Η ΓΛΩΣΣΑ C

Μάθημα 20:

Αρχεία

Δημήτρης Ψούνης

Περιεχόμενα Μαθήματος

A. Αρχεία
1. Γλώσσα C και αρχεία
1. Αρχεία

- Η C μας δίνει τη δυνατότητα να χειριστούμε αρχεία.
  - Συγκεκριμένα μπορούμε μέσω του προγράμματος μας να:
    - Δημιουργήσουμε ένα καινούριο αρχείο
    - Τροποποιήσουμε ένα ήδη υπάρχον αρχείο
    - Διαγράψουμε αρχεία

- Η δυνατότητα αυτή είναι πολύ σημαντική μιας και:
  - Η πιο συνηθισμένη λειτουργία ενός προγράμματος είναι:
    - Να ανοίγει ένα αρχείο
    - Να επεξεργάζεται τα δεδομένα του
    - Να αποθηκεύει τις αλλαγές στο αρχείο.

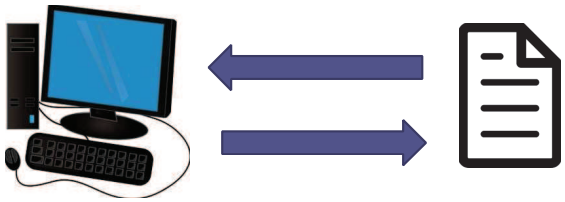
- Η C προσφέρει δύο τρόπους για την διαχείριση αρχείων:
  - **Αρχεία κειμένου**: Στα οποία γράφουμε/διαβάζουμε χαρακτήρες
    - (και μπορούμε να τα διαβάσουμε)
  - **Δυαδικά αρχεία**: Στα οποία γράφουμε/διαβάζουμε bytes.
    - (και δεν μπορούμε να τα διαβάσουμε)

A. Αρχεία
1. Γλώσσα C και αρχεία
2. Τρόπος Λειτουργίας

- Σημαντικό: Ο τρόπος λειτουργίας της C όταν επεξεργάζεται ένα αρχείο είναι:
  1. **Άνοιγμα του αρχείου**
     - Χρησιμοποιείται η συνάρτηση fopen
  2. **Επεξεργασία του αρχείου**
     - Χρησιμοποιούνται διάφορες συναρτήσεις, ανάλογα με το αν το αρχείο είναι δυαδικό ή αρχείο κειμένου.
  3. **Κλείσιμο του αρχείου**
     - Χρησιμοποιείται η συνάρτηση fclose
     - Προσοχή, ότι θα πρέπει πάντα να κλείνει το αρχείο, αλλιώς μπορεί να παρουσιαστούν μη επιθυμητές συμπεριφορές.

A. Αρχεία
1. Γλώσσα C και αρχεία
3. Ροές (streams)

- Στο μάθημα 15, συζητήσαμε για τις ροές εισόδου - εξόδου
  - Την προκαθορισμένη ροή εισόδου (stdin) από την οποία διαβάζουμε χαρακτήρες από το πληκτρολόγιο.
  - Την προκαθορισμένη ροή εξόδου (stdout) στην οποία γράφουμε χαρακτήρες στην οθόνη.



- Ένα αρχείο της C δημιουργεί μια ροή από και προς ένα αρχείο
- Η διαχείριση γίνεται μέσω μίας δομής (που ονομάζεται FILE) και έχει οριστεί στο stdio.h
  - Έχει πολλά στοιχεία που δεν χρειάζεται να ξέρουμε.
  - Αλλά δουλεύει με παρόμοιο τρόπο όπως ένας επεξεργαστής κειμένου, δηλαδή «υπάρχει» ένας δρομέας στο αρχείο, που δείχνει ποιο είναι το επόμενο σημείο που πρόκειται να γράψουμε.

A. Αρχεία
2. Αρχεία Κειμένου
1. Άνοιγμα Αρχείου

- Για να ανοίξουμε ένα αρχείο χρησιμοποιούμε τη συνάρτηση fopen:

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

  - Και έχει οριστεί στη βιβλιοθήκη: `stdio.h`

- Η συνάρτηση δέχεται σαν όρισμα το όνομα ενός αρχείου (filename) και τον τρόπο με τον οποίο ανοίγει (mode):
  - Αν το άνοιγμα πραγματοποιηθεί αρχικοποιείται μία δομή τύπου FILE με τις απαραίτητες πληροφορίες για τον χειρισμό του αρχείου
    - και επιστρέφεται ένας δείκτης σε αυτήν τη δομή
  - Αν το άνοιγμα δεν πραγματοποιηθεί (π.χ. δεν υπάρχει το αρχείο, ή ο κατάλογος κ.λπ.) επιστρέφει NULL.

Σημείωση:
- Η δομή FILE περιέχει διάφορες πληροφορίες για το χειρισμό του αρχείου, αλλά δεν χρειάζεται να τις ξέρουμε
- Εμείς απλά θα διοχετεύουμε τον δείκτη που μας επέστρεψε η fopen στις συναρτήσεις του χειρισμού του αρχείου.

A. Αρχεία
2. Αρχεία Κειμένου
1. Άνοιγμα Αρχείου

- Το mode είναι μια συμβολοσειρά, η οποία καθορίζει αν θέλουμε να ανοιξουμε το αρχείο π.χ. για γράψιμο ή για διάβασμα.
- Οι τρόποι ανοίγματος ενός αρχείου κειμένου συνοψίζονται στον πίνακα:

mode	Λειτουργία	Αν δεν υπάρχει το αρχείο	Αν υπάρχει το αρχείο
r	Διάβασμα	Επιστρέφει NULL.	Δείκτης στην αρχή του αρχείου.
w	Γράψιμο	Δημιουργείται το αρχείο.	Σβήνονται τα περιεχόμενα του.
a	Προσάρτηση	Δημιουργείται το αρχείο.	Δείκτης στο τέλος του αρχείου.
r+	Διάβασμα Ενημέρωση	Επιστρέφει NULL.	Δείκτης στην αρχή του αρχείου. Γράψιμο οπουδήποτε στο αρχείο.
w+	Γράψιμο Ενημέρωση	Δημιουργείται το αρχείο.	Σβήνονται τα περιεχόμενα του.
a+	Προσάρτηση Ενημέρωση	Δημιουργείται το αρχείο.	Δείκτης στην αρχή του αρχείου. Γράψιμο στο τέλος του αρχείου.

- Ενώ για το άνοιγμα ενός δυαδικού αρχείου θέτουμε ένα “b” ακόμη στο mode.
  - π.χ. r+b προσδιορίζει δυαδικό αρχείο για διάβασμα και ενημέρωση.

A. Αρχεία
2. Αρχεία Κειμένου
2. Κλείσιμο Αρχείου

- Το κλείσιμο του αρχείου γίνεται με τη συνάρτηση:

```
int fclose(FILE *fp)
```

  - Και έχει οριστεί στη βιβλιοθήκη: `stdio.h`

- Η συνάρτηση δέχεται σαν όρισμα τον δείκτη σε αρχείο και:
  - Επιστρέφει 0, αν όλα πήγαν καλά στο κλείσιμό του.
  - Επιστρέφει -1, αν συνέβη κάποιο λάθος

- Υπάρχει και η συνάρτηση που κλείνει όλα τα ανοικτά αρχεία:

```
int fcloseall(void)
```

Σημείωση:
- Είναι συνηθισμένο προγραμματιστικό λάθος να ξεχνάμε να κλείσουμε το αρχείο.
- Το αρχείο πιθανότατα δεν θα δεχθεί τις αλλαγές, αν ξεχάσουμε να το κλείσουμε.

A. Αρχεία
2. Αρχεία Κειμένου
3. Η συνάρτηση fprintf

- Η συνάρτηση fprintf χρησιμοποιείται για να γράψουμε σε ένα αρχείο
- Το πρωτότυπο της είναι:

```
int fprintf(FILE *fp, char *fmt, …)
```

  - Το 1ο όρισμα είναι ο δείκτης αρχείου στο οποίο θέλουμε να γράψουμε.
  - Τα υπόλοιπα ορίσματα χρησιμοποιούνται ακριβώς με τον ίδιο τρόπο όπως στην printf.
  - Και έχει οριστεί στη βιβλιοθήκη: stdio.h

- Έτσι για παράδειγμα η εντολή:

```
fprintf(fp, “O akeraios einai: %d“, x);
```

  - Αν π.χ. το x είναι 5, θα τυπώσει στο αρχείο την γραμμή:

```
O akeraios einai: 5
```

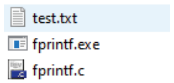
- Βλέπουμε ένα ολοκληρωμένο παράδειγμα στην επόμενη διαφάνεια:

A. Αρχεία
2. Αρχεία Κειμένου
3. Η συνάρτηση fprintf

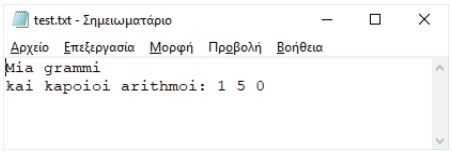
```
/* fprintf.cpp Grapsimo se arxeio */
#include <stdio.h>
#include <stdlib.h>

main() {
  FILE *fp;
  int x=1;

  /* Anoigma arxeiou */
  fp = fopen("test.txt","w");
  if (fp==NULL) {
    printf("Lathos sto anoigma tou arxeiou");
    exit(0);
  }

  /* Grapsimo se arxeio */
  fprintf(fp, "Mia grammi\n");
  fprintf(fp, "kai kapoioi arithmoi: %d %d %d", x, x*5, x/2);

  /* Kleisimo arxeiou */
  fclose(fp);
}
```


A. Αρχεία
2. Αρχεία Κειμένου
3. Η συνάρτηση fprintf

- Με το πέρας της εκτέλεσης του προγράμματος έχει δημιουργηθεί το αρχείο test.txt



  - (στον ίδιο φάκελο με το εκτελέσιμο)
- Με τα εξής περιεχόμενα:



A. Αρχεία
2. Αρχεία Κειμένου
4. Οι συναρτήσεις putc, fputc και fputs

- Δύο βοηθητικές συναρτήσεις για το γράψιμο σε αρχείο είναι οι εξής:
  - Χρησιμοποιούμε τη συνάρτηση putc για να γράψουμε έναν χαρακτήρα στο αρχείο:

```
int putc(int ch, FILE *fp)
```

    - Π.χ. για να γράψουμε τον χαρακτήρα c, έχουμε:

```
putc(‘c’, fp);
```

    - Επιστρέφει το χαρακτήρα που γράφτηκε, ή EOF αν κάτι πήγε στραβά.
  - Χρησιμοποιούμε τη συνάρτηση putc για να γράψουμε έναν χαρακτήρα στο αρχείο:

```
char fputc(int ch, FILE *fp)
```

    - Π.χ. για να γράψουμε τη συμβολοσειρά name:

```
fputc(‘a’, fp);
```

    - Επιστρέφει 0 αν κάτι πήγε στραβά ή μη μηδενική τιμή αν όλα πήγαν καλά.

A. Αρχεία
2. Αρχεία Κειμένου
4. Οι συναρτήσεις putc, fputc και fputs

- Η συνάρτηση fputs:

```
int fputs(char *s, FILE *fp)
```

  - Γράφει τη συμβολοσειρά s στο αρχείο με δείκτη fp

```
fputs("This is a string", fp);
```

  - Επιστρέφει το χαρακτήρα που γράφτηκε, ή EOF αν κάτι πήγε στραβά.
- Επιστρέφει 1 αν όλα πήγαν καλά, και EOF σε περίπτωση λάθους.

- Παρατήρηση: Όλες οι συναρτήσεις εξόδου που είδαμε μπορούν να χρησιμοποιηθούν και για την έξοδο στην οθόνη βάζοντας στο όρισμα fp, την προκαθορισμένη ροή εξόδου stdout.

A. Αρχεία
2. Αρχεία Κειμένου
5. Η συνάρτηση fscanf

- Η συνάρτηση fscanf χρησιμοποιείται για να διαβάσουμε από ένα αρχείο κειμένου
- Το πρωτότυπο της είναι:

```
int fscanf(FILE *fp, char *fmt, ...)
```

  - Το 1ο όρισμα είναι ο δείκτης αρχείου στο οποίο θέλουμε να γράψουμε.
  - Τα υπόλοιπα ορίσματα χρησιμοποιούνται ακριβώς με τον ίδιο τρόπο όπως στην scanf.
  - Επιστρέφει το πλήθος των αντικειμένων που ταίριαξαν και διαβάστηκαν.
  - Και έχει οριστεί στη βιβλιοθήκη: stdio.h

- Έτσι για παράδειγμα η εντολή:

```
fscanf(fp, "%d", &x);
```

  - Διαβάζει έναν ακέραιο από το αρχείο.

A. Αρχεία
2. Αρχεία Κειμένου
5. Η συνάρτηση fscanf (1. Διάβασμα ενός αρχείου δεδομένων)

- Ο προγραμματιστής επιλέγει τον τρόπο με τον οποίο θα οργανώσει τα δεδομένα του σε ένα αρχείο.
- Ο τρόπος με τον οποίο οργανώνει κάποιος τα δεδομένα του σε ένα αρχείο καλείται και «γραμμογράφηση»

- Στο παράδειγμα της επόμενης διαφάνειας:
  - Ο προγραμματιστής έχει ένα αρχείο από σημεία στο επίπεδο (points.txt)
    - Στην 1η γραμμή αποθηκεύει το πλήθος των σημείων
    - Στις επόμενες γραμμές αποθηκεύει ζεύγη ακεραίων που απεικονίζουν τα σημεία.
  - Βλέπουμε την διαχείριση που πρέπει να γίνει προκειμένου να αρχικοποιήσει μια κατάλληλη δομή με τα στοιχεία που περιέχει το αρχείο

A. Αρχεία
2. Αρχεία Κειμένου
5. Η συνάρτηση fscanf (1. Διάβασμα ενός αρχείου δεδομένων)

```
/* fscanf.cpp Diavasma apo arxeio */
#include <stdio.h>
#include <stdlib.h>
#define N 100

struct point {
  int x;
  int y;
};



main() {
  FILE *fp;
  int i;
  int n; /* plithos eggrafwn */
  struct point points[N]; /* eggrafes */
```

```
  /* Diavasma apo to arxeio */
  fp = fopen("points.txt","r");
  if (fp==NULL) {
    printf("Lathos sto anoigma tou arxeiou");
    exit(0);
  }

  fscanf(fp, "%d", &n);
  for (i=0; i<n; i++)
    fscanf(fp, "%d %d", &points[i].x, &points[i].y);

  fclose(fp);

  /* Ektipwsi stin othoni */
  for (i=0; i<n; i++)
    printf("%d %d\n", points[i].x, points[i].y);


}
```

A. Αρχεία
2. Αρχεία Κειμένου
6. Οι συναρτήσεις getc, fgetc και fgets

- Οι συναρτήσεις
  `int getc(FILE *fp)` `int fgetc(FILE *fp)`
  - διαβάζουν έναν χαρακτήρα (ή τον EOF) από το αρχείο και τον επιστρέφουν.
  - και οι δύο έχουν οριστεί στο `stdio.h`
  - αν αποτύχει το διάβασμα για κάποιο λόγο επιστρέφει EOF.
- Ένας συνηθισμένος βρόχος που μπορεί να διαβάσει και να τυπώσει ένα ολόκληρο αρχείο κειμένου είναι ο ακόλουθος:

```
while (1)
{
   c  = fgetc(fp);
   if (c == EOF)
       break;
    printf("%c", c);
}
```

- ή πιο συνεπτυγμένα ως εξής:

```
while ((c=fgetc(fp))!=EOF)
    printf("%c", c);
```

- Βλέπουμε και ολοκληρωμένο το πρόγραμμα στην επόμενη διαφάνεια

A. Αρχεία
2. Αρχεία Κειμένου
6. Οι συναρτήσεις getc και fgetc (1. Διάβασμα ενός αρχείου κειμένου)

```
/* fgetc.c Diavasma kai ektipwsi enos arxeiou keimenoy */
#include <stdio.h>
#include <stdlib.h>

main() {
  FILE *fp;
  char c;

  fp = fopen("test.txt","r");
  if (fp==NULL) {
    printf("Lathos sto anoigma tou arxeiou");
    exit(0);
  }

  while ((c=fgetc(fp))!=EOF)
    printf("%c", c);

  fclose(fp);
}
```


A. Αρχεία
2. Αρχεία Κειμένου
6. Οι συναρτήσεις getc, fgetc και fgets

- Το πρωτότυπο της συνάρτησης fgets είναι:

```
char *fgets(char *str, int n, FILE *fp)
```

  - Και έχει οριστεί στο αρχείο βιβλιοθήκης: `stdio.h`
- Η συνάρτηση διαβάζει μία συμβολοσειρά από την ροή fp και την αποθηκεύει στην str μέχρι:
  - Είτε διαβάζει τον χαρακτήρα '\n' τον οποίο αντικαθιστά με '\0'
  - Είτε διαβάσει n-1 χαρακτήρες. Τότε προσθέτει τον χαρακτήρα '\0' στο τέλος και τερματίζει.
- Με απλά λόγια η fgets βάζει όριο στους χαρακτήρες που θα διαβάσει προκειμένου να χωράνε στην συμβολοσειρά που διοχετεύουμε ως όρισμα.
- Το παράδειγμα της επόμενης διαφάνειας αναδεικνύει την χρήση των παραπάνω

A. Αρχεία
3. Δυαδικά Αρχεία
1. Άνοιγμα Δυαδικού Αρχείου

- Τα δυαδικά αρχεία (binary files) περιέχουν bytes δεδομένων
  - και συγκεκριμένα μεταβλητές, ακριβώς όπως αποθηκεύονται στη μνήμη του υπολογιστή.
- Κατά συνέπεια τα δυαδικά αρχεία είναι αναγνώσιμα μόνο από ένα πρόγραμμα που τα επεξεργάζεται και εμείς ανοίγοντας τα θα βλέπουμε «κινέζικα».

- Για να ανοίξουμε ένα δυαδικό αρχείο:
  - **Χρησιμοποιούμε την fopen με τον ίδιο τρόπο**
    - **προσθέτοντας ένα "b" στον τρόπο ανοίγματος (mode)**
    - σε όποιο σημείο του mode θέλουμε.
  - Για παράδειγμα η ακόλουθη εντολή ανοίγει ένα δυαδικό αρχείο για διάβασμα:

```
fopen("test.txt", "rb");
```

  - ενώ η ακόλουθη εντολή ανοίγει ένα δυαδικό αρχείο για προσάρτηση και ενημέρωση:

```
fopen("test.txt", "a+b");
```

- και βεβαιώς το αρχείο πρέπει να κλείσει με τη fclose.

A. Αρχεία
3. Δυαδικά Αρχεία
2. Η συνάρτηση fwrite

- Η συνάρτηση fwrite χρησιμοποιείται για να γράψουμε σε ένα δυαδικό αρχείο
- Το πρωτότυπο της είναι:

```
int fwrite(void *buf, int size, int count, FILE *fp)
```

  - buf: Διεύθυνση των δεδομένων που θα γράψουμε
  - size: Μέγεθος σε bytes ενός δεδομένου που θα γράψουμε
  - count: Πλήθος των δεδομένων που θα γράψουμε
  - Επιστρέφει το πλήθος των δεδομένων που γράφτηκαν
  - Έχει οριστεί στο `stdio.h`
- Παραδείγματα:

```
fwrite(&x, sizeof(int), 1, fp);
```

  - Για να γράψουμε την ακέραια μεταβλητή x
  - Αναμένουμε να επιστρέψει 1.

```
fwrite(array, sizeof(int), N, fp);
```

  - Για να γράψουμε τον πίνακα N ακεραίων με όνομα array.
  - Αναμένουμε να επιστρέψει N
  - Θα μπορούσαμε να γράψουμε τον πίνακα και ως εξής: `fwrite(array, sizeof(array), 1, fp);`

A. Αρχεία
3. Δυαδικά Αρχεία
2. Η συνάρτηση fwrite (παράδειγμα)

```
/* fwrite.c Grafei enan pinaka domwn se arxeio */
#include <stdio.h>
#include <stdlib.h>

#define N 10

struct point {
   int x;
   int y;
};

main() {
  FILE *fp;
  char c;
  int i;
  struct point data[N];

  fp = fopen("binarydata.dat","wb");
  if (fp==NULL) {
    printf("Lathos sto anoigma tou arxeiou");
    exit(0);
  }

  /* Tyxaia Arxikopoiisi */
  for (i=0; i<N; i++)
  {
     data[i].x=rand()%100;
     data[i].y=rand()%100;
  }

  /* Grapsimo sto arxeio */
  fwrite(data, sizeof(struct point), N, fp);

  fclose(fp);
}
```


A. Αρχεία
3. Δυαδικά Αρχεία
2. Η συνάρτηση fwrite

- Το πρόγραμμα της προηγούμενης διαφάνειας κατασκευάζει ένα αρχείο με ζεύγη ακεραίων που απεικονίζουν σημεία.
- Το αρχείο δεν είναι αναγνώσιμο με γυμνό μάτι.
  - Αν το ανοίξουμε με ένα σημειωματάριο θα δούμε «κινέζικα»

- Αν και μπορούμε να ονομάσουμε τα αρχεία όπως θέλουμε,
  - συνηθίζεται τα αρχεία κειμένου να έχουν επέκταση .txt
  - ενώ τα δυαδικά αρχεία να έχουν επέκταση .dat

A. Αρχεία
3. Δυαδικά Αρχεία
3. Η συνάρτηση fread

- Η συνάρτηση fread χρησιμοποιείται για να διαβάσουμε από ένα δυαδικό αρχείο
- Το πρωτότυπο της είναι:

```
int fread(void *buf, int size, int count, FILE *fp)
```

  - buf: Διεύθυνση των δεδομένων στα οποία θα γράψουμε από τα δεδομένα του αρχείου
  - size: Μέγεθος σε bytes ενός δεδομένου που θα διαβάσουμε
  - count: Πλήθος των δεδομένων που θα διαβάσουμε
  - Επιστρέφει το πλήθος των δεδομένων που διαβάστηκαν
  - Έχει οριστεί στο `stdio.h` και πρακτικά είναι συμμετρική στη fwrite.
- Παραδείγματα:

```
fread(&x, sizeof(int), 1, fp);
```

  - Για να διαβάσουμε από το αρχείο μία ακέραια μεταβλητή (και να την αποθ/με στη x)
  - Αναμένουμε να επιστρέψει 1.

```
fread(array, sizeof(int), N, fp);
```

  - Για να διαβάσουμε από το αρχείο έναν πίνακα N ακεραίων (και αποθ/ση στο array).
  - Αναμένουμε να επιστρέψει N
  - Θα μπορούσαμε να γράψουμε τον πίνακα και ως εξής: `fread(array, sizeof(array), 1,fp);`

A. Αρχεία
3. Δυαδικά Αρχεία
3. Η συνάρτηση fread (παράδειγμα)

```
/* fread.c Diavasma dyadikoy arxeioy */
#include <stdio.h>
#include <stdlib.h>

#define N 10

struct point {
   int x;
   int y;
};

main() {
  FILE *fp;
  char c;
  int i;
  struct point data[N];
```

```
  fp = fopen("binarydata.dat","rb");
  if (fp==NULL) {
    printf("Lathos sto anoigma tou arxeiou");
    exit(0);
  }

  /* Diavasma apo to arxeio */
  fread(data, sizeof(struct point), N, fp);

  /* Ektypwsi stin othoni */
  for (i=0; i<N; i++)
     printf("%d %d\n", data[i].x, data[i].y);

  fclose(fp);
}
```


A. Αρχεία
4. Τυχαία Προσπέλαση
1. Ο «δρομέας» ενός αρχείου

- Κάθε αρχείο που έχουμε ανοίξει, έχει μεταξύ άλλων στη δομή FILE και έναν δρομέα (cursor) που δείχνει σε ποιο σημείο βρισκόμαστε:
  - Στο σημείο αυτό θα γίνει το επόμενο διάβασμα ή γράψιμο.
- Έχουμε τη δυνατότητα να μετακινήσουμε αυτόν τον «δρομέα» ώστε να ελέγχουμε σε ποιο σημείο του αρχείου θα γίνει το επόμενο διάβασμα ή γράψιμο.
- Μπορούμε να:
  - Γυρίσουμε τον δρομέα στην αρχή του αρχείου (rewind)
  - Πάρουμε την τρέχουσα θέση του δρομέα (ftell)
  - Μετακινήσουμε το δρομέα σε μία θέση (fseek)
- Οι συναρτήσεις αυτές, μπορούν να χρησιμοποιηθούν τόσο σε αρχεία κειμένου, όσο και σε δυαδικά αρχεία.

A. Αρχεία
4. Τυχαία Προσπέλαση
2. Η συνάρτηση rewind

- Η συνάρτηση:

```
void rewind(FILE *fp)
```

  - επιστρέφει το δρομέα στην αρχή του αρχείου. Έχει οριστεί στο stdio.h
- Το ακόλουθο παράδειγμα δείχνει τη λειτουργία της συνάρτησης rewind.
  - Διαβάζει 5 χαρακτήρες του αρχείου και έπειτα το διαβάζει όλο από την αρχή.

```
/* rewind.c */
#include <stdio.h>
#include <stdlib.h>

main() {
  FILE *fp;
  char c;
  int i;

  fp = fopen("test.txt","r");
  if (fp==NULL) {
    printf("Lathos sto anoigma tou arxeiou");
    exit(0);
  }
```

```
  for (i=0; i<5; i++)
      printf("%c",fgetc(fp));
  printf("\n");

  rewind(fp);

  while ((c=fgetc(fp))!=EOF)
    printf("%c", c);

  fclose(fp);
}
```


A. Αρχεία
4. Τυχαία Προσπέλαση
3. Η συνάρτηση ftell

- Η συνάρτηση:

```
long ftell(FILE *fp)
```

  - Επιστρέφει την απόσταση του δρομέα σε bytes από την αρχή του αρχείου
  - Η αρχή έχει τεθεί να είναι το 0.
- Επεκτείνουμε το προηγούμενο πρόγραμμα:

```
/* ftell.c */
#include <stdio.h>
#include <stdlib.h>

main() {
  FILE *fp;
  char c;
  int i;

  fp = fopen("test.txt","r");
  if (fp==NULL) {
    printf("Lathos sto anoigma tou arxeiou");
    exit(0);
  }
```

```
  printf("\npos: %d\n", ftell(fp));

  for (i=0; i<5; i++)
     printf("%c",fgetc(fp));

  printf("\npos: %d\n", ftell(fp));

  rewind(fp);

  printf("\npos: %d\n", ftell(fp));

  while ((c=fgetc(fp))!=EOF)
     printf("%c", c);

  printf("\npos: %d\n", ftell(fp));

  fclose(fp);
}
```

A. Αρχεία
4. Τυχαία Προσπέλαση
4. Η συνάρτηση fseek

- Η συνάρτηση:

```
int fseek(FILE *fp, long offset, int origin)
```

  - Τοποθετεί το δρομέα στη θέση που απέχει offset bytes από
    - την αρχή του αρχείου (αν origin = SEEK_SET)
    - την τρέχουσα θέση του δρομέα (αν origin = SEEK _CUR)
    - το τέλος του αρχείου (αν origin = SEEK_END)
  - Η αρχή έχει τεθεί να είναι το 0.
  - και έχει οριστεί στο `stdio.h`
  - όπως και οι συμβολικές σταθερές SEEK_SET(0), SEEK_CUR(1), SEEK_END(2)
    - (στην παρένθεση είναι οι αντίστοιχες αριθμητικές τιμές τους)
  - Επιστρέφει 0 αν όλα πήγαν καλά, ή μη μηδενική τιμή αν συνέβη κάποιο λάθος.

A. Αρχεία
4. Τυχαία Προσπέλαση
4. Η συνάρτηση fseek

```
/* fseek.c */
#include <stdio.h>
#include <stdlib.h>

#define N 10

struct point {
   int x;
   int y;
};

main() {
  FILE *fp;
  int i;
  struct point data;

  fp = fopen("binarydata.dat","rb");
  if (fp==NULL) {
    printf("Lathos sto anoigma tou arxeiou");
    exit(0);
  }
  printf("Dwse simeio(0-9): ");
  scanf("%d",&i);
  fseek(fp, sizeof(struct point)*i, SEEK_SET);
  fread(&data, sizeof(struct point), 1, fp);

  printf("Simeio %d: ", i);
  printf("%d %d\n", data.x, data.y);

  fclose(fp);
}
```


A. Αρχεία
5. Εντοπισμός τέλους αρχείου
1. Το EOF και η συνάρτηση feof

- Είδαμε ότι η fgetc επιστρέφει EOF (end of file) αν δεν υπάρχουν άλλα δεδομένα για να διαβαστούν (ή αν κάτι πήγε στραβά).
  - Το EOF είναι μία τιμή (-1 συνήθως) που δίνει το σήμα ότι δεν υπάρχουν άλλα δεδομένα για διάβασμα.
- Για το λόγο αυτό, για τον εντοπισμό ότι δεν υπάρχουν άλλα δεδομένα για διάβασμα, χρησιμοποιείται η συνάρτηση feof:

```
int feof(FILE *fp)
```

  - Παίρνει σαν όρισμα το ανοικτό αρχείο και
  - επιστρέφει 1, αν ο δρομέας είναι στο τέλος του αρχείου
  - και 0 αλλιώς.

A. Αρχεία
5. Εντοπισμός τέλους αρχείου
2. feof και αρχεία κειμένου

- Βλέπουμε την τροποποίηση του προγράμματος ανάγνωσης αρχείου κειμένου με χρήση της feof:

```
/* feof.c */
#include <stdio.h>
#include <stdlib.h>

main() {
  FILE *fp;
  char c;

  fp = fopen("test.txt","r");
  if (fp==NULL) {
    printf("Lathos sto anoigma tou arxeiou");
    exit(0);
  }
  while (!feof(fp))
  {
    c=fgetc(fp);
    printf("%c", c);
  }

  fclose(fp);
}
```

A. Αρχεία
5. Εντοπισμός τέλους αρχείου
3. feof και δυαδικά αρχεία

- Η feof χρησιμοποιείται με τον ίδιο τρόπο και για τα δυαδικά αρχεία.
  - Στο πρόγραμμα αυτό διαβάζουμε τις εγγραφές ενός δυαδικού αρχείου, χωρίς να ξέρουμε εκ των προτέρων, πόσες είναι οι εγγραφές

```
/* feof_binary.c */
#include <stdio.h>
#include <stdlib.h>

struct point {
   int x;
   int y;
};

main() {
  FILE *fp;
  char c;
  int i;
  struct point data;

  fp = fopen("binarydata.dat","rb");
  if (fp==NULL) {
    printf("Lathos sto anoigma tou arxeiou");
    exit(0);
  }

  /* Diavasma apo to arxeio */
  while (!feof(fp))
  {
     if(fread(&data, sizeof(struct point), 1, fp)==1)
              printf("%d %d\n", data.x, data.y);
  }

  fclose(fp);
}
```


A. Αρχεία
6. Συναρτήσεις Διαχείρισης Αρχείων
1. Διαγραφή Αρχείου

- Η συνάρτηση remove χρησιμοποιείται για να διαγράψει ένα αρχείο.
- Το πρωτότυπο είναι:

```
int remove(const char *filename)
```

  - Παίρνει ως όρισμα το όνομα του αρχείου και το διαγράφει
  - Επιστρέφει 0 αν το αρχείο διαγράφηκε και -1 αλλιώς
- Παράδειγμα χρήσης:

```
if (remove("test.txt")==-1)
   printf("Lathos stin diagrafi tou arxeioy");
```


A. Αρχεία
6. Συναρτήσεις Διαχείρισης Αρχείων
2. Μετονομασία Αρχείου

- Η συνάρτηση rename χρησιμοποιείται για να μετονομοάσει ένα αρχείο.
- Το πρωτότυπο είναι:

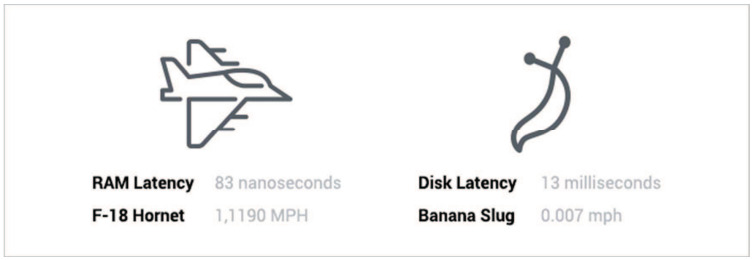
```
int rename(const char *old, const char *new)
```

  - Παίρνει ως όρισμα το όνομα του αρχείου (old) και το μετονομάζει σε new
  - Επιστρέφει 0 αν το αρχείο μετονομαστηκε και -1 αλλιώς
- Παράδειγμα χρήσης:

```
if (rename("test.txt", "new.txt")==-1)
   printf("Lathos stin metonomasia tou arxeioy");
```


A. Αρχεία
7. Περισσότερα για τις ροές (streams)
1. Ροές (streams)

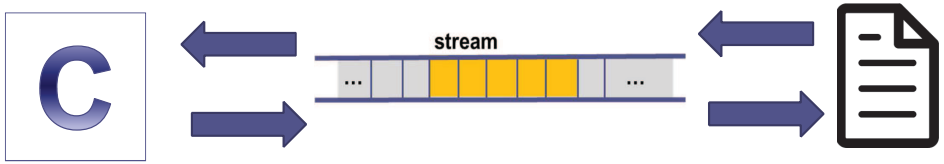
- Μία ροή (stream) είναι ένας ενδιάμεσος χώρος αποθήκευσης δεδομένων
  - Έχουμε δει μέχρι τώρα τις ροές εισόδου/εξόδου και τη ροή από/προς τα αρχεία.
  - Κοινό χαρακτηριστικό τους είναι ότι οι αλληλεπιδράσεις με αρχεία ή προς την οθόνη
    - είναι εξαιρετικά αργές σε σχέση με την εργασία με την μνήμη
  - Το ακόλουθο παράδειγμα (scoutapm.com) δείχνει μια απλουστευμένη αναλογία:



  - Η πρόσβαση στη μνήμη σε σχέση με τη πρόσβαση στο δίσκο
    - είναι τόσο πιο γρήγορη, όσο και η σχέση των F-18 με τα σαλιγκάρια.

A. Αρχεία
7. Περισσότερα για τις ροές (streams)
1. Ροές (streams)

- Για το λόγο αυτό έχουν οριστεί τα streams:
  - Αποθηκεύουν στη μνήμη περισσότερο από ότι εμείς πραγματικά ζητάμε
  - Π.χ. όταν ανοίγουμε ένα αρχείο, ενδέχεται να αποθηκεύσει στο χώρο του stream ακόμη και 2MB δεδομένων
    - τα οποία να είναι έτοιμα για τις fread που πρόκειται να κάνουμε)
  - Και κατά τη διάρκεια εκτέλεσης του προγράμματος ο χώρος αποθήκευσης του stream θα τροποιείται ώστε να έχουμε κατά το δυνατόν λιγότερες προσπελάσεις στο δίσκο.
- Συνοψίζοντας:
  - Το πρόγραμμα αλληλεπιδρά με το stream, το οποίο είναι ένας ενδιάμεσος χώρος αποθήκευσης
    - Το stream πηγαινοφέρνει δεδομένα στο πραγματικό μέσο (αρχείο, οθόνη, πληκτρολόγιο κ.λπ) και εμείς δεν έχουμε πρόσβαση σε αυτές τις ενέργειες



A. Αρχεία
7. Περισσότερα για τις ροές (streams)
1. Ροές (streams)

- Αυτός είναι και ο λόγος που:
  - αν δεν κλείσουμε ένα αρχείο είναι πολύ πιθανόν να μη γίνουν ποτέ οι αλλαγές στο πραγματικό αρχείο.
  - Η fclose αναγκάζει το stream που έχει συσχετιστεί με το αρχείο να περάσει όλες τις αλλαγές που έχουν γίνει σε αυτό στο πραγματικό αρχείο
- Συνεπώς όταν δηλώνουμε ένα δείκτη FILE *
  - άστοχα λέμε ότι είναι δείκτης σε αρχείο
  - είναι ένας δείκτης στη δομή που διαχειρίζεται το stream και άστοχα ονομάστηκε FILE (https://www.gnu.org/software/libc/manual/html_node/Streams.html)
- Και αυτός είναι και ο λόγος που πολλές από τις συναρτήσεις που είδαμε (όπως π.χ. η fprintf και η fgets) δουλεύουν κανονικά αν τους περάσουμε ως όρισμα την stdout και την stdin αντίστοιχα.
  - Διότι και αυτές είναι ροές! Συνεπώς συναρτήσεις που κάνουν διαχείριση επί της ροής είναι σχεδιασμένες μπορούν ασφαλώς να δουλέψουν και με αυτές.

A. Αρχεία
7. Περισσότερα για τις ροές (streams)
2. Η συνάρτηση fflush

- Η συνάρτηση fflush τραβάει το καζανάκι και κάνει όλες τις ενέργειες για την αποθήκευση σε ένα ρεύμα εξόδου
  - Π.χ. σε ένα αρχείο θα αποθηκεύσει στο αρχείο όλες τις αλλαγές που υπάρχουν σε αυτό στο stream
- Το πρωτότυπο είναι:

```
int fflush(FILE *fp)
```

  - Επιστρέφει 0 σε επιτυχία και EOF αλλιώς
- Ενώ η συνάρτηση fflushall

```
int fflush(void)
```

  - κάνει το ίδιο σε όλα τα ανοικτά αρχεία

Σημείωση:
- Κάποιοι (αλλά όχι) όλοι οι μεταγλωττιστές έχουν υλοποιήσει την fflush ώστε να δουλεύει και για ρεύματα εισόδου, καθαρίζοντας τον buffer εντελώς (π.χ. fflush(stdin))
- Γενικά δεν συνίσταται η χρήση της με ρεύματα εισόδου

B. Ασκήσεις
1.1. Το mode “a”

Κατασκευάστε ένα προσωρινό αρχείο κειμένου με κάποιο τυχαίο κείμενο και εξασκηθείτε στον τρόπο ανοίγματος “a” (append):

1. Ελέγξτε που βρίσκεται ο δρομέας κατά το άνοιγμα του αρχείου
2. Γράψτε 4 χαρακτήρες στο αρχείο
3. Γυρίστε το δρομέα στην αρχή
4. Τυπώστε στην οθόνη το αρχείο

Τι παρατηρείτε; Επαναλάβετε με το mode “a+”

Β. Ασκήσεις
1.2. Το mode “r”

Κατασκευάστε ένα προσωρινό αρχείο κειμένου με κάποιο τυχαίο κείμενο και εξασκηθείτε στον τρόπο ανοίγματος “r” (read):

1. Ελέγξτε που βρίσκεται ο δρομέας κατά το άνοιγμα του αρχείου
2. Τυπώστε το αρχείο στην οθόνη
3. Γυρίστε το δρομέα στην αρχή
4. Γράψτε 4 χαρακτήρες στο αρχείο
5. Τυπώστε το αρχείο στην οθόνη

Τι παρατηρείτε; Επαναλάβετε με το mode “r+”

Β. Ασκήσεις
1.3. Το mode “w”

Κατασκευάστε ένα προσωρινό αρχείο κειμένου με κάποιο τυχαίο κείμενο και εξασκηθείτε στον τρόπο ανοίγματος “w” (write):

1. Ελέγξτε που βρίσκεται ο δρομέας κατά το άνοιγμα του αρχείου
2. Γράψτε 4 χαρακτήρες στο αρχείο
3. Γυρίστε το δρομέα στην αρχή
4. Τυπώστε το αρχείο στην οθόνη

Τι παρατηρείτε; Επαναλάβετε με το mode “w+”

Β. Ασκήσεις
2. Αντιγραφή αρχείου

Γράψτε ένα πρόγραμμα το οποίο:

1. Να ζητάει από το χρήστη ένα όνομα αρχείου
2. Να αντιγράφει τα περιεχόμενά του σε ένα νέο αρχείο (του οποίου το όνομα επίσης να ζητάει από το χρήστη)

Β. Ασκήσεις
3.1. Άνοιγμα ενός αρχείου δεδομένων

Δεδομένης μίας δομής «record» με στοιχεία:

- Όνομα
- Επώνυμο
- Ηλικία
- Βαθμός

Γράψτε μία συνάρτηση (open) η οποία:

- Θα δέχεται ως όρισμα το όνομα ενός αρχείου
- Θα ανοίγει το αρχείο και θα μετράει πόσες εγγραφές περιέχει
  - Αν δεν υπάρχει το αρχείο, θα το δημιουργεί
- Θα επιστρέφει
  - TRUE/FALSE (σταθερές που θα ορίσετε) ανάλογα με το αν ολοκληρώθηκε με επιτυχία το άνοιγμα του αρχείου
  - Τον δείκτη στη δομή FILE
  - Το πλήθος των εγγραφών του αρχείου

Η main να έχει ένα μενού επιλογών, εκ των οποίων η πρώτη να είναι το «Άνοιγμα»

Β. Ασκήσεις
3.2. Κλείσιμο αρχείου

Γράψτε μία συνάρτηση (close) η οποία:
- Δέχεται ως όρισμα τον δείκτη στο αρχείο
- Θα κλείνει το αρχείο
- Θα επιστρέφει TRUE/FALSE ανάλογα με το αν πήγαν όλα καλά.

Επεκτείνετε κατάλληλα το μενού με την επιλογή «Έξοδος»

Β. Ασκήσεις
3.3. Νέα Εγγραφή

Γράψτε μία συνάρτηση (add) η οποία:
- Θα κατασκευάζει μία νέα εγγραφή (με είσοδο του χρήστη)
- Θα την εισάγει στο τέλος του αρχείου

Επεκτείνετε κατάλληλα το πρόγραμμα με μία επιλογή «Εισαγωγή»

Β. Ασκήσεις
3.4. Διάβασμα και εκτύπωση μιας εγγραφής

Γράψτε μία συνάρτηση (read) η οποία:
- Δέχεται ως όρισμα τον δείκτη στο αρχείο, το πλήθος των εγγραφών, έναν ακέραιο αριθμό και ένα στιγμιότυπο της δομής record
- Θα αποθηκεύει το record που αντιστοιχεί στον ακέραιο στο όρισμα (δομή record)
- Θα επιστρέφει TRUE/FALSE ανάλογα με το αν πήγαν όλα καλά.

Γράψτε μία συνάρτηση (print) η οποία
- Θα παίρνει ως όρισμα ένα record και θα το τυπώνει στην οθόνη

Επεκτείνετε τη main με μία επιλογή «Διάβασμα» που θα εκτυπώνει μία εγγραφή με κατάλληλη κλήση των παραπάνω συναρτήσεων.

Β. Ασκήσεις
3.5. Εκτύπωση όλων των εγγραφών

Γράψτε μία συνάρτηση (print_all) η οποία:
- Θα τυπώνει όλες τις εγγραφές του αρχείου

Επεκτείνετε κατάλληλα το πρόγραμμα με μία επιλογή «Εκτύπωση όλων» (print_all)

Β. Ασκήσεις
3.6. Τροποποιήση

Γράψτε μία συνάρτηση (modify) η οποία:

- Θα παίρνει ως ορίσματα το δείκτη αρχείου, το πλήθος των εγγραφών, την εγγραφή που θέλει να τροποποιήσει.
- Θα τροποποιεί την εγγραφή (με είσοδο του χρήστη)
- Θα εκτυπώνει την τροποποιημένη εγγραφή στην οθόνη
- Θα ενημερώνει το αρχείο

Επεκτείνετε κατάλληλα το πρόγραμμα με μία επιλογή «Τροποποίηση»

Β. Ασκήσεις
3.7. Επεκτάσεις…

Προβληματιστείτε για το πως θα μπορούσε να ενημερωθεί το πρόγραμμα ώστε να υποστηρίζει και τη διαγραφή μίας εγγραφής.