ΕΡΓΑΣΙΑ 1

ΕΙΣΑΓΩΓΗ – ΒΑΣΙΚΑ ΣΤΟΙΧΕΙΑ

Χρήστος Μαργιώλης – Τμήμα 9

Τί είναι πρόγραμμα;

Πρόγραμμα είναι ένα σύνολο εντολών – συγκεκριμένου αριθμού – οι οποίες οδηγούν στην υλοποίση μιας λειτουργίας.

Ποιά είναι τα βασικά βήματα ανάπτυξης και εκτέλεσης ενός C προγράμματος;

1. Σύνταξη (editing): Δημιουργία του πηγαίου κώδικα *(source code)* σε έναν συντάκτη *(editor)*, όπως για παράδειγμα Visual Studio Code, Sublime Text, Notepad++ κ.α. Ο τύπος αρχείου του πηγαίου κώδικα C είναι `.c`.

2. Μεταγλώττιση (compilation): Ο μεταγλωττιστής *(compiler)* ελέγχει τον κώδικα για τυχόν συντακτικά λάθη και στην συνέχεια, σε περίπτωση που δεν βρεθεί κάποιο, δημιουργεί τον object code, ο οποίος είναι σχεδόν σε γλώσσα μηχανής. Στην C η μεταγλώττιση ενός προγράμματος συνήθως γίνεται με τον μεταγλωττιστή `gcc` χρησιμοποιώντας την εντολή `gcc -o όνομα-αρχείου.c`. Το αρχείο που παράγεται συνήθως έχει την μορφή `όνομα-αρχείου.obj` ή `όνομα-αρχείου.o`.

3. Ένωση (linking): Ο linker βρίσκει τις βιβλιοθήκες συναρτήσεων που έχουν χρησιμοποιηθεί στο πρόγραμμα και συνδυάζει τον source με τον object code και έτσι παράγει ένα εκτελέσιμο αρχείο μορφής π.χ `.exe` ή `a.out` κ.α.

3. Εκτέλεση (execution): Εκτέλεση του τελικού προγράμματος.

Ποιές είναι οι γενικές κατηγορίες λαθών των προγραμμάτων;

1. Συντακτικά (syntax errors): Λάθη που παραβιάζουν το συντακτικό μιας γλώσσας προγραμματισμού – για παράδειγμα `printf(number;` αντί για `printf(“number”);`

2. Λογικά (logic errors): Λάθη που προκαλούνται από λανθασμένη σκέψη του προγραμματιστή όσον αφορά την λειτουργία του προγράμματος.

3. Λάθη κατά την εκτέλεση (runtime errors): Λάθη που προκαλούνται κατά την διάρκεια της εκτέλεσης του προγράμματος και δεν οφείλονται ούτε σε λάθη λογικής ούτε σύνταξης. *Για παράδειγμα*, ένα τέτοιο λάθος είναι η απώλεια ενός αρχείου από τον υπολογιστή, το οποίο είναι απαραίτητο για την λειτουργία του προγράμματος.

Τί είναι μεταβλητή; Ποιά τα βασικά χαρακτηριστικά μίας μεταβλητής;

Μεταβλητή μια θέση μνήμης στην οποία δίνουμε ένα συμβολικό όνομα, και στην οποία αποθηκεύεται μια τιμή συγκεκριμένου τύπου δεδομένων. Η τιμή αυτή καθορίζεται και μπορεί να μεταβληθεί κατά την διάρκεια της σύνταξης ή της εκτέλεσης ενός προγράμματος.

Τα *βασικά* χαρακτηριστικά μίας μεταβλητής είναι τα εξής:
1. Δεν αλλάζει το όνομα της.
2. Δεν αλλάζει ο τύπος της.
3. Δεν μπορεί το όνομα της μεταβλητής να ξεκινάει με αριθμούς ή να περιέχει κενά και ειδικούς χαρακτήρες, όπως παρενθέσεις κλπ.

Ερώτημα 4

Τί είναι η standard είσοδος και τί η standard έξοδος ενός προγράμματος;

- **Standard είσοδος *(standard input ή stdin)*:** Ροή εισόδου (input stream) στην οποία εισάγονται δεδομένα και διαβάζονται από το πρόγραμμα.
- **Standard έξοδος *(standard output ή stdout)*:** Ροή εξόδου (output stream) στην οποία το πρόγραμμα παίρνει δεδομένα του και τα εμφανίζει στην οθόνη.

Ερώτημα 5 (πρώτο μέρος)

Τί γνωρίζετε για τις συναρτήσεις με τις οποίες χειριζόμαστε την standard είσοδο και έξοδο στον C προγραμματισμό;

- **`scanf()`**: Είναι συνάρτηση εισόδου η οποία παρέχεται από την βιβλιοθήκη stdio.h. Μέσα στις παρενθέσεις εισάγονται δεδομένα και *συνήθως* καταχωρούνται μέσα σε κάποια μεταβλητή – για παράδειγμα η `scanf("%d", &num)` θα λάβει από τον χρήστη έναν ακέραιο αριθμό και θα τον καταχωρήσει στην μεταβλητή `num`.

- **`printf()`**: Είναι συνάρτηση εξόδου η οποία *επίσης* παρέχεται από την βιβλιοθήκη stdio.h. Μέσα στις παρενθέσεις ο προγραμματιστής δηλώνει τι θέλει να εμφανίσει η συνάρτηση στην οθόνη – για παραδείγμα, αν πάρουμε δεδομένο οτι η τιμή της μεταβλητής `num` είναι 50, η `printf("The number is %d", num)` θα εμφανίσει στην οθόνη *The number is 50*.

Και οι δύο συναρτήσεις έχουν μια *σχετικά* κοινή δομή στην σύνταξη τους. Αρχικά γράφουμε `" "` για να δηλώσουμε τα δεδομένα που θα εισάγουμε στην `scanf()` και θα εξάγουμε αντίστοιχα με την `printf()`. Αυτά τα δεδομένα μπορούνε να είναι είτε κείμενο/χαρακτήρες, είτε αριθμοί. Στην συνέχεια, *αν θελήσουμε* να καταχωρηθούν δεδομένα σε κάποια μεταβλητή, ή να εμφανιστούν τα δεδομένα κάποιας μεταβλητής, πρέπει μετά τα `" "` να αναφέρουμε και τις μεταβλητές που θα χρειαστούν οι συναρτήσεις, όπως στο παραπάνω παράδειγμα. Προκειμένου όμως η συνάρτηση να "καταλάβει" οτι θέλουμε να χρησιμοποιηθεί κάποια μεταβλητή, πρέπει μέσα στα `" "` να δηλώσουμε με το σύμβολο `%` *(format identifier)* και το ανάλογο γράμμα *(`d`, `f`, `lf`, `s`, `c` κ.α)* τον τύπο της μεταβλητής που θέλουμε να εμφανιστεί/καταχωρηθεί.

Στα παραπάνω παραδείγματα η μεταβλητή `num` είναι τύπου `int` (ακέραιος αριθμός), και για αυτό δηλώνουμε με το `%d` – όπου `d` σημαίνει *decimal integer* – οτι η μεταβλητή που θα χρειαστούμε σε εκείνη την θέση θα είναι ακεραίου τύπου.

Υπάρχουν και άλλες συναρτήσεις με τις οποίες χειριζόμαστε την standard είσοδο/έξοδο στην C – μερικές από αυτές είναι η `fgets()` και η `fputs()`, οι οποίες αφορούν τις συμβολοσειρές.

Τί είναι οι αριθμητικοί, τί οι σχεσιακοί και τί είναι οι λογικοί τελεστές; Αναφέρετε την λειτουργία των βασικότερων τελεστών και από τις τρεις κατηγορίες.

- **Αριθμητικοί τελεστές *(arithmetic operators)*:** Τελεστές οι οποίοι εκτελούν συγκεκριμένες αριθμητικές πράξεις μεταξύ αριθμών ή μεταβλητών.

Τελεστής	**Πράξη**	**Παράδειγμα**
+	Πρόσθεση	5 + 4
-	Αφαίρεση	5 - 4
*	Πολλαπλασιαμός	5 * 4
/	Διαίρεση	5 / 4
%	Υπόλοιπο διαίρεσης (mod)	5 % 4

Πηγή: Wikiversity

- **Σχεσιακοί τελεστές *(relational operators)*:** Τελεστές οι οποίοι συγκρίνουν αριθμούς ή μεταβλητές.

Τελεστής	**Πράξη**	**Παράδειγμα**
==	Ίσο	(x == y)
!=	Διάφορο	(x =! y)
>	Μεγαλύτερο	(x > y)
>=	Μεγαλύτερο ή ίσο	(x >= y)
<	Μικρότερο	(x < y)
<=	Μικρότερο ή ίσο	(x <= y)

Πηγή: Wikiversity

- **Λογικοί τελεστές *(logical operators)*:** Τελεστές οι οποίοι συνθήκες και επιστρέφουν τιμές 1 ή 0, οι οποίες αντιστιχούν σε `true` ή `false` αντίστοιχα.

Τελεστής	**Πράξη**	**Παράδειγμα**
&&	Σύζευξη (AND)	(1 && 0)
||	Διάζευξη (OR)	(1 || 0)
!	Άρνηση (NOT)	(!1)

Πηγή: Wikiversity

Ποιά η διαφορά του τελεστή προσαύξησης από τον τελεστή μετααύξησης;

Ας πάρουμε ως παράδειγμα το παρακάτω πρόγραμμα

```
#include <stdio.h>
int main()
{
  int var=1;

  //Πρώτα αυξάνεται η μεταβλητή σε 2, και μετά
εμφανίζεται
  printf("%d",++var);


   //Πρώτα εμφανίζεται το 1, και στην συνέχεια αυξάνεται
και γίνεται 2
  printf("%d\n",var++);

  return 0;
}
```

- **Τελεστής προσάυξησης (++var):** Πρώτα αυξάνεται *κατα 1* η τιμή της μεταβλητής και μετά εμφανίζεται.

- **Τελεστής μετααύξησης (var++):** Πρώτα εμφανίζεται με την *τρέχουσα τιμή* της η μεταβλητή και έπειτα αυξάνεται *κατά 1*.

C1a.c – Αρχικός κώδικας με λάθη

```c
#include <stdio.h>

int main(int argc, int **argv)
{
    int A, B;
    int C, D, E, F;
    system ("chcp 1253");
    printf ("Βασικές αριθμητικές πράξεις με Ακεραίους\n");
    printf ("=======================================\n\n");
    printf ("Εισάγετε τον πρώτο αριθμό  : ");
    scanf ("%d",&A);
    printf ("Εισαγετε το δεύτερο αριθμό : );
    scanf ("%d", &B);
    C = A + B;
    D = A - B;
    E = A * B;
    F = A / B;
    printf ("Άθροισμα  : %d\n", c);
    printf ("Διαφορά   : %d\n", D);
    printf ("Γινόμενο  : %d\n", E);
    prantf ("Πηλίκο    : %d\n", F);
    return 0;;
}
```

Λάθη

Γραμμή 3: Καλύτερο είναι το `**argv` να δηλωθεί ως `char` και όχι ως `int`.
Γραμμή 12: Λείπουν τα εισαγωγικά κλεισίματος της συμβολοσειράς.
Γραμμή 18: Η μεταβλητή c πρέπει να γραφτεί με κεφαλαίο C επειδή η C είναι case sensitive γλώσσα και λαμβάνει το `c` ως διαφορετικό από το C.
Γραμμή 20: Έχει χρησιμοποιηθεί ελληνικό Ε αντί για λατινικό και ο compiler δεν το αναγνωρίζει.
Γραμμή 21: Η συνάρτηση πρέπει να είναι `printf()`, όχι `prantf()`.

C1a-CORRECTED.c – Διορθωμένος κώδικας

```c
#include <stdio.h>

int main(int argc, char **argv)
{
    int A, B;
    int C, D, E, F;
    system ("chcp 1253");
    printf ("Βασικές αριθμητικές πράξεις με Ακεραίους\n");
    printf ("=======================================\n\n");
    printf ("Εισάγετε τον πρώτο αριθμό  : ");
    scanf ("%d",&A);
    printf ("Εισαγετε το δεύτερο αριθμό : ");
    scanf ("%d", &B);
    C = A + B;
    D = A - B;
    E = A * B;
    F = A / B;
    printf ("Άθροισμα  : %d\n", C);
    printf ("Διαφορά   : %d\n", D);
    printf ("Γινόμενο  : %d\n", E);
    printf ("Πηλίκο    : %d\n", F);
    return 0;
}
```

Παρατηρήσεις κατά την εκτέλεση

Ακέραιος 1	Ακέραιος 2	Άθροισμα	Διαφορά	Γινόμενο	Πηλίκο
10	3	13	7	30	3
370	15	385	355	5550	24
10000000000	1000000	1411065408	1409065408	1874919424	1410

Παρατήρηση 1: Εφόσον ο τύπος δεδομένων των μεταβλητών του προγράμματος είναι `int`, η διαίρεση θα δίνει πάντα το ακέραιο πηλίκο, ασχέτως του αριθμού που θα δώσουμε, όπως για παράδειγμα στις γραμμές 1 και 2.

Παρατήρηση 2: Όπως φαίνεται από την γραμμή 3, όταν δωθούν πολύ μεγάλοι ακέραιοι, οι οποίοι είτε ξεπερνάνε τα όρια του τύπου δεδομένων `int` (από -2,147,483,648 έως 2,147,483,647), είτε βγάζουν αποτελέσματα μεγαλύτερα από αυτά των ορίων, εμφανίζεται το φαινόμενο της υπερχείλισης *(overflow)*.

IO-Exercise.c – Άσκηση 7

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

int main(int argc, char **argv)
{
    int int_1, int_2, sum, diff, prod, div, sqr;
    double f_div, sqr_root;

    printf("Ακέραιος 1: ");
    scanf("%d", &int_1);
    printf("Ακέραιος 2: ");
    scanf("%d", &int_2);

    sum = int_1 + int_2;
    diff = int_1 - int_2;
    prod = int_1 * int_2;

    sqr = pow(int_1, 2);
    sqr_root = sqrt(int_2);

    printf("Άθροισμα: %d\n", sum);
    printf("Διαφορά: %d\n", diff);
    printf("Γινόμενο: %d\n", prod);
    printf("Τετράγωνο: %d\n", sqr);
    printf("Τετραγωνική ρίζα: %.2lf\n", sqr_root);
    if (int_2 != 0)
    {
        div = int_1 / int_2;
        f_div = (double)int_1 / int_2;
        printf("Πηλίκο: %d\n", div);
        printf("Πραγματικό πηλίκο: %.2lf\n", f_div);
    }
    else
    {
        printf("Δεν γίνεται διαίρεση με το 0.\n");
    }

    return 0;
}
```

Ερώτημα 7 (πρώτο μέρος)

Περιγραφή υλοποίησης άσκησης 7

- **Μεταβλητές:**

  `int_1` και `int_2`: Ακέραιοι αριθμοί που θα χρησιμοποιηθούν για τις πράξεις
  Οι παρακάτω μεταβλητές χρησιμοποιούνται για την καταχώρηση των εξής πράξεων
  `sum`: Άθροισμα
  `diff`: Διαφορά
  `prod`: Γινόμενο
  `div`: Ακέραιο πηλίκο
  `sqr`: Τετράγωνο
  `f_div`: Πραγματικό πηλίκο
  `sqr_root`: Τετραγωνική ρίζα

- **Λειτουργία του προγράμματος:**

  Αρχικά δίνονται οι 2 ακέραιοι, και έπειτα καταχωρούνται οι ζητούμενες πράξεις σε κατάλληλες μεταβλητές για λόγους πρακτικότητας (για να μην χρειαστεί να γίνουν οι πράξεις μέσα στις `printf()`). Για να βρεθεί το τετράγωνο και η τετραγωνική ρίζα χρησιμοποίησα την βιβλιοθήκη `math.h` η οποία παρέχει έτοιμες συναρτήσεις για αυτές τις πράξεις. Στην συνέχεια εμφανίζονται τα αποτελέσματα των πράξεων και γίνεται και ένας έλεγχος για την περίπτωση διαίρεσης με το 0, *αν και με βάση την εκφώνηση δεν χρειαζόταν,* αλλά την έκανα για καλύτερη λειτουργία του προγράμματος. Τέλος, για να βρεθεί το πραγματικό πηλίκο χρειάστηκε η μετατρόπη από `int` σε `double`, εφόσον σε αυτή την περίπτωση είναι *απαραίτητο* να υπάρξουν και δεκαδικά ψηφία.

- **Επιπλέον:**

  Χρησιμοποίησα το `%.2lf` ωστέ να εμφανιστούν μόνο τα πρώτα 2 δεκαδικά ψηφία από τις πράξεις.

  Ο λόγος που χρησιμοποίησα `double` και όχι `float` είναι για να υπάρξει μεγαλύτερη ακρίβεια στα αποτελέσματα *σε περίπτωση που δωθούν πολύ μεγάλοι δεκαδικοί αριθμοί*, ασχέτως αν επέλεξα να εμφανιστούν μόνο τα 2 πρώτα δεκαδικά ψηφία.

Cube-Sphere.c – Άσκηση 8

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

int main(int argc, char **argv)
{
    double length, rad, area_cube, vol_cube, area_sphere,
vol_sphere;
    const double PI = 3.14;

    printf("Μήκος (σε μέτρα): ");
    scanf("%lf", &length);

    area_cube = 6.0*pow(length, 2);
    vol_cube = pow(length, 3);
    rad = length;
    area_sphere = 4.0*PI*pow(rad, 2);
    vol_sphere = (4.0/3.0)*PI*pow(rad, 3);

    printf("Εμβαδόν κύβου: %.2lf\n", area_cube);
    printf("Όγκος Κύβου: %.2lf\n\n", vol_cube);
    printf("Εμβαδόν σφαίρας: %.2lf\n", area_sphere);
    printf("Όγκος σφαίρας: %.2lf\n", vol_sphere);

    return 0;
}
```

Περιγραφή υλοποίησης άσκησης 8

- **Μεταβλητές:**

  `length`: Μήκος/Ακμή κύβου
  `rad`: Ακτίνα σφαίρας
  `area_cube`: Εμβαδόν κύβου
  `vol_cube`: Όγκος κύβου
  `area_sphere`: Εμβαδόν σφαίρας
  `vol_sphere`: Όγκος σφαίρας

- **Σταθερές:**

  `PI`: π

- **Λειτουργία του προγράμματος:**

  Αρχικά δήλωσα το *π* ως σταθερά εφόσον η τιμή του δεν πρόκειται να αλλάξει και για να αποφευχθεί κάποιο λάθος σε περίπτωση που χρειαζόταν να γραφτεί μηχανικά κάθε φορά ως αριθμός. Οι μεταβλητές όλες είναι τύπου `float` γιατί είναι πολύ πιθανό να παίρνουν πραγματικές τιμές. Στην συνέχεια το πρόγραμμα διαβάζει το μήκος/ακμή κύβου και την καταχωρεί στην κατάλληλη μεταβλητή και γίνονται όλες οι πράξεις που ζητούνται από την άσκηση. Και σε αυτή την άσκηση χρησιμοποίησα την βιβλιοθήκη `math.h` για να γίνουν οι πράξεις που περιέχουν δυνάμεις. Τέλος εμφανίζονται όλα τα αποτελέσματα μόνο με τα 2 πρώτα δεκαδικά τους ψηφία.

- **Επιπλέον:**

  Όπως και στην προηγούμενη άσκηση

  Χρησιμοποίησα το `%.2lf` ωστέ να εμφανιστούν μόνο τα πρώτα 2 δεκαδικά ψηφία από τις πράξεις.

  Ο λόγος που χρησιμοποίησα `double` και όχι `float` είναι για να υπάρξει μεγαλύτερη ακρίβεια στα αποτελέσματα *σε περίπτωση που δωθούν πολύ μεγάλοι δεκαδικοί αριθμοί*, ασχέτως αν επέλεξα να εμφανιστούν μόνο τα 2 πρώτα δεκαδικά ψηφία.

Πηγές

- C File input/output – DevDocs
- Chapter 16: The Standard I/O (stdio) Library
- 16.4: Line Input and Output (fgets, fputs, etc.)
- 16.5: Formatted Output (printf and friends)
- 16.6: Formatted Input (scanf)
- C – printf and scanf
- C library function – printf()
- C library function – scanf()
- C library function – fgets()
- Δομημένος Προγραμματισμός Ενότητα 2: Τύποι δεδομένων, Μεταβλητές, Τελεστές και παραστάσεις
- Είσοδος-Έξοδος (Input/Output) – καθιερωμένη είσοδος/έξοδος
- Τμήμα:C/Μάθημα 3ο
- Increment ++ and Decrement -- Operator as Prefix and Postfix
- Stdin
- What is "standard input"?
- C file input/output
- stdin, stdout, stderr
- Confused about stdin, stdout and stderr?
- Standard streams

Για την δημιουργία τις εργασίας χρησιμοποίηθηκαν

- **Editors:** Visual Studio Code, CLion, Notepadqq
- **Compiler:** gcc
- **Συγγραφή:** LibreOffice Writer
- **Γραμματοσειρές:** Liberation Sans *για τα κείμενα* και Liberation Mono *για τους κώδικες*
- **Λειτουργικό σύστημα:** Linux Mint Cinnamon 19.2