Η ΓΛΩΣΣΑ C

Μάθημα 17:

Είσοδος/Έξοδος: Επικοινωνία με το Λειτουργικό Σύστημα

Δημήτρης Ψούνης

www.psounis.gr

Περιεχόμενα Μαθήματος

A. Επικοινωνία με το Λειτουργικό Σύστημα
1. Γενικά

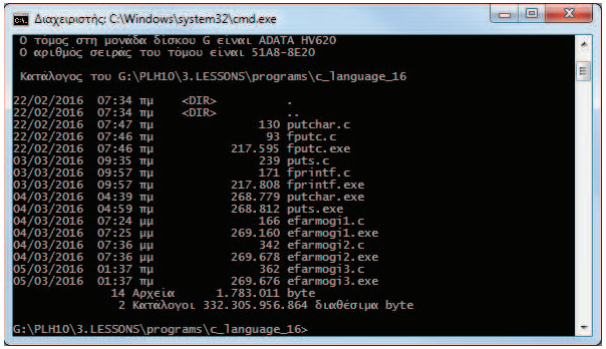
- Στο μάθημα αυτό το πρόγραμμα μας θα αλληλεπιδράσει με το λειτουργικό σύστημα.
  - Θα δούμε πως το πρόγραμμά μας μπορεί να λάβει ορίσματα από την γραμμή εντολής (DOS)
  - Επίσης θα δούμε πως μπορούμε να ανακατευθύνουμε την είσοδο και την έξοδο από τις συνήθεις ροές σε αρχεία!

A. Επικοινωνία με το Λειτουργικό Σύστημα
2. Λειτουργικό Σύστημα

- Το λειτουργικό σύστημα είναι το «πρόγραμμα» που τρέχει άλλα προγράμματα:
  - Το πιο συνηθισμένο είναι τα “Windows” στις διάφορες εκδόσεις του, της εταιρίας Microsoft
    - Υπάρχουν πολλά ακόμη λειτουργικά συστήματα όπως είναι το MAC OS, το UNIX, το LINUX στις διάφορες εκδόσεις τους.
  - Ο «μπαμπάς» των Windows είναι το DOS, στο οποίο (τα αρχαία χρόνια) γραφόντουσαν απευθείας εντολές στο λειτουργικό σύστημα όπως για παράδειγμα:
    - Η εντολή “dir” που εμφανίζει τα περιεχόμενα του τρέχοντος καταλόγου.
    - Η εντολή “cls” που διαγράφει τα περιεχόμενα της οθόνης.
    - Η εντολή “cd” που αλλάζει τον τρέχοντα κατάλογο
    - Και γενικά να γράψουμε το όνομα ενός προγράμματος και να το τρέξουμε.
  - Στα Windows δίνεται η δυνατότητα να τρέξουμε εντολές μέσω της κονσόλας (όπως κάποτε ήταν το DOS) και να τρέξουμε τα προγράμματά μας.

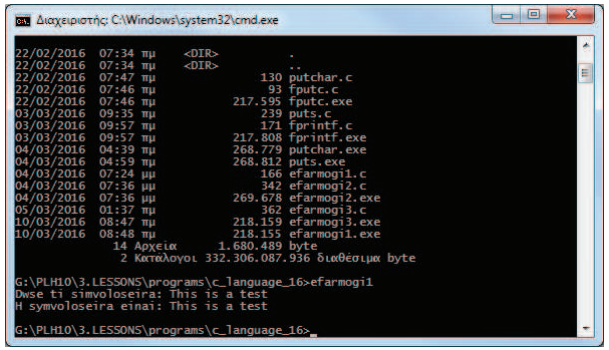
Α. Επικοινωνία με το Λειτουργικό
2. Λειτουργικό Σύστημα

- Για παράδειγμα, πηγαίνοντας «Έναρξη=>Γραμμή Εντολών» (start=>command prompt) μπορούμε να τρέξουμε τις εντολές DOS.
- Στην παρακάτω οθόνη φαίνονται τα περιεχόμενα του καταλόγου του προηγούμενου μαθήματος:



Α. Επικοινωνία με το Λειτουργικό
2. Λειτουργικό Σύστημα

- Εδώ μπορούμε να εκτελέσουμε ένα από τα προγράμματα γράφοντας απλά το όνομα του:



Α. Επικοινωνία με το Λειτουργικό
3. Ορίσματα Γραμμής Εντολής
1. Ορισμός της main με ορίσματα

- Υπάρχει η δυνατότητα να λαμβάνουμε τιμές από την κλήση του προγράμματος από την γραμμή εντολής ως εξής:
  - Ορίζουμε ορίσματα γραμμής εντολής τροποποιώντας το πρωτότυπο της main:

```
main(int argc, char *argv[])
{
...
}
```

  - Όπου κάνοντας κατάλληλη κλήση του προγράμματος:
    - **το πλήθος των συμβολοσειρών (+1)** που ακολουθούν το όνομα του προγράμματος αποθηκεύεται την μεταβλητή **argc**
    - **τα ορίσματα της γραμμής εντολής** έχουν αποθηκευτεί ως συμβολοσειρές στις διαδοχικές θέσεις **(argv[1],argv[2],…argv[argc-1])**
  - Η συμβολοσειρά argv[0] αποθηκεύει το όνομα του εκτελέσιμου προγράμματος μαζί με το πλήρες όνομα του καταλόγου του αρχείου.

Α. Επικοινωνία με το Λειτουργικό
3. Ορίσματα Γραμμής Εντολής
1. Ορισμός της main με ορίσματα

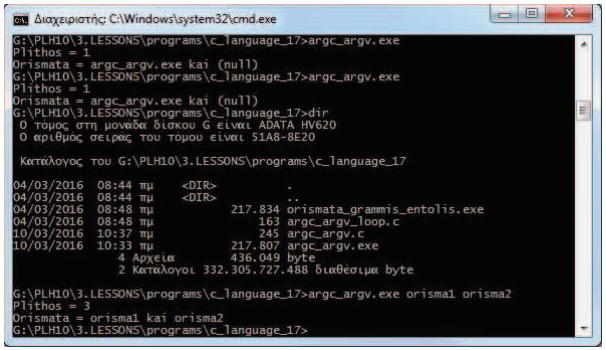
- Μεταγλωττίστε το ακόλουθο πρόγραμμα:

```
/* argc_argv.c: programma epideiksis
       lipsis orismatwn apo ti grammi entolis */

#include <stdio.h>

main(int argc, char *argv[])
{

   printf("Plithos = %d",argc);

   printf("\nOrismata = %s kai %s",argv[1],argv[2]);

}
```

- Και εκτελέστε το από το DEV-C++. Θα παρατηρήσετε ότι εχουμε μήνυμα λάθους!

A. Επικοινωνία με το Λειτουργικό
3. Ορίσματα Γραμμής Εντολής
1. Ορισμός της main με ορίσματα

- Το εκτελούμε από την γραμμή εντολής διοχετεύοντας κατάλληλα ορίσματα! Παρατηρούμε ότι τα ορίσματα έχουν διοχετευτεί στο πρόγραμμα (άρα μπορούμε να το αξιοποιήσουμε στα προγράμματά μας)



A. Επικοινωνία με το Λειτουργικό
3. Ορίσματα Γραμμής Εντολής
2. Παράδειγμα με διαφορετικό πλήθος ορισμάτων

- Το ακόλουθο πρόγραμμα επιδεικνύει την αρχικοποίηση των ορισμάτων γραμμής εντολής ανάλογα με τα ορίσματα που διοχετεύουμε:

```
/* argc_argv_loop.c Epideikniei tin xrisi
   orismatwn grammis entolis */

#include <stdio.h>

main(int argc, char *argv[])
{
   int i;

   printf("argc=%d",argc);
   for (i=0; i<argc; i++)
      printf("\nargv[%d]=%s",i,argv[i]);

}
```

- Πειραματιστείτε με το πρόγραμμα βάζοντας διαφορετικό πλήθος ορισμάτων γραμμής εντολής!

A. Επικοινωνία με το Λειτουργικό
3. Ορίσματα Γραμμής Εντολής
3. Μετατροπή συμβολοσειράς σε άλλο τύπο δεδομένων

- Επειδή τα ορίσματα αποθηκεύονται ως συμβολοσειρές, πολύ χρήσιμη θα είναι η συνάρτηση

```
int atoi(char *s)
```

  - που έχει οριστεί στο `stdlib.h`
  - Η συνάρτηση παίρνει σαν όρισμα μία συμβολοσειρά (που κωδικοποιεί έναν ακέραιο) και επιστρέφει τον ακέραιο αριθμό σε τύπο int.

- Το παράδειγμα της επόμενης διαφάνειας αναδεικνύει τη χρήση της atoi.

A. Επικοινωνία με το Λειτουργικό
3. Ορίσματα Γραμμής Εντολής
3. Μετατροπή συμβολοσειράς σε άλλο τύπο δεδομένων

- Το πρόγραμμα αναδεικνύει τη χρήση της atoi:

```
/* atoi.c: epideikniei tin sinartisi atoi */

#include <stdio.h>

main(int argc, char *argv[])
{
   int x,y;

   if(argc!=3)
   {
       printf("Prepei na eisagete 2 orismata!");
   }
   else
   {
      x=atoi(argv[1]);
      y=atoi(argv[2]);

      printf("\nOrismata = %d kai %d",x,y);
   }
}
```

A. Επικοινωνία με το Λειτουργικό
3. Ορίσματα Γραμμής Εντολής
3. Μετατροπή συμβολοσειράς σε άλλο τύπο δεδομένων

- Αντίστοιχα ορίζονται συναρτήσεις που μετατρέπουν μια συμβολοσειρά σε άλλους τύπους δεδομένων:

Συναρτηση	Ενέργεια	Βιβλιοθήκη
`int atoi(char *s)`	Μετατρέπει την s σε int	stdlib.h
`long atol(char *s)`	Μετατρέπει την s σε long	stdlib.h
`long long atoll(char *s)`	Μετατρέπει την s σε long long	stdlib.h
`double atof(char *s)`	Μετατρέπει την s σε double	stdlib.h

A. Επικοινωνία με το Λειτουργικό
3. Ορίσματα Γραμμής Εντολής
4. Προχωρημένες συναρτήσεις μετατροπής συμβολοσειράς

- Αντίστοιχα ορίζονται συναρτήσεις που μετατρέπουν μια συμβολοσειρά σε άλλους τύπους δεδομένων:

Συνάρτηση	Ενέργεια	Βιβλιοθήκη
`double strtod( const char *s, char **p);`	Μετατρέπει την s σε double	stdlib.h
`long strtol( const char *s, char **p, int base );`	Μετατρέπει την s σε long	stdlib.h
`unsigned long strtoul( const char *s, char **p, int base );`	Μετατρέπει την s σε long long	stdlib.h

- Οι συναρτήσεις αυτές δέχονται ως όρισμα μία συμβολοσειρά και την μετατρέπουν στον αντίστοιχο τύπο δεδομένων, αλλά:
  - Μέσω του δείκτη p επιστρέφουν και την υπόλοιπη συμβολοσειρά
- Π.χ. αν η συμβολοσειρά που βάλει ο χρήστης είναι 0.54fd, τότε η strtod θα επιστρέψει το 0.54 και ο δείκτης p θα είναι ίσος με τη συμβολοσειρά «fd».

- Το όρισμα base καθορίζει το σύστημα αρίθμηση στο οποίο είναι γραμμένος ο αριθμός s (π.χ. μπορεί να είναι γραμμένος στο 8-αδικό σύστημα αρίθμησης)
  - Για δεκαδικό σύστημα αρίθμησης, το όρισμα μπορεί να είναι 0.

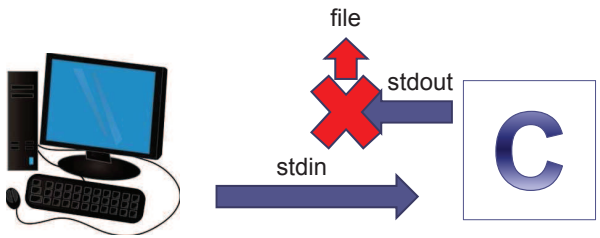
A. Επικοινωνία με το Λειτουργικό
4. Ανακατεύθυνση Εισόδου-Εξόδου
1. Ανακατεύθυνση Εξόδου

- Η ανακατεύθυνση εξόδου χρησιμοποιείται για να αποθηκεύονται τα αποτελέσματα της εξόδου ενός προγράμματος σε αρχείο κειμένου του υπολογιστή.

- Η ανακατεύθυνση γίνεται μέσω εντολής στο λειτουργικό:

```
program > ονομα-αρχειου
```

- Με τον τρόπο αυτό μπορούμε να «κρατάμε» τα αποτελέσματα της εκτέλεσης σε ένα αρχείο



A. Επικοινωνία με το Λειτουργικό
4. Ανακατεύθυνση Εισόδου-Εξόδου
1. Ανακατεύθυνση Εξόδου

- Για παράδειγμα έστω το ακόλουθο πρόγραμμα:

```
/* redirect_output.c: tha to xrisimopoiisoume
                      gia anakateuthinsi eksodou */

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

main()
{
   int i;

   srand(time(NULL));
   for (i=0; i<20; i++)
      printf("%d\n",rand()%1000);
}
```

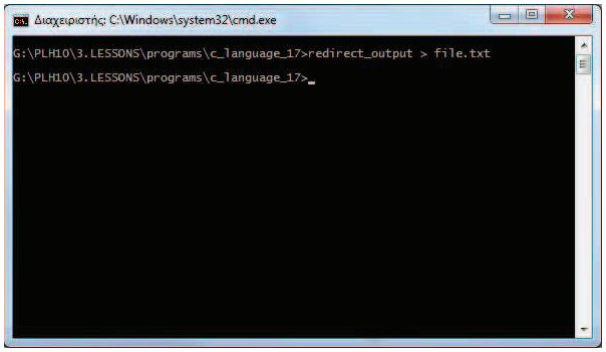
- Το οποίο είναι ένα τυπικό πρόγραμμα εκτύπωση τυχαίων τριψήφιων αριθμών!

A. Επικοινωνία με το Λειτουργικό
4. Ανακατεύθυνση Εισόδου-Εξόδου
1. Ανακατεύθυνση Εξόδου

- Και το τρέξουμε από την κονσόλα ως εξής:

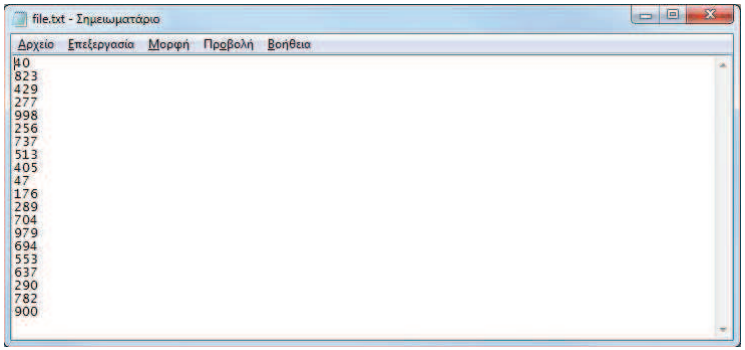
```
redirect_output > file.txt
```

  - Τότε δεν θα παρατηρήσουμε κάποια έξοδο στην οθόνη:



A. Επικοινωνία με το Λειτουργικό
4. Ανακατεύθυνση Εισόδου-Εξόδου
1. Ανακατεύθυνση Εξόδου

- Διότι η έξοδος θα έχει ανακατευθυνθεί από την stdout στο αρχείο κειμένου file.txt
  - (Το οποίο μπορούμε να χειριστούμε πλέον ως ένα ακόμη αρχείο κειμένου!)
  - Έτσι π.χ. αν το ανοίξουμε θα δούμε το αποτέλεσμα των ενεργειών μας:



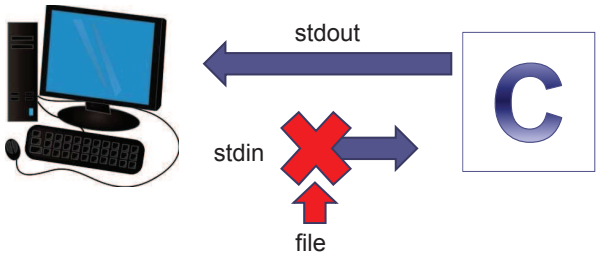
A. Επικοινωνία με το Λειτουργικό
4. Ανακατεύθυνση Εισόδου-Εξόδου
2. Ανακατεύθυνση Εισόδου

- Η ανακατεύθυνση εισόδου χρησιμοποιείται για να μην διαβάζονται τα δεδομένα από την stdin αλλά π.χ. από ένα αρχείο!

- Η ανακατεύθυνση γίνεται μέσω εντολής στο λειτουργικό:

```
program < ονομα-αρχειου
```

- Με τον τρόπο αυτό μπορούμε να διαβάζουμε τα δεδομένα από αρχείο.



A. Επικοινωνία με το Λειτουργικό
4. Ανακατεύθυνση Εισόδου-Εξόδου
2. Ανακατεύθυνση Εισόδου

- Το ακόλουθο πρόγραμμα θα αρχικοποιούσε από την είσοδο έναν πίνακα 20 ακεραίων:

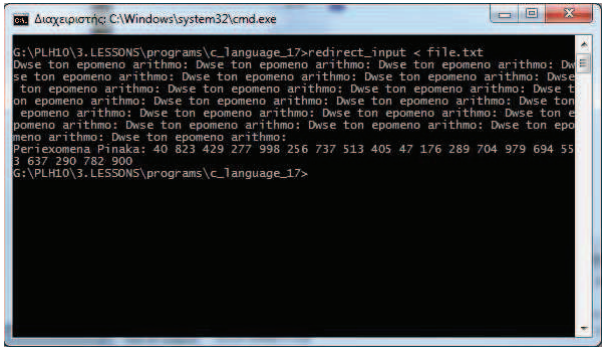
```
/* redirect_input.c: tha to xrisimopoiisoume
                  gia anakateuthinsi eisodou */

#include <stdio.h>
#define N 20

main()
{
   int i;
   int array[N];

   for (i=0; i<N; i++)
   {
      printf("Dwse ton epomeno arithmo: ");
      scanf("%d", &array[i]);
   }
   printf("\nPeriexomena Pinaka: ");
   for (i=0; i<N; i++)
      printf("%d ",array[i]);
}
```

A. Επικοινωνία με το Λειτουργικό
4. Ανακατεύθυνση Εισόδου-Εξόδου
2. Ανακατεύθυνση Εισόδου

- Και το τρέξουμε από την κονσόλα ως εξής:

```
redirect_input < file.txt
```

- Τότε θα παρατηρήσουμε την εξής έξοδο:



A. Επικοινωνία με το Λειτουργικό
4. Ανακατεύθυνση Εισόδου-Εξόδου
2. Ανακατεύθυνση Εισόδου

- Συνεπώς ο πίνακας μας αρχικοποιείται με τις τιμές που υπάρχουν στο αρχείο!:

Παρατηρήσεις:

1. Η ανακατεύθυνση εισόδου – εξόδου είναι μέρος του λειτουργικού συστήματος (Windows, UNIX) κ.λπ. και όχι μέρος της C
   1. Για το λόγο αυτό δεν θα επεκταθούμε περαιτέρω στη μελέτη αυτή.
   2. Για περισσότερα απαιτείται μελέτη των λειτουργικών συστημάτων.
2. Επίσης αν και φαίνεται πολύ ελκυστική διότι πλέον δεν θα χρειάζεται η πληκτρολόγηση της εισόδου,
   - Θα προτιμήσουμε την χρησιμοποίηση έτοιμων συναρτήσεων που μας παρέχει η C για τον χειρισμό αρχείων (όπως π.χ. η fprintf και η fscanf) τις οποίες και θα μελετήσουμε σε επόμενο μάθημα

B. Ασκήσεις
Εφαρμογή 1: Διαχείριση Συμβολοσειρών

Γράψτε ένα πρόγραμμα το οποίο:

1. Θα διαβάζει από τα ορίσματα γραμμής εντολής ακεραίους αριθμούς
2. Θα δεσμεύει δυναμικά έναν πίνακα τόσων θέσεων όσα και τα ορίσματα που δέχθηκε
3. Θα ταξινομεί τον πίνακα με χρήση της insertion-sort
4. Θα τυπώνει τον ταξινομημένο πίνακα.

Υποδείξεις:

1. Το μέγεθος του πίνακα θα είναι ίσο με argc-1
2. Χρησιμοποιήστε την υλοποίηση της insertion-sort που κατασκευάσαμε στο μάθημα «Αλγόριθμοι σε C – Μάθημα 3: Ταξινόμηση Πίνακα»
3. Μην ξεχάσετε να απελευθερώσετε τη μνήμη που δεσμεύσατε δυναμικά.

B. Ασκήσεις
Εφαρμογή 2: Κατασκευή ενός medium

Προσληφθήκαμε ως προγραμματιστές στην εταιρία προβλέψεων «Μπάμπα Βάνγκα» ώστε να γράψουμε ένα πρόγραμμα το οποίο θα διαβάζει μέσω ορισμάτων γραμμής εντολής: Το όνομα, το επώνυμο, την ηλικία και το ύψος ενός ατόμου (με αυτήν τη σειρά) και:

1. Θα υπολογίζει τον λεξάριθμο του ονόματος (άθροισμα των αντίστοιχων αριθμών (A=1, B=2, C=3 κ.λπ.))
2. Θα υπολογίζει τον λεξάριθμο του επωνύμου (ομοίως με το όνομα)
3. Θα πολλαπλασιάζει την ηλικία επί 10
4. Θα πολλαπλασιάζει το ύψος επί 100
5. Θα αθροίζει τα επιμέρους αποτελέσματα

Έπειτα το πρόγραμμα-Μπάμπα θα προβλέπει με βάση το αποτέλεσμα:

- Αν είναι μεταξύ 1 και 300: Θα τυπώνει «Η Μπάμπα λέει: Σεισμός και Καταποντισμός»
- Αν είναι μεταξύ 301 και 600: Θα τυπώνει «Η Μπάμπα λέει: Λιμός και Πόλεμος»
- Αν είναι μεταξύ 601 και 1000: Θα τυπώνει «Η Μπάμπα λέει: Αρρώστια και Κασίδα»
- Αν είναι πάνω από 1000: Θα τυπώνει «Η Μπάμπα λέει: Δώσε άλλα 100 ευρώ», τυχαία θα επιλέγει έναν αριθμό από το 1 έως το 1000 και θα τυπώνει το ανάλογο μήνυμα.