Δομημένος προγραμματισμός

https://commons.wikimedia.org

Χειρισμός αρχείων

Κώστας Βασιλάκης, kostas@cs.hmu.gr

Περίγραμμα της ύλης που θα καλυφθεί

- Γιατί αρχεία?
- Ροές δεδομένων και αρχεία
- Οι συναρτήσεις fopen() & fclose()
- Οι συναρτήσεις fgetc() & fputc()
- Οι συναρτήσεις fgets() & fputs()
- Οι συναρτήσεις fprintf() & fscanf()
- Οι συναρτήσεις fwrite() & fread()
- Η συνάρτηση fseek()
- Άλλες συναρτήσεις για αρχεία

https://pxhere.com/

https://commons.wikimedia.org

Γιατί αρχεία;

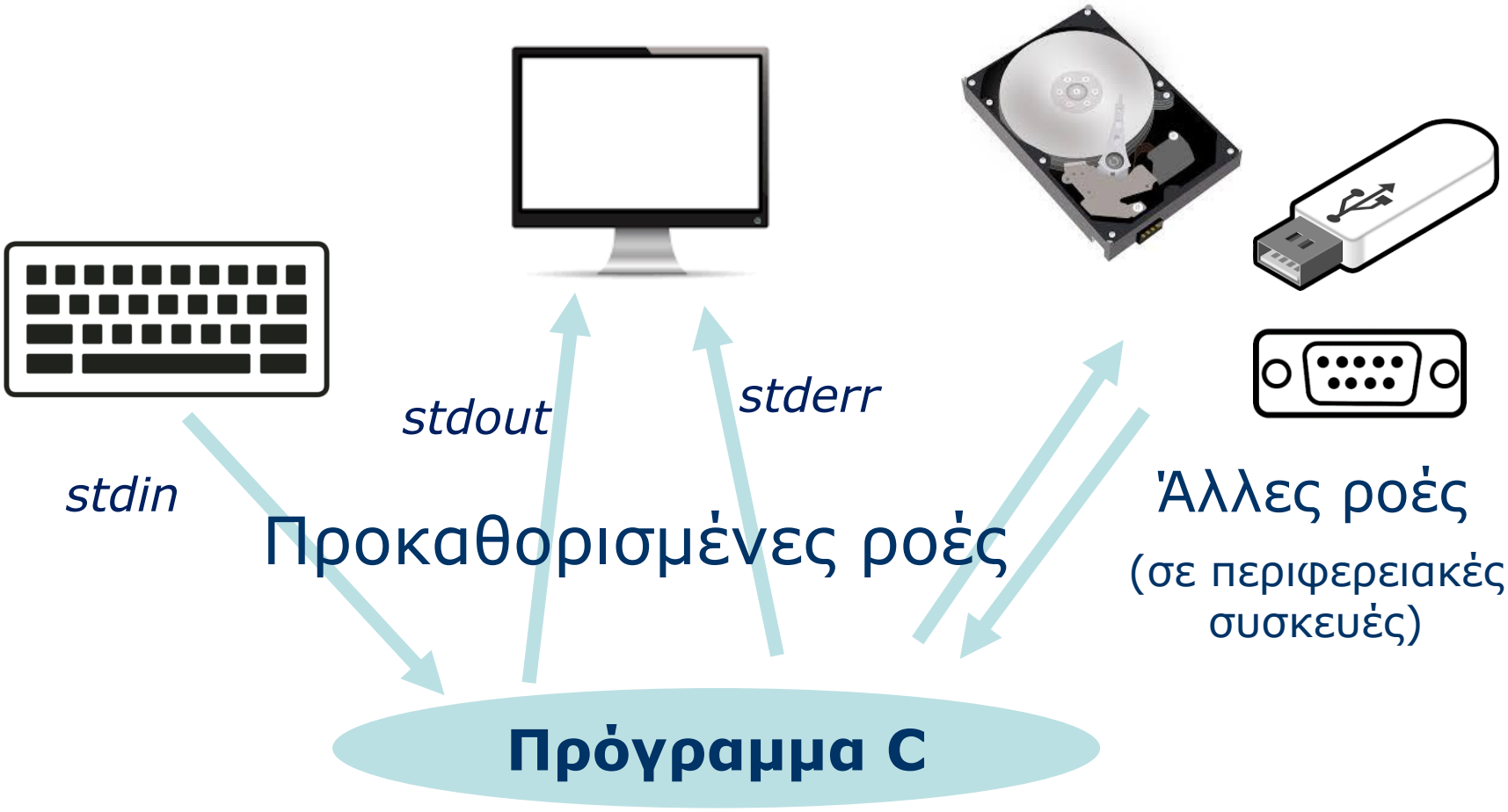
- Τα δεδομένα που χρησιμοποιούνται σε ένα πρόγραμμα βρίσκονται στην κύρια μνήμη του υπολογιστή όσο εκτελείται το πρόγραμμα.
- Μετά τη εκτέλεση του προγράμματος τα δεδομένα αυτά χάνονται.
- Για να διατηρήσουμε δεδομένα και μετά τη εκτέλεση του προγράμματος θα πρέπει να μπορούμε να τα αποθηκεύσουμε σε αρχεία, στη περιφερειακή μνήμη του υπολογιστή.
- Στη C υπάρχουν συναρτήσεις εισόδου-εξόδου που μας δίνουν την δυνατότητα να αποθηκεύουμε δεδομένα σε αρχεία και να τα ανακτούμε από αυτά.
- Αυτές οι συναρτήσεις δηλώνονται στο αρχείο *<stdio.h>*.
- Συνεπώς, η C θεωρεί τα αρχεία σαν μονάδες εισόδου-εξόδου.

Streams (ροές δεδομένων)

- Μια *ροή δεδομένων* (stream) είναι μια «αφηρημένη» έννοια που συνδέει ένα C πρόγραμμα με τις περιφερειακές μονάδες του υπολογιστή (πληκτρολόγιο, οθόνη, δίσκοι κλπ.).
  - *Παρεμβάλλεται μεταξύ του προγράμματος και της μονάδας/συσκευής.*
  - *Πρόκειται για ένα εικονικό σύστημα προσπέλασης μονάδων*.
- Με τις ροές δεδομένων η διαχείριση των μονάδων του υπολογιστή γίνεται με ομοιόμορφο τρόπο, ανεξάρτητα αν η μονάδα είναι:
  - *αρχείο στο δίσκο,*
  - *οθόνη (standard output, stdout),*
  - *πληκτρολόγιο (standard in, stdin),*
  - *σειριακή θύρα κλπ.*
- Κάθε C πρόγραμμα χρησιμοποιεί 3 προκαθορισμένες ροές:

  stdin (πληκτρολόγιο), *stdout* και *stderr* (οθόνη).

Οι ροές δεδομένων στη C



Ροές και αρχεία

Ο χειρισμός αρχείου αφορά κυρίως σε 2 ενέργειες:

ανάγνωση και *εγγραφή* Bytes

Υπάρχουν 2 ειδών ροές αρχείων:

- Ροές αρχείων *κειμένου* (text)

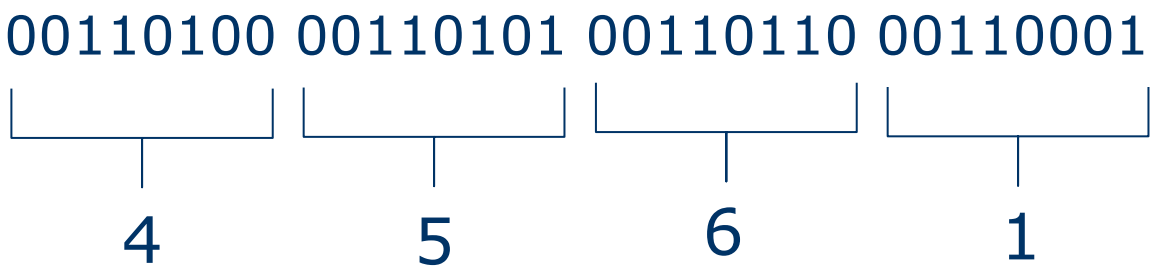
  Αναπαράσταση δεδομένων με χαρακτήρες (πχ με ASCII κωδικοποίηση). Ο χαρακτήρας νέας γραμμής μετατρέπεται από τη ροή διαφορετικά, ανάλογα με το λειτουργικό σύστημα που εκτελείται το πρόγραμμα (στα Windows σαν \n\r, στο Linux σαν \n).

- Ροές *δυαδικών* (binary) αρχείων

  Αναπαράσταση κάθε τύπου δεδομένων. Ακολουθίες από Bytes. Ένα δυαδικό αρχείο συνήθως δεν είναι αναγνώσιμο. Δεν υπάρχουν μετατροπές χαρακτήρων.

Αρχεία κειμένου – δυαδικά / τρόπος αποθήκευσης

- Πως θα αποθηκευτεί ο ακέραιος *4561* σε αρχείο κειμένου;



Αναπαραστάσεις των χαρακτήρων 4, 5, 6 και 1 (ascii, iso κλπ.)

- Πως θα αποθηκευτεί ο ίδιος ακέραιος σε δυαδικό αρχείο;

0001000111010001

Πρόκειται για τον αριθμό 4561 σε δυαδική μορφή!

- Στα αρχεία κειμένου, το τέλος (μπορεί να) σηματοδοτείται με κάποιο ειδικό χαρακτήρα (στα Windows είναι το <*ctrl*>*Z*).

Η προσπέλαση στα αρχεία

- Αρχείο: μια ακολουθία από Bytes σε περιφερειακή μνήμη.
- Μια ροή:
  - *συσχετίζεται με κάποιο αρχείο με τη διαδικασία «ανοίγματος» του αρχείου – συνάρτηση: fopen().*
  - *από-συσχετίζεται αρχείο από τη ροή με τη διαδικασία «κλεισίματος» - συνάρτηση: fclose().*
- Στη C η προσπέλαση σε αρχείο μπορεί να χαρακτηριστεί ως:
  - *σειριακή προσπέλαση (για έχω προσπέλαση στο n-οστο Byte θα πρέπει πρώτα να προσπελάσω τα προηγούμενα Bytes),*
  - *τυχαία προσπέλαση (μεταφορά απ' ευθείας σε συγκεκριμένο Byte).*
- *Τρέχουσα θέση*: η θέση στο αρχείο όπου θα ξεκινήσει η επόμενη προσπέλαση.

Η δομή FILE

- Οι αναφορές στα αρχεία γίνονται με τη βοήθεια ενός *δείκτη σε δομή τύπου FILE*.
- Η δήλωση της δομής τύπου *FILE* (βρίσκεται στο <stdio.h>). Είναι μια δομή που περιγράφει ένα αρχείο και χρησιμοποιείται για τον χειρισμό των αρχείων.
- Ένας δείκτης δομής τύπου FILE ορίζεται στη C ως εξής:

```
FILE *fp; //δημιουργεί ένα δείκτη fp τύπου FILE
```

- Ένας τέτοιος δείκτης μας δίνει τη δυνατότητα να συσχετίσουμε ροές με συγκεκριμένα αρχεία και στη συνέχεια να τα χειριστούμε με κατάλληλες συναρτήσεις.
- Πριν εκτελέσουμε οποιαδήποτε λειτουργία σε κάποιο αρχείο, θα *πρέπει πρώτα* να το ανοίξουμε (file open).
- Όταν τελειώσει η λειτουργία, *πρέπει να κλείσουμε* το αρχείο (file close).

Άνοιγμα αρχείου: η συνάρτηση fopen()

- Το πρότυπο της συνάρτησης είναι ένας δείκτης σε δομή τύπου FILE:

*FILE *fopen (char *fname, char *mode)*

όπου,

- *fname: το όνομα του αρχείου (απαιτείται η πλήρης διαδρομή του αρχείου, αν αυτό βρίσκεται σε διαφορετικό κατάλογο από αυτόν του προγράμματος που εκτελείται).*
- *mode: για ποιο λόγο ανοίγουμε το αρχείο (τι ενέργειες θέλουμε να κάνουμε).*

- Η συνάρτηση *fopen()* επιστρέφει:
  - *ένα δείκτη δομής τύπου FILE, αν είναι επιτυχής εκτέλεση της,*
  - *τη τιμή NULL (η NULL έχει μια σταθερή τιμή που ορίζεται στο <stdio.h>) σε περίπτωση αποτυχίας.*

- Παράδειγμα:

```
fp=fopen("toarxeiomou.txt","r"); //ανοίγει για ανάγνωση
```

Πρέπει πάντα να ελέγχουμε τη επιστροφή της fopen()

Επιτρεπτές ενέργειες -mode της fopen()

Για αρχεία κειμένου:

mode	Επιτρεπτή ενέργεια
r	Άνοιγμα αρχείου κειμένου για ανάγνωση δεδομένων. Το αρχείο πρέπει να υπάρχει. *Τρέχουσα θέση: αρχή*.
w	Δημιουργία αρχείου κειμένου για εγγραφή. Αν το αρχείο υπάρχει τα δεδομένα διαγράφονται. Αν δεν υπάρχει, δημιουργείται. *Τρέχουσα θέση: αρχή.*
a	Προσθήκη δεδομένων σε αρχείο κειμένου. Αν το αρχείο δεν υπάρχει δημιουργείται. *Τρέχουσα θέση: στο τέλος του αρχείου*.
r+	Άνοιγμα αρχείου κειμένου για ανάγνωση/εγγραφή δεδομένων. Το αρχείο πρέπει να υπάρχει. *Τρέχουσα θέση: αρχή*.
w+	Δημιουργία αρχείου κειμένου για ανάγνωση/εγγραφή δεδομένων. Αν το αρχείο υπάρχει τα δεδομένα διαγράφονται. Αν δεν υπάρχει, δημιουργείται. *Τρέχουσα θέση: αρχή*.
a+	Προσθήκη/ανάγνωση δεδομένων. Αν το αρχείο δεν υπάρχει, δημιουργείται. *Τρέχουσα θέση: τέλος αρχείου*.

Για τα δυαδικά αρχεία προσθέτουμε τον χαρακτήρα b (π.χ. wb, rb+)

fopen() – παραδείγματα χρήσης

```
FILE *fptr;
.....
fptr=fopen("students.txt", "r");
```

Άνοιγμα του αρχείου students.txt στον τρέχοντα κατάλογο για ανάγνωση (mode: “r”).

Άνοιγμα του αρχείου students.txt στον κατάλογο C:\course1 για εγγραφή (mode: “w”).

```
FILE *fptr;
fptr=fopen("C:\\course1\\students.txt","w");
```

Σε compilers των MS-Windows ο χαρακτήρας \ δίδεται σαν \\:
Δηλαδή, αντί για *C:\course* γράφουμε *C:\\course*.
Αυτό διότι ο χαρακτήρας \ έχει ειδική σημασία στη C.

fopen() – παραδείγματα χρήσης με έλεγχο επιτυχίας

```
FILE *fptr;
fptr=fopen("students.txt","w");
if (fptr==NULL) {
      printf("Error!");
      exit(1);   // χρειάζεται η stdlib.h
  }
```

Άνοιγμα του αρχείου students.txt στον τρέχοντα κατάλογο για εγγραφή με έλεγχο επιτυχίας.

Η κλήση της *exit()* υποδεικνύει στο λειτουργικό σύστημα ότι τελείωσε η εκτέλεση του προγράμματος.

- *exit(0):* επιτυχής τερματισμός
- *exit (1):* ανεπιτυχής τερματισμός.

Για τη χρήση της συνάρτησης *exit() χρειαζόμαστε τη* <stdlib.h>.

```
FILE *fptr;
If ((fptr=fopen("students.txt","w"))==NULL) {
      printf("Error!");
      exit(1);
  }
```

Ακόμα καλύτερα!

Η συνάρτηση fclose()

- Το πρότυπο της συνάρτησης στο <stdio.h>είναι:

*int fclose (FILE *fp);*

- Παράδειγμα χρήσης:

fclose (fptr);

Κλείνει το ανοικτό αρχείο που είναι συσχετισμένο με τον δείκτη *fptr* (ροή).

- Ο δείκτης του σχετικού αρχείου πρέπει να είναι έγκυρος. Δηλαδή, το αρχείο να έχει ανοίξει επιτυχώς με την *fopen().*
- Επιστρέφει:
  - *0 αν το κλείσιμο του αρχείου είναι επιτυχές,*
  - *την τιμή EOF (End Of File) αν το κλείσιμο είναι ανεπιτυχές.*
- Η ειδική τιμή *EOF* ορίζεται στο <stdio.h> και χρησιμοποιείται σαν ένδειξη «τέλους του αρχείου» ή ότι έγινε κάποιο λάθος. Συνήθως έχει την τιμή *-1*.

Οι συναρτήσεις fgetc()/getc() και fputc()/putc()

- Χρησιμοποιούνται για να διαβάσουμε ή να γράψουμε Bytes σε αρχεία.
- Τα πρότυπα:

```
int fgetc(FILE *fp) ;
int fputc(int ch,  FILE *fp);
```

- Τρόπος χρήσης:

Οι *getc()* και *putc()* μπορούν να χρησιμοποιηθούν επίσης αντί για τις *fgetc()* και *fputc()* αντίστοιχα.

```
i=fgetc(fp);
```

Διαβάζει το επόμενο Byte από το αρχείο της ροής *fp* σαν *μη προσημασμένο ακέραιο* και το επιστρέφει σαν ακέραιο, αν είναι επιτυχές το διάβασμα. Διαφορετικά επιστρέφει την τιμή *EOF* (σφάλμα ή τέλος αρχείου).

```
fputc(chr, fp);
```

Γράφει τον χαρακτήρα της ακεραίας μεταβλητής *chr* στο αρχείο που σχετίζεται με την ροή *fp*. Σε περίπτωση επιτυχίας επιστρέφει τον χαρακτήρα (σε ακέραια μεταβλητή), διαφορετικά *EOF*.

Πρόγραμμα με τη συνάρτηση fputc()

Γράφει n χαρακτήρες * στο αρχείο WriteStars.txt.
Ο αριθμός των χαρακτήρων n δίνεται από τον χρήστη

```
#include <stdio.h>
#include <stdlib.h>
main() {
    FILE *fp;
    int i,n;
    fp=(fopen("WriteStars.txt","w")); //άνοιγμα αρχείου για εγγραφή
    if(fp==NULL){
        printf("Error opening file!");
        exit(1); // χρειάζεται η <stdlib.h>
     }
     printf("Enter n: ");
     scanf("%d",&n);
     for(i=0;i<n;++i)
            fputc('*',fp); //γράφονται n *'s στο αρχείο
     fclose(fp);
}
```

Ο έλεγχος μπορεί να γίνει και έτσι:

```
if ((fp=fopen("WriteStars.txt","w"))==NULL){
    printf("Error opening file!");
    exit(1);
}
```

Πρόγραμμα με τη συνάρτηση fgetc()

Διαβάζει τους χαρακτήρες που υπάρχουν στο αρχείο WriteStars.txt.

```
#include <stdio.h>
#include <stdlib.h>
main() {
   FILE *fp;
   int i;
   if ((fp=fopen("WriteStars.txt", "r"))==NULL) {
      printf("Error: cannot open the file!");
      exit(1);
    }
    for (;;) {
          i=fgetc(fp); //Διάβασμα ενός χαρακτήρα
          if (i==EOF) break; //Αν τέλος, έξοδος από το βρόγχο
          putchar(i);  //Εμφάνιση του χαρακτήρα στη οθόνη
     }
    fclose(fp);
}
```

Ατέρμων βρόγχος. Το ίδιο αποτέλεσμα και με while (1).

Πρόγραμμα με τις συναρτήσεις getc() και putc()

```
if ((fp=fopen("W-R-Stars.txt", "w")) ==NULL) {
     printf("Error: can't create file!");
     exit(1);
  }
  printf("Enter n: "); scanf("%d",&n);
  for(i=0;i<n;++i)
          putc('*',fp); //γράφονται n * στο αρχείο
  fclose(fp); //κλείσιμο του αρχείου
  if ((fp=fopen("W-R-Stars.txt", "r"))==NULL) {
     printf("Error: can't read the file!");
    exit(1);
  }
  while (1) {
        if ((i=getc(fp)) == EOF) break;
        putchar(i);
   }
  fclose(fp);
```

Γράφει n χαρακτήρες * στο αρχείο W-R-Stars.txt. Στη συνέχεια τους διαβάζει και τους εμφανίζει στη οθόνη.

Προσέξτε εδώ! Διαφορετικό διάβασμα. Δείτε και αυτό:
while ((i=getc(fp))!=EOF) putchar(i);

Επίσης, θα μπορούσαμε να χρησιμοποιήσουμε τη συνάρτηση *rewind(fp)* και να μην ανοίξουμε και κλείσουμε το αρχείο 2 φορές (πως?).

Η συνάρτηση fputs() για αρχεία κειμένου

fputs()

- Πρότυπο: *int fputs (char *str, FILE *fp);*
- Εγγράφει τη συμβολοσειρά **str* στο αρχείο με το οποίο συσχετίζεται ο δείκτης *fp*.
- Αν γίνει *επιτυχής* εγγραφή επιστρέφεται μια *μη-αρνητική* τιμή, διαφορετικά (*μη επιτυχής* εγγραφή) επιστρέφεται η τιμή *EOF*.
- Αντίθετα με την *puts()* δεν γράφει το χαρακτήρα νέας γραμμής στο τέλος.
- Παράδειγμα χρήσης:

fputs(str,fp);

Η συνάρτηση fgets() για αρχεία κειμένου

fgets()

- Πρότυπο: *char *fgets (char *str, int num, FILE *fp);*
- Διαβάζει *num-1* χαρακτήρες από το αρχείο με το οποίο συσχετίζεται ο δείκτης *fp* και τους καταχωρεί στη συμβολοσειρά *str*.
- Επιστρέφει τον δείκτη *str* αν είναι *επιτυχής* και τη τιμή *NULL* διαφορετικά (*αποτυχία*).
- Αντίθετα με την *gets()* διατηρεί τον χαρακτήρα νέα γραμμής.
- Παράδειγμα χρήσης:

fgets (str, 80, fp);

Παραδείγματα χρήσης των fgets() και fputs()

```
char str[100];
……..
fgets(str, sizeof(str), stdin);
```

Διαβάζει *99 Bytes* από τη προκαθορισμένη ροή *standard input* (stdin). Ίδιο με το *gets(s)*;

```
char str[100];
fp = fopen("datafile.dat", "r");
fgets(str, 100, fp);
fclose(fp);
```

Διαβάζει *99 Bytes* από το αρχείο της ροής *fp* (datafile.dat).

```
fp = fopen("datafile.dat", "w+");
while(fgets(str, sizeof(str), stdin) != NULL) {
        fputs(str, fp);
}
```

Διαβάζει, με έλεγχο, *99 Bytes* από το *stdin* και τους γράφει στο αρχείο της ροής *fp* (datafile.dat)

Πρόγραμμα με τις συναρτήσεις fputs() & fgets()

```
FILE *fp;
char str[80];
if ((fp=fopen("result.txt ", "w+"))==NULL) {
      printf("Error opening file!"); getchar();
      exit(1);
}
fputs("Hello! These are the result of my C program. See you later", fp);
fclose(fp);
if ((fp=fopen("result.txt", "r"))==NULL) {
      printf("Error opening file!"); getchar();
      exit(1);
}
if ( fgets (str, 23, fp)!=NULL )
                              puts(str);
fclose(fp);
```

Γράφει μια συμβολοσειρά στο αρχείο *result.txt*. Στη συνέχεια, διαβάζει τους 22 πρώτους χαρακτήρες από το αρχείο

Η συμβολοσειρά

Διαβάζονται οι 22 πρώτοι χαρακτήρες.

Η συνάρτηση fprintf() για αρχεία κειμένου

fprintf()

- Πρότυπο: *int fprintf(FILE *fp, char *format, ...);*
- Σύνταξη ίδια με αυτή της *printf().*
- Η *printf()* στέλνει τα δεδομένα στη οθόνη (*sdtout*), ενώ η *fprinf()* τα γράφει στο αρχείο της ροής του δείκτη *fp*.
- Αν γίνει *επιτυχής* εγγραφή, επιστρέφεται ο *αριθμός* των χαρακτήρων που γράφτηκαν, διαφορετικά (*αποτυχία*) επιστρέφεται μια *αρνητική τιμή*.
- Παράδειγμα χρήσης:

fprintf(fp, "%d %f ", a,b);

Η συνάρτηση fscanf() για αρχεία κειμένου.

fscanf()

- Πρότυπο: *int fscanf(FILE *fp, const char *format, ...);*
- Σύνταξη ίδια με αυτή της *scanf().*
- Διαβάζει τα δεδομένα από αρχείο, αντί να τα διαβάζει από το πληκτρολόγιο (*stdin*).
- Επιστρέφει τον *αριθμό* των δεδομένων που διαβάστηκαν, ενώ σε περίπτωση *σφάλματος* επιστρέφει η τιμή *EOF*.
- Παράδειγμα χρήσης:

fscanf(fp, "%d ", &a);

Πρόγραμμα με τη συνάρτηση fprintf()

```
char name[15];
int i,n;
float mark;
printf("Enter number of students: ");
scanf("%d",&n);
FILE *fptr;
if ((fptr=fopen("marks.txt", "w"))==NULL) {
      printf("Error!"); getchar();
      exit(1);
 }
for(i=0;i<n;++i) {
      printf("Student%d\nEnter name: ", i+1);
      scanf("%s", name);
      printf("Enter mark: ");
      scanf("%f", &mark);
      fprintf(fptr,"%s %3.1f \n", name, mark);
}
fclose(fptr);
```

Διαβάζει τα ονόματα και τους βαθμούς φοιτητών από το *stdin* και τα αποθηκεύει στο αρχείο *marks.txt*

Πρόγραμμα με τη συνάρτηση fscanf()

```
char name[15];
int i,n;
float mark;
printf("Enter number of students: ");
scanf("%d", &n);
FILE *fptr;
if ((fptr==fopen("marks.txt", "r"))==NULL) {
      printf("Error!"); getchar();
      exit(1);
   }
for(i=0;i<n;++i) {
      fscanf(fptr, "%s %f ",&name, &mark);
      printf("%s has mark %3.1f\n", name, mark);
 }
 fclose(fptr);
```

Διαβάζει τα ονόματα και τους βαθμούς φοιτητών από το αρχείο *marks.txt* και τα εμφανίζει στην οθόνη.

Δεν ελέγχεται αν γίνεται σωστά το διάβασμα από την *fscanf()*.

Πως θα το κάναμε?

Οι συναρτήσεις fwrite() & fread() – δυαδικά αρχεία

- Συναρτήσεις για εγγραφή και διάβασμα δεδομένων σε/από ένα δυαδικό αρχείο.
- Πολλές φορές είναι επιθυμητό να χρησιμοποιούμε δυαδικά αρχεία αντί αρχείων κειμένου διότι:
  - *δεν γίνονται μετατροπές στα δεδομένα και*
  - *έχουν μικρότερο μέγεθος.*
- Τα πρότυπα των συναρτήσεων:

```
int  fread(void *buffer,  size_t  size, size_t  num, FILE  *fp);
ınt  fwrite(void *buffer, size_t  size, size_t  num, FILE  *fp);
```

- Χρησιμοποιούν τον δείκτη *buffer* πού είναι τύπου *void* και άρα μπορεί να δείχνει σε οποιοδήποτε τύπο δεδομένων.
- Ο τύπος *size_t* ορίζεται στο αρχείο <*stdio.h*> και είναι ισοδύναμος με τον τύπο *unsigned long* (διαφέρει από σύστημα σε σύστημα).

Οι συναρτήσεις fwrite() & fread() - περισσότερα

```
int  fread(void *buffer,  size_t  size, size_t  num, FILE  *fp);
int  fwrite(void *buffer, size_t  size, size_t  num, FILE  *fp);
```

- Στη μνήμη που δείχνει ο *buffer* αποθηκεύονται προσωρινά τα δεδομένα που θα διαβαστούν ή θα εγγραφούν από τις *fread()* και *fwrite()* αντίστοιχα.
- Η παράμετρος *num* προσδιορίζει το πλήθος των στοιχείων που θα διαβαστούν/αποθηκευτούν.
- Η παράμετρος *size* ορίζει το μέγεθος κάθε στοιχείου (που είναι πολύ πιθανό να είναι διαφορετικό από σύστημα σε σύστημα).
- Και εδώ η παράμετρος *fp* (δείκτης) *τύπου FILE,* πρέπει να δείχνει στο σχετικό αρχείο.
- Οι συναρτήσεις *fwrite() & fread()* επιστρέφουν:
  - το *num* (επιτυχία),
  - κάποιον άλλο αριθμό (αποτυχία).
- Στο fread(), η επιστροφή κάπου άλλου αριθμού (≠num) μπορεί να σημαίνει και τέλος αρχείου.

Πρόγραμμα με τις συναρτήσεις fwrite() & fread()

```
FILE *fp = NULL;
int x[10] = {1,2,3,4,5,6,5,6,-10,11};
int i,result[10];
if ((fp=fopen("tmp.b", "wb+")) != NULL) {
        fwrite(x, sizeof(int), 10, fp);
        fclose(fp);
        if ((fp=fopen("tmp.b", "rb+")) == NULL)
                printf("Error opening temp.b-2nd time");
        else {
                fread(result, sizeof(int), 10 , fp);
                printf("\nResult:\n");
                for (i=0;i<10;i++)
                        printf("%d\n",result[i]);
        }
}
else
        printf("Error opening temp.b-1st time");
 fclose(fp);
```

Με έλεγχο

Χωρίς έλεγχο

Με έλεγχο

Χωρίς έλεγχο

Γράφει ένα πίνακα 10 ακεραίων σε αρχείο. Στη συνέχεια διαβάζει το αρχείο και δημιουργεί ένα νέο πίνακα που τον εμφανίζει στη οθόνη.

Το προηγούμενο πρόγραμμα με όλους τους ελέγχους

```
FILE *fp = NULL;
  int i, x[10] = {1,2,3,4,5,6,5,6,-10,11}, result[10];
  if ((fp=fopen("tmp.b", "wb+")) != NULL) {
      if ((fwrite(x, sizeof(x), 1, fp)) != 1)
            printf ("Cannot open tmp.b for writing\n");
       else {
            fclose(fp);
            if ((fp=fopen("tmp.b", "rb+")) == NULL)
                  printf("Error opening temp.b -2nd time\n");
            else
               if (fread(result, sizeof(x), 1 , fp) !=1)
                     printf ("Cannot open tmp.b for reading\n");
               else {  printf("\nResult:\n");
                     for (i=0;i<10;i++)  printf("%d\n",result[i]);
            }
     }
  else  printf("Error opening  tmp.b-1st time\n");
  fclose(fp);
```

Γράφει ένα πίνακα 10 ακεραίων σε αρχείο. Στη συνέχεια διαβάζει το αρχείο και δημιουργεί ένα νέο πίνακα που τον εμφανίζει στη οθόνη.

Εδώ οι έλεγχοι επιτυχίας για κάθε συνάρτηση διαχείρισης αρχείων γίνεται χωρίς την χρήση της συνάρτησης *exit()*.

Τι διαφορά έχουν οι *fwrite()* και *fread()* από το προηγούμενο παράδειγμα?

Τυχαία προσπέλαση – η συνάρτηση fseek()

- Στα παραδείγματα μέχρι τώρα η προσπέλαση των αρχείων γινόταν σειριακά.
- Με τη συνάρτηση *fseek()* μπορούμε να πάμε σε συγκεκριμένο σημείο του αρχείου για κάνουμε τις ενέργειες που θέλουμε.

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

- Η παράμετρος *offset* καθορίζει την απόσταση από την αρχή ή το τέλος (αρνητική τιμή) σε Bytes.
- Η παράμετρος *origin* καθορίζει το σημείο εκκίνησης και έχει τις τιμές (μακροεντολές):
  - *SEEK_SET (από τη αρχή του αρχείου),*
  - *SEEK_CUR (από την τρέχουσα θέση) και*
  - *SEEK_END (από το τέλος του αρχείου.*
- Επιστρέφει την τιμή *0*, αν έχουμε επιτυχή μετακίνηση του δείκτη, διαφορετικά επιστρέφει μια *μη-μηδενική τιμή*.

Παραδείγματα χρήσης

```
fseek(fp, 0, SEEK_SET);
```
← Πάει στη αρχή του αρχείου

```
fseek(fp, 100, SEEK_SET);
```
← Πάει από τη αρχή στο 100^{στο} Byte.

```
fseek(fp, -30, SEEK_CUR);
```
← Πάει πίσω 30 Bytes από τη τρέχουσα θέση

```
fseek(fp, -10, SEEK_END);
```
← Πάει πίσω 10 Bytes από το τέλος

Πρόγραμμα με τη συνάρτηση fseek()

```
FILE *fp = NULL;
  float x[10] = {1.5,2.3,3.6,4.7,5.6,6.2,5.3,6.8,1.7,1.1}, V;
  int i,L;
  if ((fp=fopen("tmpd.b", "wb+")) != NULL) {
        fwrite(x, sizeof(x), 1, fp); fclose(fp);
        printf("Which element? "); scanf("%d",&L);
         if (fseek(fp, (L-1)*sizeof(float), SEEK_SET)){
                printf("Error on seeking... ");
                exit(1);
          }
          fread(&V, sizeof(float), 1 , fp);
          printf("\n Element %d is %.2f", L, V);
  }
  else
           printf("Error opening tempd.b-1st time");
  fclose(fp);
```

Γράφει ένα πίνακα 10 floats σε αρχείο. Στη συνέχεια διαβάζει και εμφανίζει στη οθόνη το στοιχείο που θέλει ο χρήστης.

Η τρέχουσα θέση στην αρχή του L στοιχείου

Η συνάρτηση feof()

- Όταν οι συναρτήσεις *fgetc()* και *fscanf()* επιστρέφουν τη τιμή *EOF*, δεν γνωρίζουμε αν έχουμε κάποιο σφάλμα ή αν φτάσαμε στο τέλος του αρχείου.
- Το πρότυπο της συνάρτησης:

*int feof (FILE *fp);*

```
if ((fptr=fopen("marks.txt", "r"))==NULL) {
      printf("Error!"); getchar();
      exit(1);
   }
printf("\nReadning from file: marks.txt:\n");
while (!feof(fptr)){
   fscanf(fptr,"%s %f ",&name,&mark);
   printf("%s has mark %3.1f\n", name, mark);
 }
 fclose(fptr);
```

Η συνάρτηση *feof()* επιστρέφει:

- *0* αν δεν έχουμε φτάσει στο τέλος του αρχείου και
- *μη μηδενική* τιμή αν έχουμε φτάσει στο τέλος.

Η συνάρτηση ferror()

- Χρησιμοποιείται για να διαπιστώσουμε αν υπάρχει κάποιο σφάλμα κατά τη λειτουργία μιας συνάρτησης που προσπαθεί να αποκτήσει πρόσβαση σε κάποιο αρχείο.
- Το πρότυπο της συνάρτησης:

*int ferror (FILE *fp);*

```
if ((fptr=fopen("marks.txt", "r"))==NULL) {
        printf("Error!"); getchar();  exit(1);
 }
while (!feof(fptr)) {
     fscanf(fptr,"%s %f ",&name, &mark);
     if (ferror(fptr)) {
            printf("\n***Error: fscanf\n");
            break;
      }
      printf("%s has mark %3.1f\n", name, mark);
}
```

Η συνάρτηση *ferror()* επιστρέφει:

- *μη-μηδενική* τιμή αν υπάρχει σφάλμα,
- διαφορετικά επιστρέφει *0*.

Άλλες χρήσιμες συναρτήσεις

rewind()

- Πρότυπο: *void rewind (FILE *fp*);
- Ορίζει σαν τρέχουσα θέση την αρχή του αρχείου.
- Δεν επιστρέφει κάποια τιμή.
- Ίδιο αποτέλεσμα με το *fseek(fp, 0, SEEK_SET)*.

fflush()

- Πρότυπο: *int fflush(FILE *fp);*
- Χρήσιμη συνάρτηση που αδειάζει τον χώρο της προσωρινής αποθήκευσης (buffer) του αρχείου με το οποίο σχετίζεται ο δείκτης *fp*.
- Επιστρέφει *0* αν είναι επιτυχής και *EOF* σε περίπτωση αποτυχίας.
- *fflush (NULL)*: αδειάζουν τα buffers, όλων των αρχείων.

Και άλλες χρήσιμες συναρτήσεις

ftell()

- Πρότυπο: *long int ftell(FILE *fp);*
- Επιστρέφει τη τρέχουσα θέση του αρχείου στο οποίο δείχνει ο *fp* (επιτυχία).
- Σε περίπτωση αποτυχίας επιστρέφει `-1`.

remove()

- *remove (file_name)*: διαγράφει το αρχείο.

Περισσότερη εξάσκηση

- Λύστε τη σειρά των ασκήσεων που έχουν αναρτηθεί στο δικτυακό τόπο του μαθήματος.
- Αναρτήστε τις στη ειδική εργασία (εβδομαδιαία άσκηση) που έχει δημιουργηθεί.
- Χρονικά περιθώρια: αναφέρονται στη περιγραφή της άσκησης