

ΔΗΜΟΚΡΙΤΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΡΑΚΗΣ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ



Μάθημα:

ΔΟΜΗΜΕΝΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

Διδάσκουσα καθηγήτρια:

Ελευθερία Κατσίρη

ΕΡΓΑΣΤΗΡΙΑΚΟ ΕΓΧΕΙΡΙΔΙΟ ΣΤΗ C

Αλέξανδρος Γαζής

πτυχιούχος ΗΜΜΥ ΔΠΘ,
μεταπτυχιακός φοιτητής ΔΠΘ

Κωσταντίνος Σταμάτης

πτυχιούχος ΗΜΜΥ ΔΠΘ,
μεταπτυχιακός φοιτητής ΔΠΘ

ΞΑΝΘΗ, ΟΚΤΩΒΡΙΟΣ 2017

► Το παρόν εγχειρίδιο δημιουργήθηκε με σκοπό την παροχή επεξηγήσεων και την καθοδήγηση των προπτυχιακών φοιτητών του Δημοκριτείου Πανεπιστημίου Θράκης, αναφορικά με το εργαστηριακό τμήμα του μαθήματος «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

Υπεύθυνος εργαστηρίου: *Αλέξανδρος Γαζής*, agazis@ee.duth.gr

Βοηθός εργαστηρίου: *Κωσταντίνος Σταμάτης*, kstamati@ee.duth.gr

► Τέλος, θα θέλαμε να ευχαριστήσουμε τον κ. Καράκο Αλέξανδρο, αφυπηρετήσαντα καθηγητή του τμήματος ΗΜΜΥ του Δημοκριτείου Πανεπιστημίου Θράκης καθώς και τον κ. Συμεωνίδη Σίμο, υποψήφιο διδάκτορα του τμήματος ΗΜΜΥ, για την συνεισφορά τους.

ΠΕΡΙΕΧΟΜΕΝΑ

	σελ.
Εισαγωγή	5
1. Εγγραφή στο μάθημα μέσω της πλατφόρμας του eclass.....	5
2. Παρουσίαση Εργαστηριακών Εργαλείων και Πρώτο Πρόγραμμα	7
Εργαστήριο 1- Μεταβλητές και Βασικές Εντολές Εκτύπωσης Τιμών.....	11
1. 1 ^ο πρόγραμμα υλοποίησης - «Γεια σου Κόσμε».....	12
2. Υλοποίηση αποτελέσματος οθόνης στην Γραμμή Εντολών (των Windows).....	13
3. Μεταβλητές/Τύποι Δεδομένων	14
4. Αριθμητικοί και Λογικοί Τελεστές	15
5. Εντολές Εξόδου Δεδομένων.....	16
6. Μετατροπές σε Μεταβλητές/Τύπους Δεδομένων	18
7. C: Case sensitive	18
Παραδείγματα Εργαστηρίου	19
Εργαστήριο 2 - Εντολές ελέγχου και Επανάληψης.....	23
1. Εντολές Λογικού Ελέγχου (AN)	23
2. Δομές Επανάληψεων	24
Εργαστήριο 3 - Δομές Επανάληψεων	28
Εργαστήριο 4 - Συναρτήσεις.....	31
Εργαστήριο 5 - Αναδρομή Συναρτήσεων και Επαναληπτική Λειτουργία	36
Εργαστήριο 6 - Δείκτες.....	39
Εργαστήριο 7 & 8- Αρχεία	42
Εργαστήριο 9 - Ειδικές εντολές-Αναδρομή-Επανάληψη.....	48
Βιβλιογραφία	50
Ηλεκτρονικές Αναφορές	50

ΠΙΝΑΚΑΣ ΕΙΚΟΝΩΝ

Εικόνα 1 Βήμα 1: εισαγωγή ηλεκτρονικής διεύθυνσης σε περιηγητή	6
Εικόνα 2 Βήμα 2: εισαγωγή στοιχείων στην Σύνδεση χρήστη	6
Εικόνα 3 Βήμα 3: επιλογή αριστερά Μαθήματα.....	7
Εικόνα 4 Βήμα 4: επιλογή Προπτυχιακό.....	7
Εικόνα 5 Βήμα 5: επιλέγουμε και εγγραφόμαστε στο μάθημα	7
Εικόνα 6 Περιβάλλον εργασίας διαδικτυακού εργαλείου προγραμματιστικού κώδικα	8
Εικόνα 7 Βασικό παράδειγμα υλοποίησης προγράμματος σε γλώσσα C στο εργαλείο Dev-C++	9
Εικόνα 8 Λειτουργικά Συστήματα Εγκατάστασης ChIDE	10
Εικόνα 9 Βασικό παράδειγμα υλοποίησης προγράμματος σε γλώσσα C στο εργαλείο ChIDE.....	10
Εικόνα 10 Διαδικασία υλοποίησης για την εξαγωγή αποτελεσμάτων στην οθόνη του χρήστη (και παράδειγμα λειτουργίας σε αρχεία Java)	11
Εικόνα 11 Διάγραμμα ροής δομής πολλαπλής επανάληψης (switch)	31
Εικόνα 12 Βήμα 1 επιλέγω Parameters.....	43
Εικόνα 13 Βήμα 2 ορίζω την παράμετρο. Προσοχή στο γράμμα/διεύθυνση που θα ορίσετε στο πρόγραμμά σας	43
Εικόνα 14 Βήμα 1 επιλέγω arguments, για το πρόγραμμα που χρησιμοποιώ.....	44
Εικόνα 15 Βήμα 2 ορίζω την αποθηκευτική μονάδα. Προσοχή στο γράμμα/διεύθυνση που θα ορίσετε στο πρόγραμμά σας	44

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. Στην 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

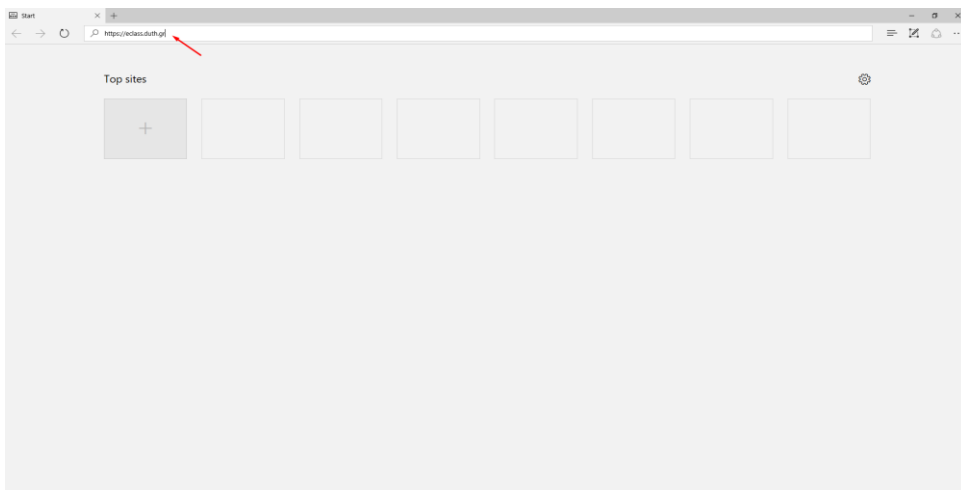
.

Οποιαδήποτε επικοινωνία με τον διδάσκοντα ή τον υπεύθυνο εργαστηρίου πραγματοποιείται μέσω του email του πανεπιστημίου.

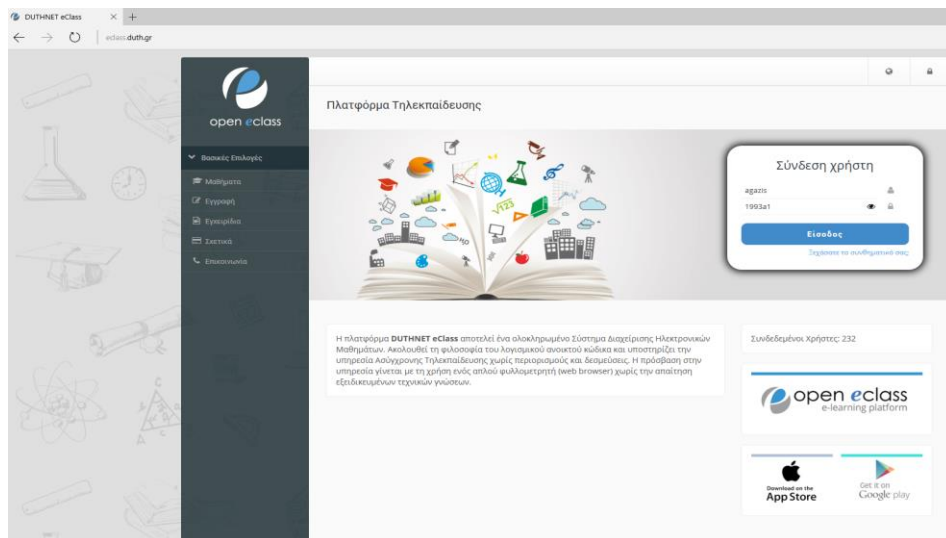
3. Στο νέο παράθυρο, όπου μεταφέρεστε, μετά την επιτυχή σύνδεση, αναζητείτε στις επιλογές, στο αριστερό τμήμα της οθόνης την καρτέλα «Μαθήματα».
4. Στην συνέχεια, επιλέγετε τον σύνδεσμο για το προπτυχιακό επίπεδο (με γαλάζια γράμματα).

- 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.

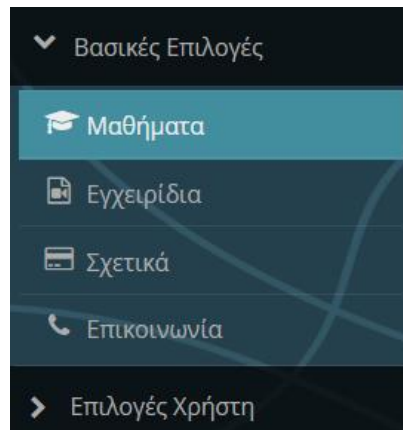
Ακολουθεί σχηματική απεικόνιση των παραπάνω βημάτων:




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

Σχολή - Τμήμα: Δημοκρίτειο Πανεπιστήμιο Θράκης » Πολυτεχνικής Σχολή » Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Άλλο (TMA) - 6 διαθέσιμα μαθήματα
Μεταπτυχιακό (TMA) - 39 διαθέσιμα μαθήματα
Προπτυχιακό (TMA) - 181 διαθέσιμα μαθήματα 

Εικόνα 4 Βήμα 4: επιλογή Προπτυχιακό

Δομημένος Προγραμματισμός (2017-2018) (TMA555)
Ελ. Κατσίρη

Εικόνα 5 Βήμα 5: επιλέγουμε και εγγραφόμαστε στο μάθημα

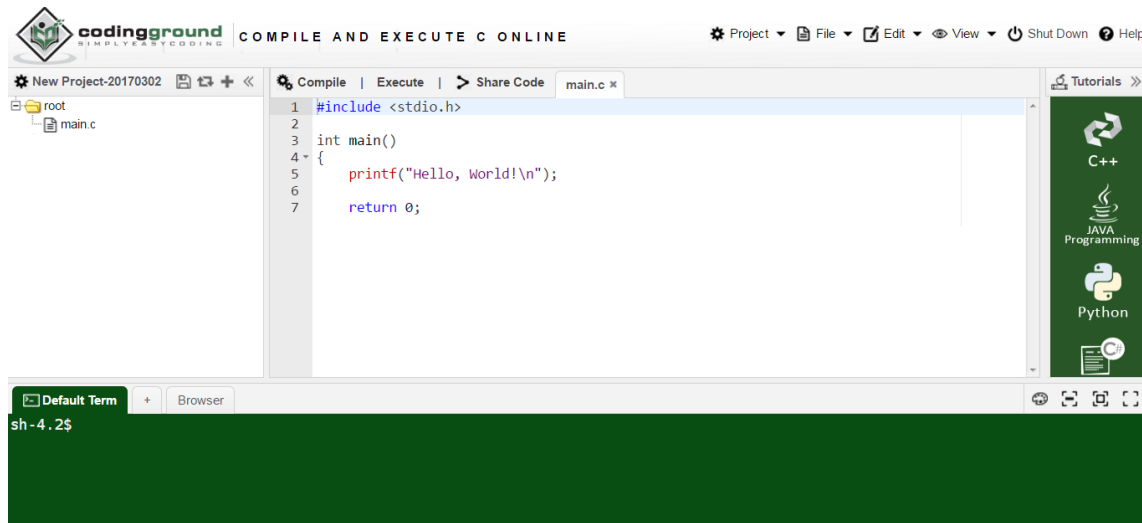
2. Παρουσίαση Εργαστηριακών Εργαλείων και Πρώτο Πρόγραμμα

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):

1. Οδηγίες για την εφαρμογή Dcoder (Android)
download - λειτουργία - αποθήκευση για Android συσκευές

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

Εργαλεία για Mac OS- iOS
Προτεινόμενοι compilers και προγράμματα διαχείρισης συμπιεσμένων αρχείων

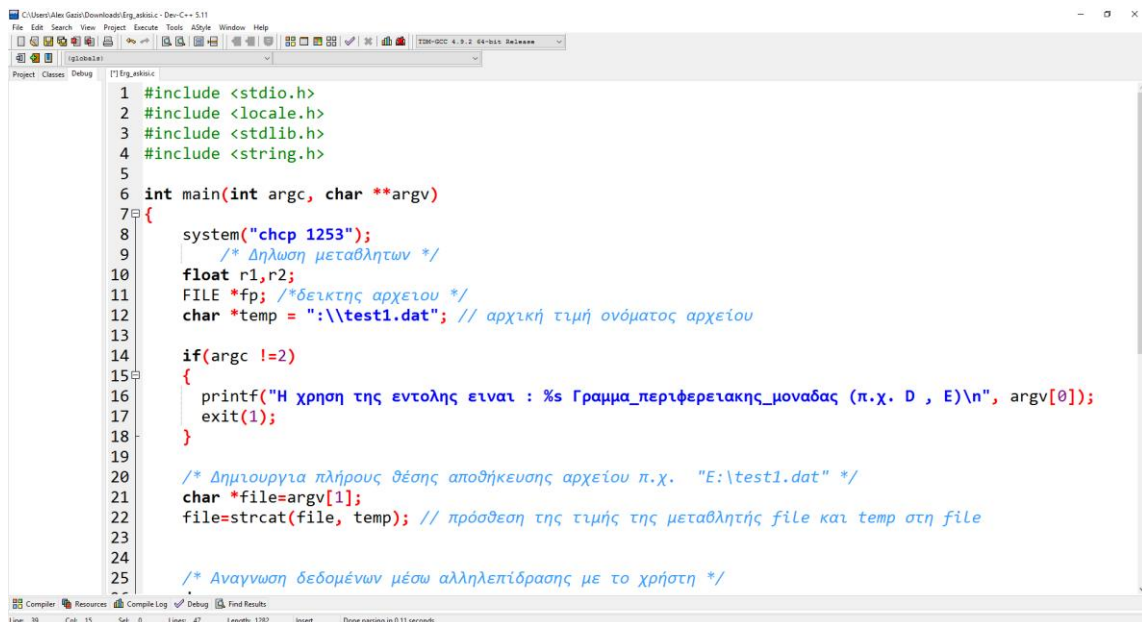
❑ Χρήση Τοπικών Εργαλείων - Dev-C++ και ChIDE

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.



```

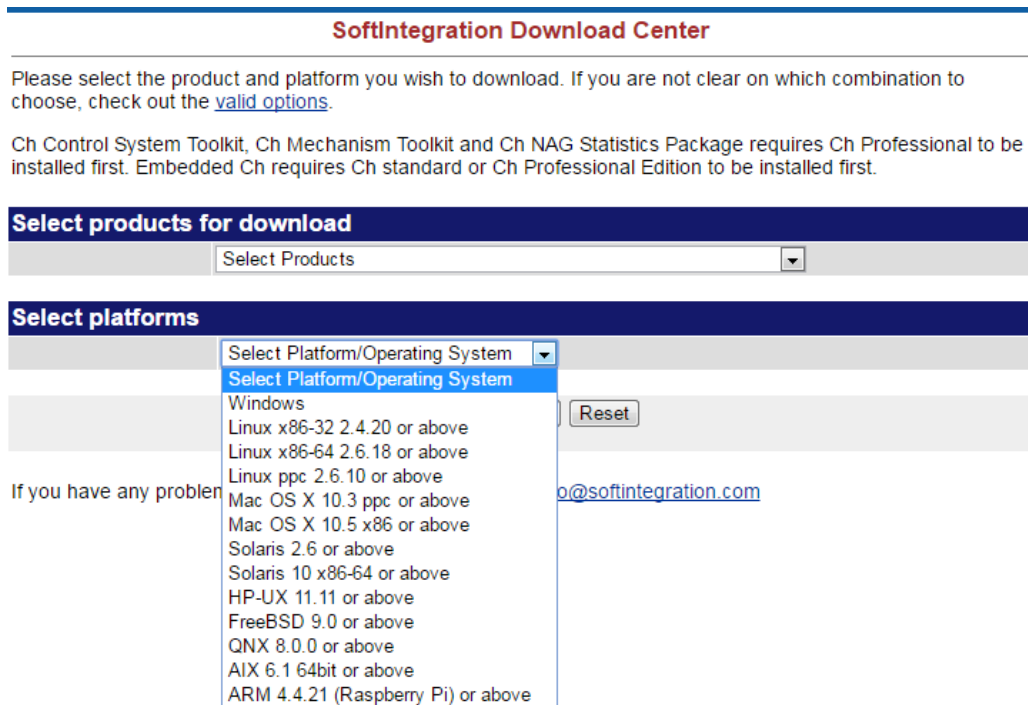
1 #include <stdio.h>
2 #include <locale.h>
3 #include <stdlib.h>
4 #include <string.h>
5
6 int main(int argc, char **argv)
7 {
8     system("chcp 1253");
9     /* Δηλώση μεταβλητών */
10    float r1,r2;
11    FILE *fp; /*δευκτης αρχείου */
12    char *temp = "\\test1.dat"; // αρχική τιμή ονόματος αρχείου
13
14    if(argc !=2)
15    {
16        printf("Η χρήση της εντολής είναι : %s Γράμμα_περιφερειακής_μονάδας (π.χ. D , E)\n", argv[0]);
17        exit(1);
18    }
19
20    /* Δημιουργία πλήρους θέσης αποθήκευσης αρχείου π.χ. "E:\test1.dat" */
21    char *file=argv[1];
22    file=strcat(file, temp); // πρόσθεση της τιμής της μεταβλητής file και temp στη file
23
24
25    /* Ανάγνωση δεδομένων μέσω αλληλεπίδρασης με το χρήστη */

```

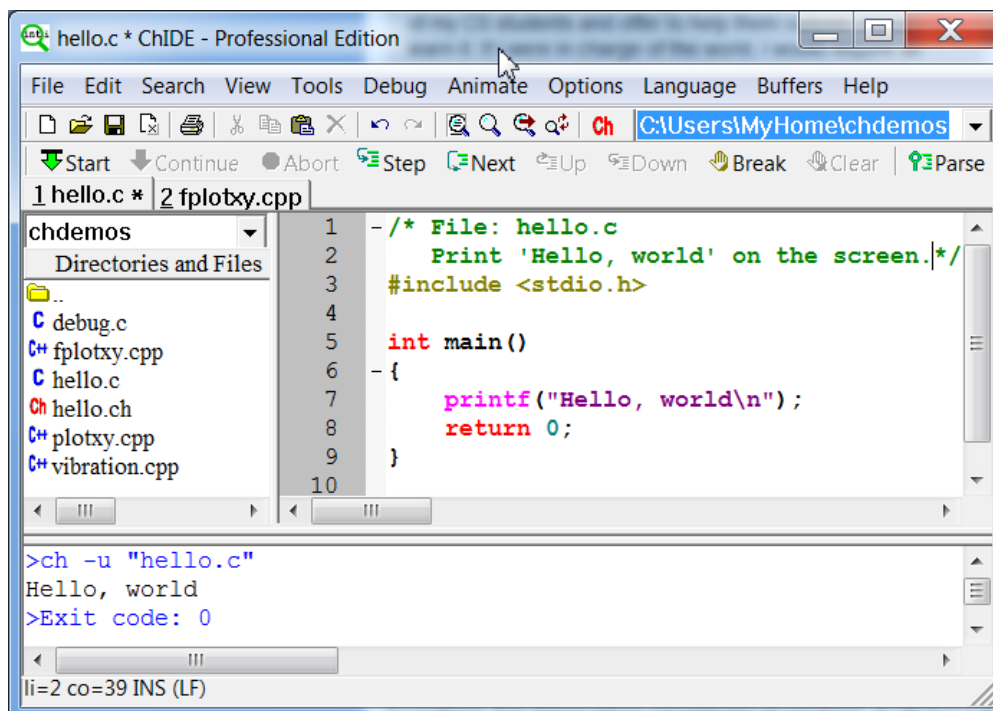
Εικόνα 7 Βασικό παράδειγμα υλοποίησης προγράμματος σε γλώσσα C στο εργαλείο Dev-C++

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 Raspberry Pi.



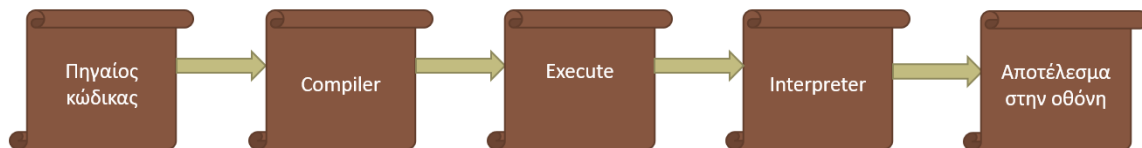
Εικόνα 8 Λειτουργικά Συστήματα Εγκατάστασης ChIDE



Εικόνα 9 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 10 Διαδικασία υλοποίησης για την εξαγωγή αποτελεσμάτων στην οθόνη του χρήστη (και παράδειγμα λειτουργίας σε αρχεία Java)

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:

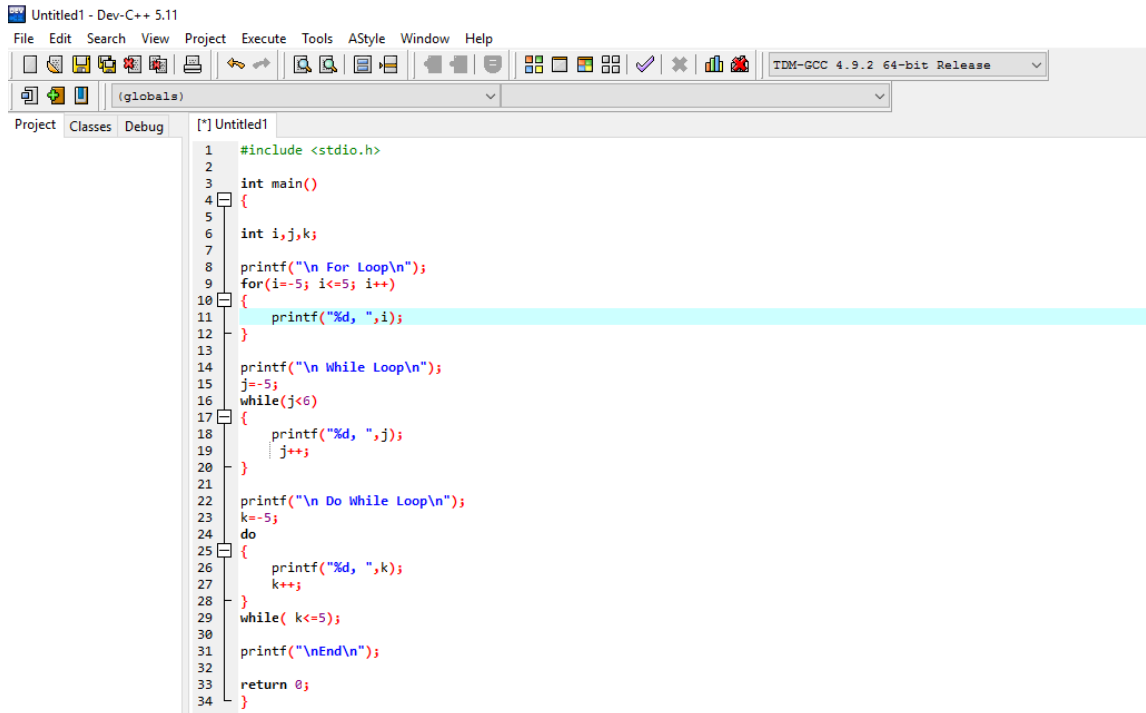
- Compiler: πρόγραμμα που μετατρέπει το πηγαίο κώδικα από μια γλώσσα προγραμματισμού σε μια άλλη (με σκοπό να τρέξουμε ένα πρόγραμμα).
- Interpreter: πρόγραμμα που εκτελεί *μόνο ένα* κώδικα ή πρόγραμμα.

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 11 Εγκατάσταση DevC++



Εικόνα 12 Περιβάλλον DevC++

1. 1^ο πρόγραμμα 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. Υλοποίηση αποτελέσματος οθόνης στην 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.

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`.

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, dir², cd³, 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.

3. 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;

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).

Έμφαση απαιτείται να δοθεί στον όρο **συγκεκριμένη** 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 εμφανές από τα αντίστοιχα παραδείγματα των παραπάνω τύπων δεδομένων.

4. 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.

Αριθμητικοί τελεστές

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

Παράδειγμα: $w = ((x + 3) * (x - y)) / 10$

Λογικοί τελεστές

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

Παράδειγμα: $x > 5$, $!(x > 5)$ ισοδύναμο με $x \leq 5$, $y == x$

5. Εντολές Εξόδου Δεδομένων

Με εφευρέσιο 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 color: %s\n", "blue");
    printf("First number: %d\n", 12345);
    printf("Second number: %04d\n", 25);
    printf("Third number: %i\n", 1234);
    printf("Float number: %3.2f\n", 3.14159);
    printf("Hexadecimal: %x\n", 255);
    printf("Octal: %o\n", 255);
    printf("Unsigned value: %u\n", 150);
}
```

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*/
```

6. 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 );
}
```

όπου παράγεται το εξής μήνυμα στην οθόνη του χρήστη:

```
PROSOXI stis metatropes dedomenwn! Alli timi exei to x:12.324 prin tin metatropi 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 );
}
```

όπου παράγεται το εξής μήνυμα στην οθόνη του χρήστη:

```
PROSOXI stis metatropes dedomenwn! Alli timi exei to x:21 prin tin metatropi kai alli meta opou y:21.0
```

7. 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);
}
```

όπου παράγεται το εξής μήνυμα στην οθόνη του χρήστη:

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

Στη συνέχεια, αντιγράψουμε 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, 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.

```
sh-4.2$ gcc -o main *.c
sh-4.2$ main
1836
1050
sh-4.2$
```

```
#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 */
}
```

Παράδειγμα 2

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.

```
sh-4.2$ gcc -o main *.c
sh-4.2$ main
120
1189
sh-4.2$
```

```
#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);
}
```

Παράδειγμα 3

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.



```
sh-4.2$ gcc -o main *.c
sh-4.2$ main
66528
sh-4.2$
```

```
#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);
}
```

Εργαστήριο 2 - Εντολές ελέγχου και Επανάληψης

1. Εντολές Λογικού Ελέγχου (AN)

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 AN (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. Δομές Επανάληψεων

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

Η γενική σύνταξη της *for* είναι:

```
int i;
for(i=0; i<5; i=i+1){
    //ΕΚΤΕΛΕΣΕ ΤΩΝ ΚΩΔΙΚΑ 6 ΦΟΡΕΣ (ΑΠΟ 0-5)
}
```

Η γενική σύνταξη της *while* είναι:

```
while (λογική συνθήκη) {
    //ΕΚΤΕΛΕΣΕ ΤΩΝ ΚΩΔΙΚΑ ΟΣΟ Η ΣΥΝΘΗΚΗ ΕΙΝΑΙ ΑΛΗΘΗΣ
}
```

Η γενική σύνταξη της *DoWhile* είναι:

```
do {
    // ΕΚΤΕΛΕΣΕ ΤΩΝ ΚΩΔΙΚΑ ΟΣΟ Η ΣΥΝΘΗΚΗ ΕΙΝΑΙ ΑΛΗΘΗΣ
    // ΑΛΛΑ ΤΟΥΛΑΧΙΣΤΟΝ ΜΙΑ ΦΟΡΑ ΓΙΑΤΙ Ο ΕΛΕΓΧΟΣ ΓΙΝΕΤΑΙ ΣΤΟ ΤΕΛΟΣ
} while (λογική συνθήκη) ;
```

Οι τρεις 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 { }
1 ^η	0	ΝΑΙ ($0 \leq 5$)	ΝΑΙ
2 ^η	0+1=1	ΝΑΙ ($1 \leq 5$)	ΝΑΙ
3 ^η	1+1=2	ΝΑΙ ($2 \leq 5$)	ΝΑΙ
4 ^η	2+1=3	ΝΑΙ ($3 \leq 5$)	ΝΑΙ
5 ^η	3+1=4	ΝΑΙ ($4 \leq 5$)	ΝΑΙ
6 ^η	4+1=5	ΝΑΙ ($5 \leq 5$)	ΝΑΙ
7 ^η	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 1^{ης} 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.

```

Default Term + Browser
sh-4.2$ gcc -o main *.c
sh-4.2$ main
-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        s
sh-4.2$

```

```
#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) ;
}
```

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.

```

sh-4.2$ gcc -o main *.c
sh-4.2$ main
Parakalw eisagete mia timi
12
Parakalw eisagete NEA timi
0
Parakalw eisagete NEA timi
-56
Parakalw eisagete NEA timi
10
Parakalw eisagete NEA timi
3
Dekti i timi eisagwgis
Hello Student! Hello Student! Hello Student!
sh-4.2$
    
```

```

#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;
}
    
```

Εργαστήριο 3 - Δομές Επανάληψεων

Σε συνέχεια του προηγούμενου εργαστηρίου, θα εξεταστούν αναλυτικότερα έννοιες που αφορούν τις δομές επανάληψεων. 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)	if (συνθηκη)
printf("X");	{
printf("Y");	printf("X");
	}
	printf("Y");

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

- Το X εκτυπώνεται ΜΟΝΟ όταν ισχύει η συνθήκη.
- Το Y εκτυπώνεται ανεξαρτήτως της συνθήκης.

Ακολουθεί παράδειγμα προγραμματιστικού κώδικα, για μια απλή δομή επανάληψης:

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

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

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

Παράδειγμα 1

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.

```

sh-4.2$ gcc -o main *.c
sh-4.2$ main
Enter number in binary format:21
You must enter exactly 10digits!
sh-4.2$ main
Enter number in binary format:1010101011
The number is 683
sh-4.2$
  
```

```

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

    printf("Eisage 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);
}
  
```

Παράδειγμα 2

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.

```
sh-4.2$ gcc -o main *.c
sh-4.2$ main
Eisagete THETIKI timi mikroteri tou 1000:21
21 Den einai telios arithmos
sh-4.2$ main
Eisagete THETIKI timi mikroteri tou 1000:6
6 Einai telios arithmos
sh-4.2$
```

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

```
#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;
}
```

Εργαστήριο 4 - Συναρτήσεις

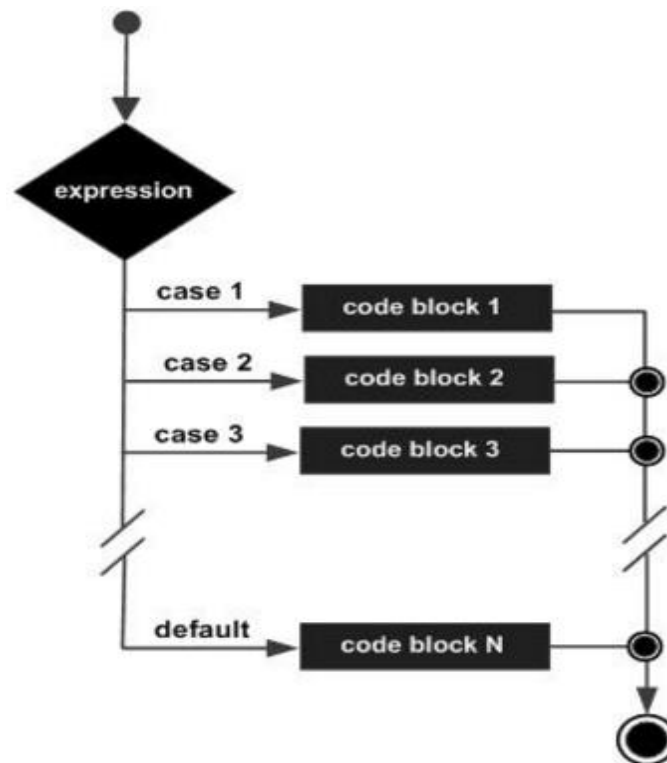
Στόχος του 4^{ου} εργαστηρίου είναι η περαιτέρω εξοικείωση με βασικές έννοιες του προγραμματισμού και ειδικότερα των συναρτήσεων.

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

Ο κώδικας υλοποίησης έχει την εξής μορφή:

```
switch(μεταβλητή_ελέγχου)
{
case: (Τιμή_μεταβλητής_ελέγχου_1) :
    //ΕΚΤΕΛΕΣΕ ΤΟΝ ΚΩΔΙΚΑ
    break;
case: (Τιμή_μεταβλητής_ελέγχου_2) :
    //ΕΚΤΕΛΕΣΕ ΤΟΝ ΚΩΔΙΚΑ
    break;
default:
    //ΕΚΤΕΛΕΣΕ ΤΟΝ ΚΩΔΙΚΑ
    break;
}
```

Το διάγραμμα λειτουργίας της είναι το εξής:



Εικόνα 11 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 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 (μία για κάθε λειτουργία).

Σε περίπτωση εσφαλμένης εισαγωγής στοιχείων:

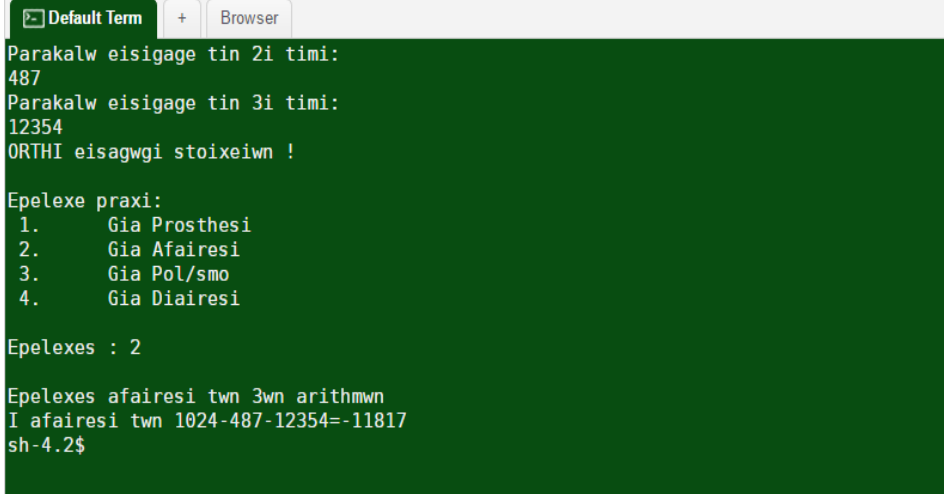
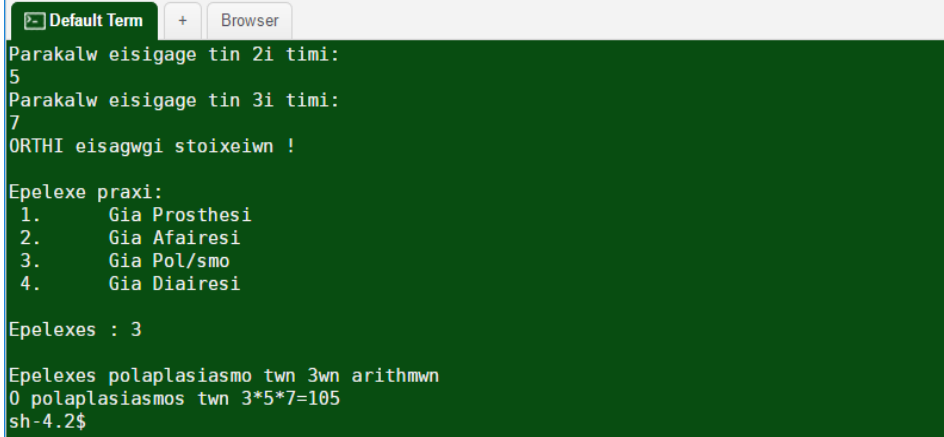
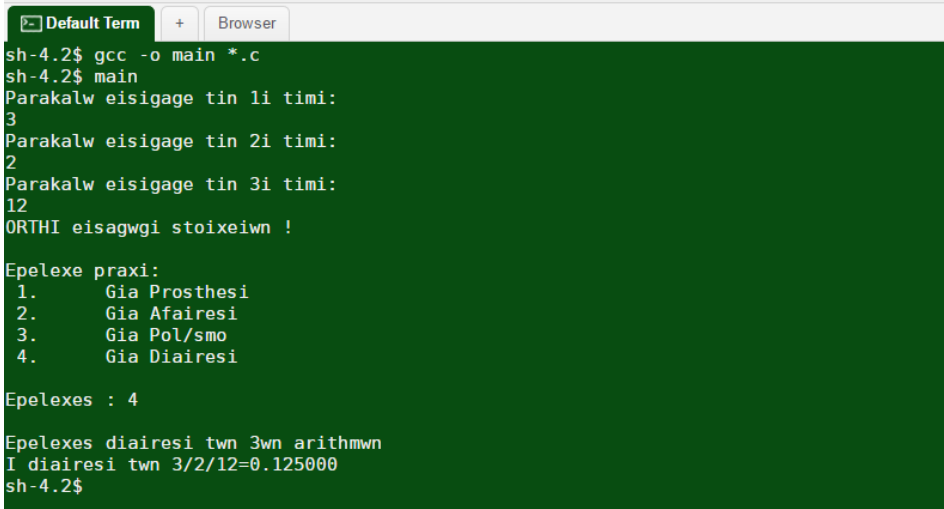
```

Default Term + Browser
Parakalw eisigage tin 2i timi:
0
Parakalw eisigage tin 3i timi:
2
LATHOS eisagwgi stoxeiwn eisigage ek neou 3 times!
Parakalw eisigage tin 1i timi:
1
Parakalw eisigage tin 2i timi:
2
Parakalw eisigage tin 3i timi:
0
LATHOS eisagwgi stoxeiwn eisigage ek neou 3 times!
Parakalw eisigage tin 1i timi:
1
Parakalw eisigage tin 2i timi:
2
Parakalw eisigage tin 3i timi:
3
LATHOS eisagwgi stoxeiwn eisigage ek neou 3 times!
Parakalw eisigage tin 1i timi:

```

Σε περίπτωση ορθής εισαγωγής στοιχείων:

<p>Επιλογής Πρόσθεσης (1)</p>	<pre> sh-4.2\$ gcc -o main *.c sh-4.2\$ main Parakalw eisigage tin 1i timi: 10 Parakalw eisigage tin 2i timi: 20 Parakalw eisigage tin 3i timi: 70 ORTH1 eisagwgi stoxeiwn ! Epelexe praxi: 1. Gia Prosthesi 2. Gia Afairesi 3. Gia Pol/smo 4. Gia Diairesi Epelexes : 1 Epelexes prosthesi twv 3wn arithmwv I prosthesi twv 10+20+70=100 sh-4.2\$ </pre>
--	---

Επιλογής Αφαίρεσης (2)	 <pre> Default Term + Browser Parakalw eisigage tin 2i timi: 487 Parakalw eisigage tin 3i timi: 12354 ORTHI eisagwgi stoixeewn ! Epelexe praxi: 1. Gia Prothesi 2. Gia Afairesi 3. Gia Pol/smo 4. Gia Diairesi Epelexes : 2 Epelexes afairesi tw n 3wn arithmw n I afairesi tw n 1024-487-12354=-11817 sh-4.2\$ </pre>
Επιλογής Πολ/σμου (3)	 <pre> Default Term + Browser Parakalw eisigage tin 2i timi: 5 Parakalw eisigage tin 3i timi: 7 ORTHI eisagwgi stoixeewn ! Epelexe praxi: 1. Gia Prothesi 2. Gia Afairesi 3. Gia Pol/smo 4. Gia Diairesi Epelexes : 3 Epelexes polaplasiasmo tw n 3wn arithmw n 0 polaplasiasmos tw n 3*5*7=105 sh-4.2\$ </pre>
Επιλογής Διαίρεσης (4)	 <pre> Default Term + Browser sh-4.2\$ gcc -o main *.c sh-4.2\$ main Parakalw eisigage tin 1i timi: 3 Parakalw eisigage tin 2i timi: 2 Parakalw eisigage tin 3i timi: 12 ORTHI eisagwgi stoixeewn ! Epelexe praxi: 1. Gia Prothesi 2. Gia Afairesi 3. Gia Pol/smo 4. Gia Diairesi Epelexes : 4 Epelexes diairesi tw n 3wn arithmw n I diairesi tw n 3/2/12=0.125000 sh-4.2\$ </pre>

```
#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 Prothesi\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 tw n xaraktirwn
        scanf("%d",&input4);
        switch(input4){
            case 1:
                printf ("Epelexes prosthesi tw n 3wn
arithmwn\n");
                K=prothesi(input1,input2,input3);
                printf ("I prosthesi tw n
%d+%d+%d=%d\n",input1,input2,input3,K);

                break;
```

```

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

            break;

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

            break;

        case 4:
            printf ("Epelexes diairesi tw n 3wn
arithmw n\n");
            float K1=0; //dilwsi neou K kathws diairesi
paragei FLOAT arithmus
            K1=diairesi(input1,input2,input3);
            printf ("I diairesi tw n
%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);
}

```

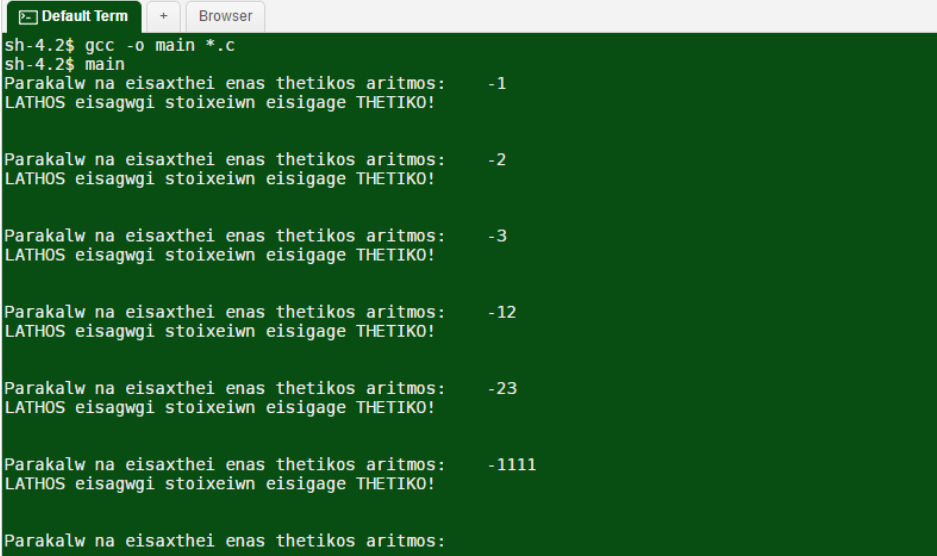
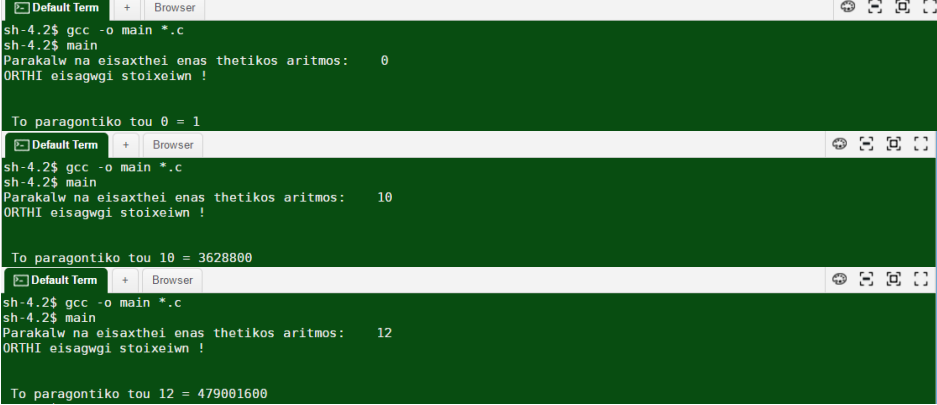
Εργαστήριο 5 - Αναδρομή Συναρτήσεων και Επαναληπτική Λειτουργία

Παράδειγμα 1

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

- Θα δέχεται από τον χρήστη μια ΑΚΕΡΑΙΑ τιμή.
- Σε περίπτωση εισαγωγής αρνητικού αριθμού, θα εμφανίζει σχετικό μήνυμα και θα ειδοποιεί τον χρήστη για την εκ νέου εισαγωγή στοιχείων.
- 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). Επιπλέον, το πρόγραμμα απαιτείται να μεταγλωττίζεται και να εκτελείται επιτυχώς.

<p>Λάθος Εισαγωγής Στοιχείων εισόδου</p>	
<p>Ορθή Εισαγωγή Στοιχείων εισόδου</p>	

Διαδικασία Υπολογισμού Παραγοντικού:

Παραδείγματα υπολογισμού:

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

```
#include <stdio.h>
long int paragontiko(int arithmos)
{
    if (arithmos>0){
        return (arithmos*paragontiko(arithmos-1) );
    }
    else {
        return 1;
    }
}

main()
{
    int inputXristi,flag=0;

    while(flag==0){

        printf("Parakalw na eisaxthei enas "
            "thetikos aritmos: \t");

        scanf("%d", &inputXristi);

        if (inputXristi<0){
            printf("LATHOS eisagwgi stoxeiwn eisigage THETIKO! \n\n\n");
        }
        else
```

```

    {
        printf("ORTH1 eisagwgi stoxeiwn !\n\n");
        flag=1;
    }

    printf("\n To paragontiko tou %d = %ld \n",inputXristi,paragontiko(inputXristi));
}

```

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

Ένα απλό πρόγραμμα για τα μεγέθη:

```

#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;
}

```

Εργαστήριο 6 - Δείκτες

Παράδειγμα 1

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 .

```

sh-4.2$ gcc -o main *.c
sh-4.2$ main
Eisigage mia thetiki timi
21
I timi tu xristi einai: 21
I timi tu xristi einai: 21
I eisodos tu xristi exei apothikeutei sti dieuthinsi mnimis:0x7ffe3a060f8c kai exei timi: 21

=====
Stoixeio      Timi      Keli Mnimis
Stoixeio[0]   -4      0x7ffe3a060f60
Stoixeio[1]   -3      0x7ffe3a060f64
Stoixeio[2]   -2      0x7ffe3a060f68
Stoixeio[3]   -1      0x7ffe3a060f6c
Stoixeio[4]    0      0x7ffe3a060f70
Stoixeio[5]    1      0x7ffe3a060f74
Stoixeio[6]    2      0x7ffe3a060f78
Stoixeio[7]    3      0x7ffe3a060f7c
Stoixeio[8]    4      0x7ffe3a060f80

=====

0 pointer tu pinaka mas: 0x7ffe3a060f60
I timi tu lou stioxeiou tu pinaka: -4

```

```

#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);

```

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

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

//2ος 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 stoixeiou tu pinaka: %d \n",*Array);
//printf("To 3o stoixeio tou pinaka einai:%d \n",*(Array+2));
}
```

Παράδειγμα 2

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.


```

Default Term + Browser
sh-4.2$ gcc -o main *.c
sh-4.2$ main
Table A --πίνακας 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 --πίνακας 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

```

```

#include <stdio.h>

int main()
{
    int x;
    char w,k;
    /* δήλωση των δύο πινάκων χαρακτήρων 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 --πίνακας 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 --πίνακας B-- ( indicators): \n");
    x=0;
    while(*(B+x)!='\0')
    {
        printf("%c ", *(B+x));
        x++;
    }
    printf("\n\n");
    return 0;
}

```

Εργαστήριο 7 & 8- Αρχεία

Παράδειγμα 1

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 hi1.dat, 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.

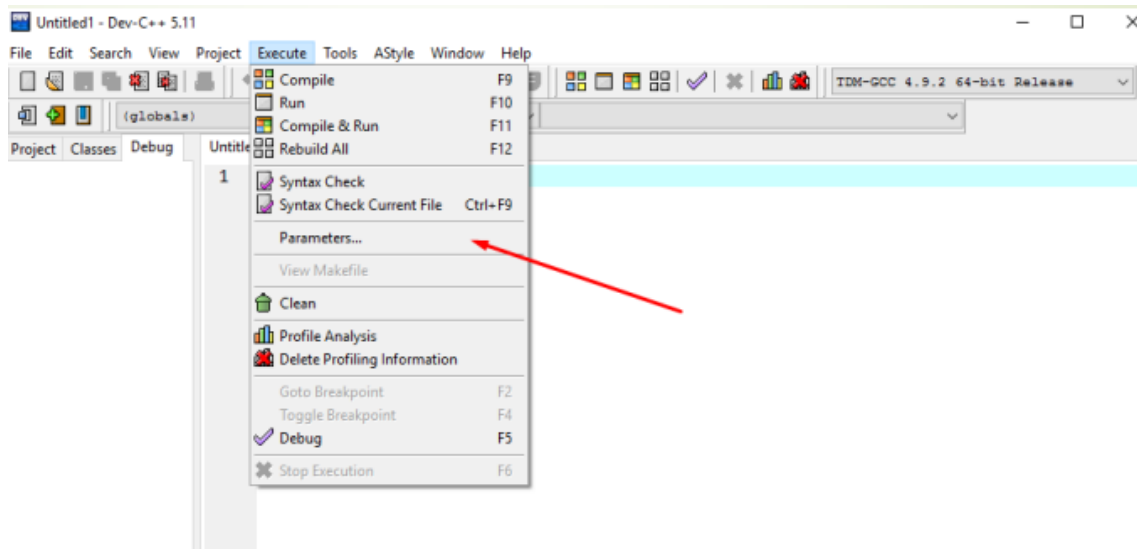
```
#include <stdio.h>
#include <locale.h>
#include <stdlib.h>
#include <string.h>
int main(int argc, char **argv)
{
    system("chcp 1253");
    /* Δηλώση μεταβλητών */
    float r1,r2;
    FILE *fp; /* δεικτης αρχείου*/
    char *temp = ":\hi1.dat"; // αρχική τιμή ονόματος αρχείου
    if(argc !=2)
    {
        printf("Η χρήση της εντολής είναι : %s Γραμμα_περιφερειακής_μονάδας (π.χ. D , E)\n",
        argv[0]);
        exit(1);
    }
    /* Δημιουργία 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
    /* Αναγνώση δεδομένων μέσω αλληλεπίδρασης με το χρήστη*/
    do
    {
        printf("Δώστε έναν θετικό πραγματικό αριθμό\n");
        fflush(stdin);
    }while(0!=(scanf("%f",&r1)) && r1<=0);

    do
    {
```

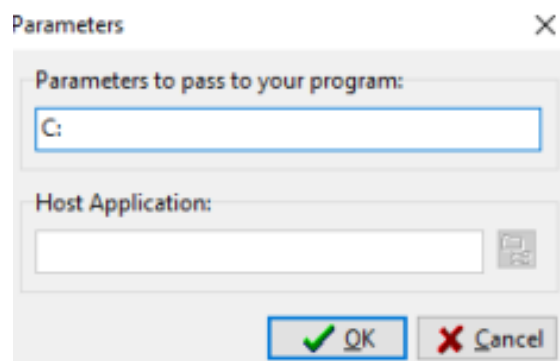
```
printf("Δώστε έναν αρνητικό πραγματικό αριθμό\n");
fflush(stdin);
}while(0!=(scanf("%f",&r2)) && r2>=0);

/* Ανοιγμα, εγγραφή στο αρχείο & κλείσιμο αρχείο με την εκτύπωση μηνύματος */
fp=fopen(file, "w");
fprintf(fp, "%f %f \n", r1, r2);
fclose(fp);
printf("\nΕδώσες 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:

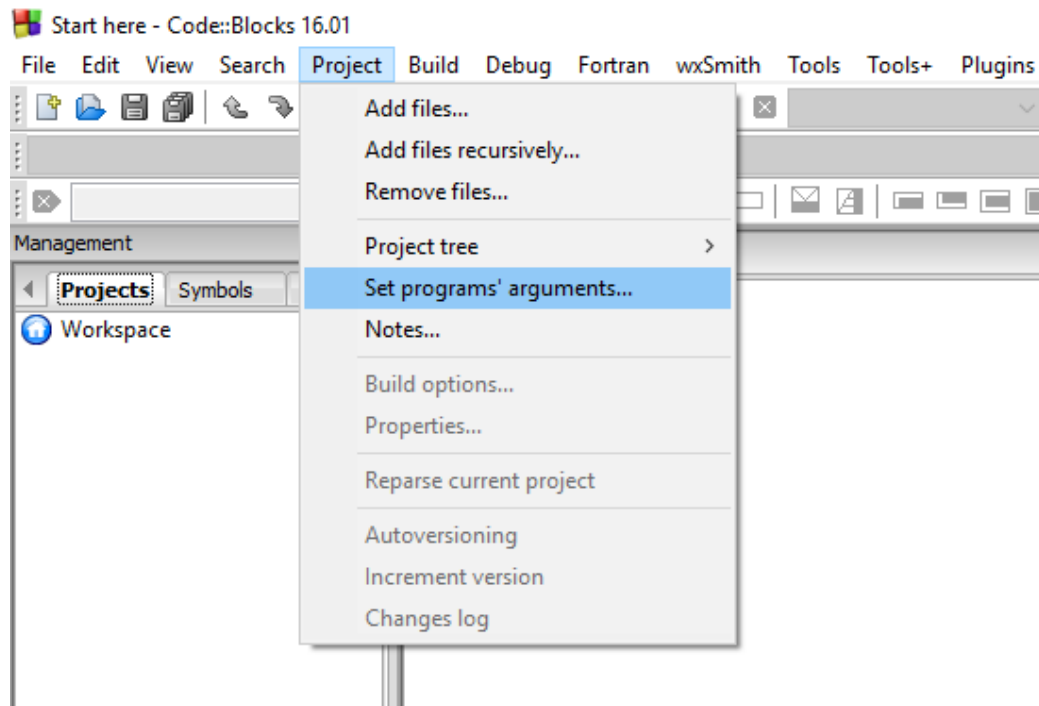


Εικόνα 12 Βήμα 1 επιλέγω Parameters

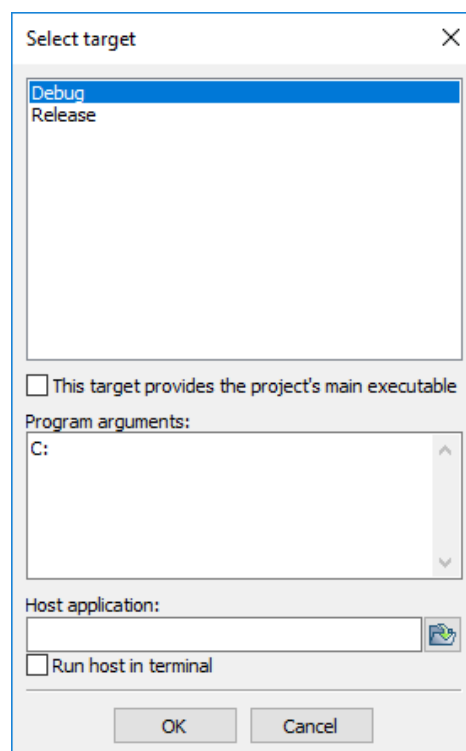


Εικόνα 13 Βήμα 2 ορίζω την παράμετρο. Προσοχή στο γράμμα/διεύθυνση που θα ορίσετε στο πρόγραμμά σας

Η αντίστοιχη διαδικασία και για το Codeblocks:



Εικόνα 14 Βήμα 1 επιλέγω arguments, για το πρόγραμμα που χρησιμοποιώ



Εικόνα 15 Βήμα 2 ορίζω την αποθηκευτική μονάδα. Προσοχή στο γράμμα/διεύθυνση που θα ορίσετε στο πρόγραμμά σας

Παράδειγμα 2

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 δίνει μόνο το
    ακέραιο υπόλοιπο για να βρούμε το πραγματικό και
    h=(f*t)-(r*d); //το φανταστικό μέρος κάνουμε τις πράξεις σπαστά
    και τις αποθηκεύουμε στις float μεταβλητές 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");
    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("\n0 πρώτος αριθμός είναι:
%f+%fi\n",info1.re,info1.im);
        else
            printf("\n0 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("0 δεύτερος αριθμός είναι:
%f+%fi\n\n",info2.re,info2.im);
        else
            printf("0 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');
    }
}

```

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 - Ειδικές εντολές-Αναδρομή-Επανάληψη

Παράδειγμα 1

Να κατασκευάσετε ένα πρόγραμμα, το οποίο:

- Θα επιτελεί, μέσω μιας αναδρομικής συνάρτησης, τον υπολογισμό της δύναμης ενός αριθμού (π.χ. $2^0 = 1$, $2^3 = 8$, $(-3)^2 = 9$, ...)
- 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.
- Το πρόγραμμα απαιτείται να μεταγλωττίζεται και να εκτελείται επιτυχώς.

```
#include <stdio.h>
#include <math.h>
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);
}
main ()
{
    int x,n,apotelesma;
    printf("Eisigage arithmo: \n");
    scanf("%d",&x);
    printf("Eisigage ti dynami pou thes na ypologiseis: \n");
    scanf("%d",&n);
    apotelesma=VresDinami(x,n);
    printf("Apotelesma: %d \n",apotelesma);
}
```


Παράδειγμα 2

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();
        printf("\n");
    }while(ch!='n' && ch!='N');
}
```

Βιβλιογραφία

- 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
- Αλέξανδρος Σ. Καράκος, Οδηγός Προγραμματισμού με τη γλώσσα C, Εκδόσεις: Καράκος, 2017
- Σπύρος Γερούλης, Εισαγωγή στη C, Εκδόσεις: Spin, 2006
- Kernighan Brian W., Ritchie Dennis M., Η Γλώσσα Προγραμματισμού C, Prentice Hall, Εκδόσεις: Κλειδάριθμος, 2011
- Shaw Zed A., Learn C the Hard Way, Εκδόσεις: Zed Shaw's Hard Way Series, 2015
- Webber Adam Brooks, Σύγχρονες Γλώσσες Προγραμματισμού, μετάφραση Γεωργακόπουλος Γεώργιος, Παπαδόγγονας Φρ. Ιωάννης, Εκδόσεις: Πανεπιστημιακές Εκδόσεις Κρήτης, 2009

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/orwelldvcpp>
- 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>