισαχθει μια ακέραια τιμή:
2020
Τιμή χρήστη->   timiXristi=  2020
Τιμή χρήστη->  &timiXristi=  6487576
>>Τιμή χρήστη-> &timiXristi= 000000000062FE18
Πινακας 1(integer, 4bytes):
(Στοιχείο,τιμή)= (i,pinakas1[i])=(0,-3)
Κελί μνήμης= 6487536
(Στοιχείο,τιμή)= (i,pinakas1[i])=(1,-2)
Κελί μνήμης= 6487540
(Στοιχείο,τιμή)= (i,pinakas1[i])=(2,-1)
Κελί μνήμης= 6487544
(Στοιχείο,τιμή)= (i,pinakas1[i])=(3,0)
Κελί μνήμης= 6487548
(Στοιχείο,τιμή)= (i,pinakas1[i])=(4,1)
Κελί μνήμης= 6487552
(Στοιχείο,τιμή)= (i,pinakas1[i])=(5,2)
Κελί μνήμης= 6487556
(Στοιχείο,τιμή)= (i,pinakas1[i])=(6,3)
Κελί μνήμης= 6487560

================
Πινακας 2(float, 4bytes):
(Στοιχείο,τιμή)= (i,pinakas2[i])=(0,-3.30)
Κελί μνήμης= 6487504
(Στοιχείο,τιμή)= (i,pinakas2[i])=(1,2.90)
Κελί μνήμης= 6487508
(Στοιχείο,τιμή)= (i,pinakas2[i])=(2,3.50)
Κελί μνήμης= 6487512
(Στοιχείο,τιμή)= (i,pinakas2[i])=(3,1.00)
Κελί μνήμης= 6487516
(Στοιχείο,τιμή)= (i,pinakas2[i])=(4,11.40)
Κελί μνήμης= 6487520

================
Πινακας 3(double, 8bytes):
(Στοιχείο,τιμή)= (i,pinakas3[i])=(0,-3.30)
Κελί μνήμης= 6487456
(Στοιχείο,τιμή)= (i,pinakas3[i])=(1,2.90)
Κελί μνήμης= 6487464
(Στοιχείο,τιμή)= (i,pinakas3[i])=(2,3.50)
Κελί μνήμης= 6487472
(Στοιχείο,τιμή)= (i,pinakas3[i])=(3,1.00)
Κελί μνήμης= 6487480
(Στοιχείο,τιμή)= (i,pinakas3[i])=(4,11.40)
Κελί μνήμης= 6487488
Η τιμή του 1ου στοιχείο του πίνακα1 ειναι -3
Α τροπος,ο δείκτης(pointer) του 1ου στοιχείο του πίνακα1:6487536
Β τροπος,ο δείκτης(pointer) του 1ου στοιχείο του πίνακα1: 6487536
Η τιμή του 3ου στοιχείο του πίνακα2 ειναι 3.500000
Η τιμή του 3ου στοιχείο του πίνακα2 ειναι 3.500000
Η διεύθυνση του 3ου στοιχείο του πίνακα2 ειναι 6487512
```

Παράδειγμα 4

Να κατασκευάσετε ένα πρόγραμμα, το οποίο :

1. Θα συμπληρώνει δύο πίνακες χαρακτήρων με τα είκοσι έξι (26) κεφαλαία γράμματα της λατινικής αλφαβήτου (A,B,C,D,... Y,Z):

   i. Για το πρώτο πίνακα θα χρησιμοποιεί δεικτοποίηση πινάκων.
   ii. Για το δεύτερο πίνακα θα χρησιμοποιεί δείκτες.

2. Θα εκτυπώνει τους δυο πίνακες στην οθόνη, οι οποίοι:

   i. Θα καταλαμβάνουν ο κάθε ένας από μια ΜΟΝΟ γραμμή.
   ii. Ανάμεσα σε δύο γράμματα θα υπάρχει ένα κενό διάστημα.

Το πρόγραμμα απαιτείται να μεταγλωττίζεται και να εκτελείται επιτυχώς.

```
#include <stdio.h>


int main()
{
   system("chcp 1253"); //cmd εμφάνιση ελληνικών χαρακτήρων
   int x;
   char w,k;
   /* δήλωση των δύο πινάκων χαρακτηρων 26x1 */

   /* συμπλήρωμα 1ου πίνακα με τα κεφαλαία 26 αγγλικά γράμματα- δεικτόδότηση πινάκων*/
   char A[26];
   for(w='A', x=0;w<='Z';w++, x++)
      A[x]=w;

   /* συμπλήρωμα 2ου πίνακα με τα κεφαλαία 26 αγγλικά γράμματα- δεικτές*/
      char B[26];
      for(w='A', x=0;w<='Z';w++, x++)
      {
         *(B+x)=w;
      }
      B[26]='\0';

   /*  εμφανίζονται οι δύο πίνακεςστην οθόνη και να καταλαμβάνουν μόνο μια γραμμή της
οθόνης ο κάθε ένας και ανάμεσα σε δύο γράμματα να υπάρχει ένα κενό διάστημα*/
```

```
    printf("Table A Table A --πινακας A-- ((indexing tables):  \n");
    for(x=0;x<26;x++)
        printf("%c ", A[x]);

    /*  εμφανίζονται οι δύο πίνακεςστην οθόνη και να καταλαμβάνουν μόνο μια γραμμή της
οθόνης ο κάθε ένας και ανάμεσα σε δύο γράμματα να υπάρχει ένα κενό διάστημα*/
    printf("\n\nTable B --πινακας B-- ( (indicators):  \n");
    x=0;
    while(*(B+x)!='\0')
    {
        printf("%c ", *(B+x));
        x++;
    }
    printf("\n\n");
          return 0;
}
```

Αποτελέσματα στην οθόνη του χρήστη:

```
Active code page: 1253
Table A Table A --πινακας A-- ((indexing tables):
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Table B --πινακας B-- ( (indicators):
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
```

Βιβλιογραφία

- Αλέξανδρος Σ. Καράκος, Οδηγός Προγραμματισμού με τη γλώσσα C, Εκδόσεις: Καράκος, 2017
- Αλέξανδρος Σ. Καράκος, Εισαγωγή στη Γλώσσα C με παραδείγματα και ασκήσεις (Β' έκδοση), Εκδόσεις: Καράκος, 2012
- Αλέξανδρος Σ. Καράκος, Αλγοριθμική επίλυση ασκήσεων με τη γλώσσα C, Εκδόσεις: Καράκος, 2009
- Σπύρος Γερούλης, Εισαγωγή στη C, Εκδόσεις: Spin, 2006
- Webber Adam Brooks, Σύγχρονες Γλώσσες Προγραμματισμού, μετάφραση Γεωργακόπουλος Γεώργιος, Παπαδόγγονας Φρ. Ιωάννης, Εκδόσεις: Πανεπιστημιακές Εκδόσεις Κρήτης, 2009
- Shaw Zed A., Learn C the Hard Way, Εκδόσεις: Zed Shaw's Hard Way Series, 2015
- Paul Deitel, Harvey Deitel, μετάφραση Μήλιος Αγαμέμνωνας, C Προγραμματισμός, Εκδόσεις: Μ.Γκιούρδας, 2014
- Kernighan Brian W., Ritchie Dennis M., Η Γλώσσα Προγραμματισμού C, Prentice Hall, Εκδόσεις: Κλειδάριθμος, 2011

Ηλεκτρονικές Αναφορές

- C Programming Tutorial – TutorialsPoint, https://www.tutorialspoint.com/cprogramming/
- ChIntegrated Development Environment (ChIDE), https://www.softintegration.com/docs/ch/chide/
- CppCode - offline C/C++ IDE & Compiler on the App Store - iTunes – Apple, https://itunes.apple.com/us/app/cppcode-offline-c-c-ide-compiler/id936694712?mt=8
- Dcoder, Mobile Compiler IDE - Εφαρμογές Android στο Google Play, https://play.google.com/store/apps/details?id=com.paprbit.dcoder&hl=e
- Dev-C++ | ΕΛ/ΛΑΚ για την Τριτοβάθμια Εκπαίδευση, https://opensci.grnet.gr/os_catalog/softwares/dev-c
- Dev-C++ download | SourceForge.net, https://sourceforge.net/projects/orwelldevcpp
- Download - 7-Zip,
- http://www.7-zip.org/download.html
- DUTHNET eClass, https://eclass.duth.gr/

- iZip Archiver on the Mac App Store - iTunes – Apple, https://itunes.apple.com/us/app/izip-archiver/id478738838?mt=12
- The Unarchiver on the Mac App Store - iTunes – Apple, https://itunes.apple.com/app/the-unarchiver/id425424353?mt=12&ls=1
- Xcode on the Mac App Store - iTunes – Apple, https://itunes.apple.com/us/app/xcode/id497799835?mt=12
- ZArchiver - Εφαρμογές Android στο Google Play, https://play.google.com/store/apps/details?id=ru.zdevs.zarchiver&hl=el