9. Αρχεία

Σύνοψη
Στο κεφάλαιο αυτό μελετώνται τα αρχεία. Αρχικά, ο αναγνώστης εισάγεται στις έννοιες των καναλιών ή ροών και της ενδιάμεσης μνήμης και δίνεται ο ορισμός του αρχείου στη γλώσσα C. Μελετώνται τα αρχεία κειμένου και τα δυαδικά αρχεία, όπου περιγράφονται οι λειτουργίες ανοίγματος/ κλεισίματος, ανάγνωσης και εγγραφής δεδομένων. Ακολούθως, παρουσιάζεται η τυχαία προσπέλαση δυαδικού αρχείου και η ανάγνωση/ εγγραφή ανά γραμμή. Στην τελευταία ενότητα παρουσιάζεται ένα εκτενές παράδειγμα ανάπτυξης προγράμματος, στο οποίο υλοποιείται πλήθος λειτουργιών των αρχείων και γίνεται χρήση εννοιών και εργαλείων που αναπτύχθηκαν στα προηγούμενα κεφάλαια.

Λέξεις κλειδιά
κανάλια (ροές) – αρχείο κειμένου – δυαδικό αρχείο – δείκτης αρχείου – fopen – fclose – fprintf – fscanf – fgets – fputs – getc – putc – fread – fwrite – EOF – feof – τυχαία προσπέλαση – fseek – ftell.

Προαπαιτούμενη γνώση
Λεξιλόγιο της γλώσσας C – μεταβλητές – εκφράσεις – τελεστές – έλεγχος ροής προγράμματος – συναρτήσεις – πίνακες – δείκτες – δυναμική διαχείριση μνήμης – δομές.

9.1 Γενικά

Τα **αρχεία** (files) μπορούν να θεωρηθούν ως σύνθετοι τύποι δεδομένων, οι οποίοι δεν αποθηκεύουν τα δεδομένα τους στην κύρια μνήμη αλλά σε εξωτερικά μέσα αποθήκευσης, όπως οι σκληροί δίσκοι, οι μνήμες flash, τα cd/dvd κ.λπ. Με τον τρόπο αυτό τα δεδομένα ενός αρχείου δεν εκλείπουν με το πέρας του προγράμματος στο οποίο δημιουργήθηκαν, αλλά διατηρούνται στα μέσα αποθήκευσης και μπορούν να ανακτηθούν και να τροποποιηθούν ανά πάσα στιγμή.

Η γλώσσα C θεωρεί κάθε αρχείο ως μία σειριακή ακολουθία από bytes. Το τέλος ενός αρχείου σηματοδοτείται από το *τέλος αρχείου* (end-of-file, **EOF**), που είναι ένας ακέραιος με τιμή **-1**.

9.1.1 Τα κανάλια stdin, stdout, stderr

Κάθε φορά που ξεκινά η εκτέλεση ενός προγράμματος, ο υπολογιστής ανοίγει αυτόματα:

1. Το **κανάλι καθιερωμένης εισόδου** `stdin` (standard input), το οποίο χρησιμοποιείται για ανάγνωση από την κονσόλα. Όταν γίνεται χρήση των `scanf()`, `gets()`, για να αναγνωσθούν δεδομένα από το πληκτρολόγιο, είναι σαν να γίνεται ανάγνωση από το «αρχείο» `stdin`.

2. Το **κανάλι καθιερωμένης εξόδου** `stdout` (standard output) και το κανάλι σφαλμάτων `stderr` (standard errors), τα οποία χρησιμοποιούνται για εκτύπωση στην κονσόλα. Όταν γίνεται χρήση των `printf()`, `puts()`, για να εκτυπωθούν δεδομένα στην οθόνη, είναι σαν να γράφονται τα δεδομένα στο «αρχείο» `stdout`.

Τα ανωτέρω κανάλια ή **ροές** (streams) αποτελούν τα μέσα επικοινωνίας των αρχείων με τα προγράμματα, καθώς, επειδή αποτελούν **δείκτες αρχείου** (file pointers, `FILE *`), μπορούν να χρησιμοποιηθούν σε οποιαδήποτε συνάρτηση χρησιμοποιεί μία μεταβλητή τύπου `FILE`. Έτσι, το κανάλι

stderr μπορεί να ανακατευθυνθεί και να γράφονται τα μηνύματα λάθους σε αρχείο αντί να εμφανίζονται στην οθόνη.

Όταν εκτελείται ένα πρόγραμμα **program_name.c**, με την εντολή **program_name < filename** ορίζεται ως η καθιερωμένη είσοδος αντί του πληκτρολογίου το αρχείο **filename**. Αντίστοιχα, με την εντολή **program_name > filename** ορίζεται ως κύρια έξοδος αντί για την οθόνη το αρχείο **filename**. Οι εντολές αυτές δίνονται από τη γραμμή διαταγής (command line).

Παρατήρηση: Τα **stdin**, **stdout**, **stderr** δεν είναι μεταβλητές αλλά σταθερές και δεν μπορούν να μεταβληθούν. Όπως ο υπολογιστής δημιουργεί αυτόματα αυτούς τους δείκτες αρχείου στην αρχή του προγράμματος, έτσι και τους αποσύρει αυτόματα στο τέλος του προγράμματος. Δεν θα πρέπει να κλείσουν αυτά τα κανάλια με παρέμβαση του χρήστη.

9.1.2 Η ενδιάμεση μνήμη – δείκτης αρχείου

Για ανάγνωση και εγγραφή σε συσκευές εισόδου/εξόδου (input/output, **I/O**) όπως ο σκληρός δίσκος, τα λειτουργικά συστήματα χρησιμοποιούν **ενδιάμεση μνήμη** (buffers), η οποία είναι περιοχή της κύριας μνήμης όπου τα δεδομένα αποθηκεύονται προσωρινά, πριν σταλούν στον τελικό τους στόχο. Έτσι, επιταχύνονται τα προγράμματα, γιατί ελαχιστοποιείται ο αριθμός των προσβάσεων στις I/O συσκευές.

Οι μονάδες I/O επιτρέπουν στο λειτουργικό σύστημα να έχει πρόσβαση μόνο σε καθορισμένου μεγέθους τμήματα, τα ονομαζόμενα **blocks**, μεγέθους **512** ή **1024** bytes. Επομένως, ακόμη κι αν θέλουμε να διαβάσουμε μόνο έναν χαρακτήρα από ένα αρχείο, στην πράξη διαβάζεται όλο το μπλοκ, στο οποίο βρίσκεται αποθηκευμένος ο χαρακτήρας. Έτσι, με τη χρήση του buffer εάν χρειαστούμε άλλους χαρακτήρες από το ίδιο μπλοκ, δεν επιστρέφουμε στη συσκευή, αλλά τους διαβάζουμε από τον buffer.

Το νήμα που κρατάει ενωμένο το **σύστημα I/O με ενδιάμεση αποθήκευση**, δηλαδή με χρήση της ενδιάμεσης μνήμης, είναι ο δείκτης αρχείου. Ο δείκτης αρχείου δείχνει σε πληροφορίες που καθορίζουν διάφορα ζητήματα του αρχείου, όπως είναι το όνομά του, η κατάστασή του και η τρέχουσα θέση του. Ουσιαστικά ο δείκτης αρχείου κατονομάζει ένα συγκεκριμένο αρχείο στο μέσο αποθήκευσης (π.χ. σκληρός δίσκος) και χρησιμοποιείται από το σχετικό κανάλι, για να κατευθύνει τις συναρτήσεις του συστήματος I/O εκεί όπου πρέπει να ενεργήσουν. Ο τύπος του δείκτη αρχείου (**FILE**) ορίζεται στο αρχείο κεφαλίδας **stdio.h**. Για την ανάγνωση ή την εγγραφή αρχείου πρέπει να χρησιμοποιούνται δείκτες αρχείου. Μία μεταβλητή δείκτη αρχείου δηλώνεται ως εξής:

```
FILE *fp;
```

Για λόγους συμβατότητας στον συμβολισμό, έχει επικρατήσει τα ονόματα των δεικτών αρχείου να αρχίζουν από ***f*** (**f**ile).

9.1.3 Κατηγορίες αρχείων

Η γλώσσα C υποστηρίζει δύο κατηγορίες αρχείων, ανάλογα με τον τρόπο που αποθηκεύονται τα δεδομένα:

- Τα **δυαδικά αρχεία** (binary files), τα οποία αποθηκεύουν όλους τους τύπους δεδομένων: **char**, **int**, **float**, **double**, δομή, απαριθμητικό τύπο κ.λπ. Ο τρόπος αποθήκευσης είναι ίδιος με εκείνον της κύριας μνήμης, δηλαδή δε γίνεται μεταγλώττιση των bytes αλλά απλώς διαβάζονται και γράφονται bits, ακριβώς όπως αυτά εμφανίζονται. Για παράδειγμα, ο αριθμός **12345678** εγγράφεται σε δυαδικό αρχείο ως ακέραιος, απαιτώντας **4** bytes.

  Τα δυαδικά αρχεία συνήθως δεν είναι αναγνώσιμα από τους κειμενογράφους (editors) και αναγιγνώσκονται μέσα από προγράμματα (π.χ. τα εκτελέσιμα αρχεία είναι δυαδικά). Ορισμένες φορές δεν είναι *φορητά* (δεν ανοίγουν σε όλα τα μηχανήματα).

- Τα **αρχεία κειμένου** (text files), στα οποία τα δεδομένα αποθηκεύονται ως μία ακολουθία από bytes χαρακτήρων. Ο αριθμός **12345678** εγγράφεται ως αλφαριθμητικό σε αρχείο κειμένου, απαιτώντας **9** bytes (ένα για κάθε χαρακτήρα κι ένα για τον χαρακτήρα τερματισμού **'\0'**).

Τα αρχεία κειμένου είναι αναγνώσιμα από τους συντάκτες. Μάλιστα, τα προγράμματα της γλώσσας C αποθηκεύονται ως αρχεία κειμένου (π.χ. αρχεία **.h**, **.c**). Τέλος, τα αρχεία κειμένου είναι φορητά σε κάθε υπολογιστή.

Ο τρόπος αποθήκευσης των δεδομένων δεν είναι η μοναδική διαφορά ανάμεσα στις δύο κατηγορίες αρχείων. Υπάρχουν διαφορές ανάμεσα στον τρόπο ερμηνείας του χαρακτήρα νέας γραμμής και του χαρακτήρα τέλους του αρχείου, οι οποίες θα μελετηθούν παρακάτω. Οι δύο μορφές αρχείων χρησιμοποιούνται εξίσου αποτελεσματικά, απλώς έχουν διαφορετικό πεδίο εφαρμογών.

9.2 Άνοιγμα – κλείσιμο αρχείου

Για να επικοινωνήσει ένα πρόγραμμα με ένα αρχείο, θα πρέπει το τελευταίο να δηλωθεί μέσα στο πρόγραμμα. Η δήλωση γίνεται με τη διαδικασία ανοίγματος του αρχείου, η οποία ακολουθεί τον εξής φορμαλισμό:

```
fp=fopen(filename, mode);
```

όπου ο δείκτης αρχείου **fp** έχει δηλωθεί προηγουμένως με τη δήλωση **FILE *fp;**

- Η συνάρτηση **fopen()** δεσμεύει τους απαραίτητους πόρους από το λειτουργικό σύστημα, δημιουργεί το κανάλι επικοινωνίας και επιστρέφει στο πρόγραμμα που την κάλεσε έναν δείκτη **fp**, ο οποίος δείχνει σε δομή τύπου **FILE**. Σε περίπτωση σφάλματος, όταν είτε δεν υπάρχει ένα αρχείο προς ανάγνωση είτε δεν υπάρχει αποθηκευτικός χώρος για τη δημιουργία νέου αρχείου προς εγγραφή, επιστρέφεται το **NULL**. Όλες οι προσπελάσεις γίνονται μέσω του δείκτη. Ο δείκτης **fp** χειρίζεται το αρχείο μέσα στο πρόγραμμα. Ένα από τα πεδία της δομής **FILE** είναι ο **δείκτης θέσης αρχείου** (file position indicator), ο οποίος δείχνει στο byte απ' όπου ο επόμενος χαρακτήρας πρόκειται να διαβαστεί ή όπου ο επόμενος χαρακτήρας πρόκειται να εγγραφεί.

Η συμβολοσειρά **filename** είναι το φυσικό όνομα του αρχείου, με το οποίο αποθηκεύεται στη συσκευή αποθήκευσης. Εάν δοθεί μόνο ένα όνομα, το αρχείο θα αποθηκευτεί ή θα αναζητηθεί στον τρέχοντα κατάλογο. Υπάρχει η δυνατότητα να δοθεί ολόκληρο το μονοπάτι μέσα στη συσκευή αποθήκευσης:

```
"c:\\temporary\\c_folder\\myfile.txt"
```

Για τον καθορισμό του ονόματος του αρχείου από τον χρήστη κατά τη διάρκεια εκτέλεσης του προγράμματος μπορεί να χρησιμοποιηθεί ο ακόλουθος κώδικας:

```
char *name;              /* Εναλλακτικά char name[30]; */
printf( "Enter filename -> " );
scanf( "%s",name );      /* Ανάγνωση του ονόματος του αρχείου */
fp=fopen(name,"r");
```

Η συμβολοσειρά **mode** ελέγχει το είδος της πρόσβασης στο αρχείο (εγγραφή, ανάγνωση κ.λπ.). Για παράδειγμα, εάν τεθεί

```
fp=fopen("myfile.txt","r");
```

τότε το αρχείο **myfile.txt** θα χρησιμοποιηθεί για ανάγνωση.

Παράμετροι προσδιορισμού του τρόπου πρόσβασης σε αρχεία κειμένου:

- **r**: Άνοιγμα αρχείου για ανάγνωση. Ο δείκτης θέσης αρχείου βρίσκεται στην αρχή του κειμένου.
- **w**: Δημιουργία νέου αρχείου για εγγραφή. Εάν το αρχείο υπάρχει ήδη, το μέγεθός του θα μηδενιστεί και τα περιεχόμενα θα διαγραφούν. Ο δείκτης θέσης αρχείου τίθεται στην αρχή του αρχείου.
- **a**: Άνοιγμα υπάρχοντος αρχείου κειμένου, στο οποίο όμως μπορούμε να γράψουμε μόνο στο τέλος του αρχείου (προσάρτηση σε αρχείο).
- **r+**: Άνοιγμα υπάρχοντος αρχείου κειμένου για ανάγνωση και εγγραφή. Ο δείκτης θέσης αρχείου τίθεται στην αρχή του αρχείου.
- **w+**:Δημιουργία νέου αρχείου για ανάγνωση και εγγραφή. Εάν το αρχείο υπάρχει ήδη, το μέγεθός του θα μηδενιστεί και τα περιεχόμενα θα διαγραφούν.

- **a+**: Άνοιγμα υπάρχοντος αρχείου ή δημιουργία νέου σε μορφή προσάρτησης. Μπορούμε να διαβάσουμε δεδομένα από οποιοδήποτε σημείο του αρχείου, αλλά μπορούμε να γράψουμε δεδομένα μόνο στη θέση του δείκτη **EOF**.

> *Οι προσδιοριστές για τα δυαδικά αρχεία είναι ίδιοι, με τη διαφορά ότι έχουν ένα* **b** *που τους ακολουθεί. Έτσι, για να ανοίξουμε ένα δυαδικό αρχείο προς ανάγνωση, θα πρέπει να χρησιμοποιήσουμε τον προσδιοριστή* **rb**.

Το κλείσιμο ενός αρχείου γίνεται μετά το τέλος της χρήσης της συνάρτησης **fclose()**:

```
fclose(fp);
```

Όταν το αρχείο κλείσει σωστά, επιστρέφεται το **0**, ενώ σε περίπτωση σφάλματος επιστρέφεται το **EOF**.

Παρατηρήσεις:

1. Ο δείκτης **FILE** χειρίζεται κατά τρόπο αποκλειστικό το αρχείο και σε δείκτες τέτοιου τύπου δεν επιτρέπεται αριθμητική δεικτών. Π.χ, θεωρώντας τον δείκτη **fp** ως έναν δείκτη σε αρχείο:

```
fclose(fp);     σωστό
fclose(fp+1);   λάθος
```

2. Η συνάρτηση **fopen()** δεσμεύει μνήμη. Εάν αμεληθεί να απελευθερωθεί με χρήση της **fclose()**, θα υπάρξει διαρροή μνήμης. Για τον λόγο αυτό θα πρέπει πάντοτε να γίνεται έλεγχος κατά πόσον μία **fopen()** συνοδεύεται από την αντίστοιχη **fclose()**.

3. Η συνάρτηση **fcloseall()** κλείνει όλα τα αρχεία που είναι ανοικτά τη στιγμή της εφαρμογής της. Προτείνεται να τοποθετείται στο τέλος των προγραμμάτων, έτσι ώστε να τερματίζονται όλα τα αρχεία που παραμένουν ανοικτά εκ παραδρομής.

9.3 Ανάγνωση – εγγραφή χαρακτήρων σε αρχεία

9.3.1 Η συνάρτηση εγγραφής χαρακτήρων putc

Η συνάρτηση **putc()** χρησιμοποιείται για την εγγραφή χαρακτήρων σε ένα κανάλι που έχει ανοίξει προηγουμένως με την **fopen()**. Το πρωτότυπο της συνάρτησης είναι το εξής:

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

όπου **fp** είναι ο δείκτης αρχείου που επιστρέφεται από την **fopen()** και **ch** είναι ο προς εγγραφή χαρακτήρας. Για ιστορικούς λόγους το όρισμα **ch** είναι τύπου **int**, αλλά χρησιμοποιεί μόνο ένα byte, το byte χαμηλής τάξης. Η **putc()** ορίζεται στο αρχείο κεφαλίδας **stdio.h**.

Εάν η λειτουργία της συνάρτησης επιτύχει, επιστρέφεται ο χαρακτήρας που ενεγράφη. Εάν αποτύχει, θα επιστρέψει το **EOF**.

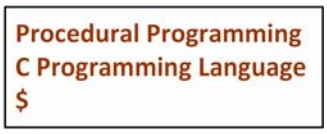
9.3.1.1 Παράδειγμα

Να καταστρωθεί πρόγραμμα, το οποίο διαβάζει χαρακτήρες από το πληκτρολόγιο και τους γράφει σε αρχείο, έως ότου πληκτρολογηθεί το σύμβολο του δολαρίου (**$**).

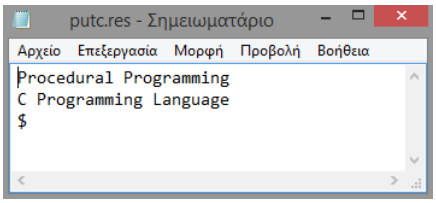
```
#include<stdio.h>

int main()
{
  FILE *fp;
  char ch;
  fp=fopen("putc.res","w");
  if (fp==NULL)
```

```
      printf( "\t\tFILE ERROR: Exit program\n" );
   else
   {
      do
      {
         ch=getchar();
         putc(ch,fp);
      } while (ch!='$');
   }
   fclose(fp);

   return 0;
}
```

Ακολουθούν η έξοδος στην οθόνη (οι χαρακτήρες που πληκτρολογήθηκαν) και το προκύπτον αρχείο **putc.res**. Για την προβολή των αρχείων θα χρησιμοποιείται ο κειμενογράφος *Σημειωματάριο* (*Notepad*).



Εικόνα 9.1.α *Η έξοδος του προγράμματος του παραδείγματος 9.3.1.1*



Εικόνα 9.1.β *Το προκύπτον αρχείο του προγράμματος του παραδείγματος 9.3.1.1*

9.3.2 Η συνάρτηση ανάγνωσης χαρακτήρων getc

Η συνάρτηση `getc()` είναι συμπληρωματική της `putc()` και χρησιμοποιείται για την ανάγνωση χαρακτήρων από ένα κανάλι που έχει ανοίξει προηγουμένως με την `fopen()`. Το πρωτότυπο της συνάρτησης είναι το εξής:

```
int getc(FILE *fp);
```

όπου `fp` είναι ο δείκτης αρχείου που επιστρέφεται από την `fopen()`. Για ιστορικούς λόγους η `getc()` επιστρέφει έναν ακέραιο, αλλά τα bytes υψηλής τάξης είναι μηδέν, άρα μόνο το byte χαμηλής τάξης περιέχει πληροφορία. Η `getc()` ορίζεται στο αρχείο κεφαλίδας `stdio.h`.

Η συνάρτηση `getc()` επιστρέφει **EOF**, όταν ο υπολογιστής φτάσει στο τέλος του αρχείου. Έτσι, για να διαβάσουμε ένα αρχείο κειμένου έως το σημάδι τέλους αρχείου, μπορούμε να χρησιμοποιήσουμε τον ακόλουθο κώδικα:

```
ch=getc(fp);
while (ch!=EOF)
   ch=getc(fp);
```

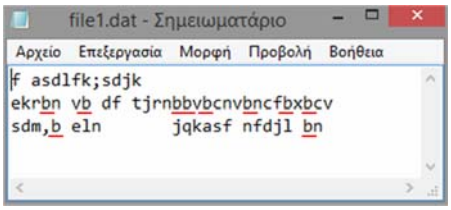
9.3.2.1 Παράδειγμα

Να γραφεί πρόγραμμα, το οποίο θα βρίσκει πόσες φορές υπάρχει ο χαρακτήρας `'\0'` στο αρχείο **file1.dat**.

```
#include <stdio.h>

int  main()
{
```

```
  FILE *fp;
  char charVar;
  int sumb=0;
  fp=fopen("file1.dat","r");
  charVar=getc(fp);
  while (charVar!=EOF)
  {
    if (charVar=='b') sumb++;
    charVar=getc(fp);
  }
  fclose(fp);
  printf( "\nNumber of 'b' appearances: %d",sumb);

  printf("\n\n");

  return 0;
}
```

Η έξοδος του προγράμματος είναι: **Number of 'b' appearances: 10**, όπως προκύπτει από το περιεχόμενο του τυχαία διαμορφωθέντος αρχείου δεδομένων **file1.dat**.



Εικόνα 9.2 *Το αρχείο ανάγνωσης του προγράμματος του παραδείγματος 9.3.2.1*

9.3.2.2 Παράδειγμα

Το παρακάτω πρόγραμμα υπολογίζει τον αριθμό των λέξεων που περιέχονται σε ένα αρχείο ASCII. Το πρόγραμμα χειρίζεται τους λευκούς χαρακτήρες (κενά, νέες γραμμές, στηλοθέτες) ως πραγματικούς χαρακτήρες. Δηλαδή, εάν υπάρχει μία συμβολοσειρά από κενά ή χαρακτήρες επιστροφής, το πρόγραμμα τους διαβάζει και αναμένει για τον πρώτο πραγματικό (μη λευκό) χαρακτήρα. Όλη αυτή τη συμβολοσειρά τη μετρά ως λέξη. Κατόπιν διαβάζει τους πραγματικούς χαρακτήρες έως την εμφάνιση του επόμενου λευκού χαρακτήρα.

Μία μεταβλητή (σημαία) ελέγχει κατά πόσον το πρόγραμμα βρίσκεται στο μέσο μίας λέξης ή στο μέσο κάποιου κενού. Το αρχείο ανάγνωσης δεδομένων είναι εκείνο του παραδείγματος 9.3.2.1.

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

int main()
{
  FILE *fptr;
  char ch,string[81];
  int white=1;    /* Σημαία λευκού χαρακτήρα */
  int count=0;    /* Μετρητής λέξεων */
  fptr=fopen("file1.dat","r");
  if (fptr==NULL)
  {
    printf( "ERROR: can't open file" );
    exit(1);
```

```
    }
    while ((ch=getc(fptr))!=EOF) /* Ανάγνωση χαρακτήρων έως EOF */
    {
       switch(ch)
       {
          /* Έλεγχος για λευκούς χαρακτήρες: τριπλή case με κοινό
             σώμα */
          case ' ':
          case '\t':
          case '\n':
             white++;
          break;
          default:  /* Μη λευκοί χαρακτήρες, μέτρηση λέξεων */
             /* Συνθήκη για αύξηση του μετρητή λέξεων και μηδενισμό της
                σημαίας, ώστε να ξεκινήσει εκ νέου η διαδικασία ελέγχου
                του τέλους μίας λέξης  */
             if (white)
             {
                white=0;
                count++;
             }
          break;
       }
    }
    fclose( fptr );
    printf( "The file contains %d words\n",count );

    return 0;
}
```

Η έξοδος του προγράμματος είναι: **The file contains 11 words**. Από την **Εικόνα 9.2** προκύπτει ότι όντως το αρχείο **file1.dat** περιέχει **11** λέξεις.

9.4 Μορφοποιούμενες συναρτήσεις εισόδου – εξόδου σε αρχεία

9.4.1 Η συνάρτηση fprintf

Η συνάρτηση **fprintf()** χρησιμοποιείται για εγγραφή σε ένα αρχείο. Έχει τους ίδιους μορφολογικούς κανόνες με την **printf()**, με τη διαφορά ότι το πρώτο όρισμα είναι ο δείκτης αρχείου, στο οποίο θα γίνει η εγγραφή:

```
fprintf( fp, ορίσματα );
```

Η **fprintf()** επιστρέφει έναν ακέραιο, ο οποίος είναι ο αριθμός των bytes που ενεγράφησαν. Σε περίπτωση σφάλματος επιστρέφει **EOF**.

Παρατήρηση:
Η συνάρτηση **printf(ορίσματα)** ισοδυναμεί με την **fprintf(stdout, ορίσματα)**, δηλαδή με την **fprintf()** που έχει κανάλι εξόδου την οθόνη αντί για αρχείο.

9.4.1.1 Παράδειγμα

Να περιγραφεί η λειτουργία του ακόλουθου προγράμματος:

```c
#include <stdio.h>

int main()
{
   int cnt;
   FILE *fp;
   char *filename="testfile.txt";
   char msg[40]="This is my song!\n";
   fp=fopen(filename,"w");
   cnt=fprintf( fp,"%s,yep!%d,%f,\n",msg,21,34.5 );
   printf( "Number of bytes written in %s:  %d\n",filename,cnt );
   fclose(fp);

   return 0;
}
```

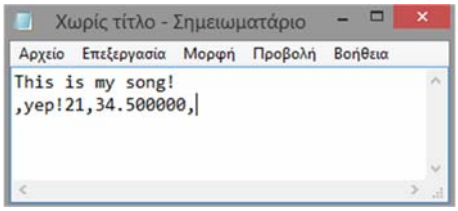
Αρχικά, ορίζεται η ακέραια μεταβλητή **cnt** και ο δείκτης αρχείου **fp**. Ακολουθεί ο ορισμός ενός δείκτη χαρακτήρων **filename**, ο οποίος δείχνει στο αλφαριθμητικό **testfile.txt**, και το αλφαριθμητικό **msg** που έχει αρχικοποιηθεί.

Ο δείκτης **fp** ορίζεται να δείχνει σε αρχείο με φυσικό όνομα το αλφαριθμητικό, στο οποίο δείχνει ο **filename**, δηλαδή το **testfile.txt**. Η συνάρτηση **fprintf()** θα τυπώσει στο αρχείο **testfile.txt**:

- τη συμβολοσειρά **msg** (**17** bytes), στο τέλος της οποίας δηλώνεται αλλαγή γραμμής,
- τους χαρακτήρες **,yep!** (**5** bytes),
- τον ακέραιο **21** (**2** bytes),
- το κόμμα (**1** byte),
- τον αριθμό κινητής υποδιαστολής **34.5** (**9** bytes, καθώς τυπώνεται ως **34.500000**),
- το κόμμα και την αλλαγή γραμμής (**2** bytes).

Η έξοδος του προγράμματος είναι: **Number of bytes written in testfile.txt: 36**.

Στο τέλος του προγράμματος το αρχείο **testfile.txt** κλείνει με χρήση της **fclose()**.



Εικόνα 9.3 *Το αρχείο εγγραφής του προγράμματος του παραδείγματος 9.3.2.2*

9.4.2 Η συνάρτηση fscanf

Η συνάρτηση **fscanf()** χρησιμοποιείται για ανάγνωση δεδομένων από ένα αρχείο. Έχει τους ίδιους μορφολογικούς κανόνες με την **scanf()**, με τη διαφορά ότι το πρώτο όρισμα είναι ο δείκτης αρχείου, από το οποίο θα γίνει η ανάγνωση:

```c
fscanf( pF, ορίσματα );
```

Η **fscanf()** επιστρέφει έναν ακέραιο, ο οποίος είναι ο αριθμός των στοιχείων που ανεγνώσθησαν. Σε περίπτωση σφάλματος θα επιστραφεί **EOF**, εάν επιχειρηθεί ανάγνωση στο τέλος του αρχείου ή το **0**, εάν δεν υπάρχουν δεδομένα προς ανάγνωση.

Παρατήρηση:
Η συνάρτηση **scanf(ορίσματα)** ισοδυναμεί με την **fscanf(stdin, ορίσματα)** δηλαδή με την **fscanf()** που έχει κανάλι εισόδου το πληκτρολόγιο αντί για αρχείο.

9.4.2.1 Παράδειγμα

Να γραφεί πρόγραμμα, το οποίο θα δημιουργεί μονοδιάστατο πίνακα τριών θέσεων, με στοιχεία δομές. Κάθε δομή θα έχει ως μέλη το όνομα, το επώνυμο και το τηλέφωνο ενός ανθρώπου. Το πρόγραμμα θα διαβάζει τα περιεχόμενα του πίνακα από προϋπάρχον αρχείο **file1.dat**, θα τα αποδίδει στον πίνακα και θα τα γράφει σε ένα άλλο αρχείο, το **file2.dat**.

```
#include <stdio.h>
#include <assert.h>

#define N 3

struct structTypeT;
{
  char nm[40];
  char srnm[40];
  char phNo[15];
};

int main()
{
  struct structTypeT id[N];
  FILE *f1,*f2;
  int i;

  f1=fopen("file1.dat","r");     assert( f1!=NULL );
  f2=fopen( "file2.dat","w" );
  for (i=0; i<N; i++)
  {
    fscanf( f1,"%s %s %s\n",id[i].nm,id[i].srnm,id[i].phNo );
    fprintf( f1,"%s %s %s\n",id[i].nm,id[i].srnm,id[i].phNo );
  }
  fclose(f2);
  close(f1);

  return 0;
}
```

9.5 Ανάγνωση – εγγραφή σε δυαδικά αρχεία

Αν και η χρήση των **fprintf()**, **fscanf()** είναι συχνά ο πιο εύκολος τρόπος, για να γράφουμε σε αρχείο ή να διαβάζουμε από ένα αρχείο, δεν είναι πάντοτε και ο πιο αποτελεσματικός. Επειδή γράφουμε φορμαρισμένα δεδομένα ASCII – όπως δηλαδή αυτά θα εμφανίζονταν στην οθόνη – και όχι δυαδικά, κάνουμε περισσότερα πράγματα σε κάθε κλήση και καταλαμβάνουμε περισσότερο χώρο. Έτσι, εάν ενδιαφέρει η ταχύτητα ή το μέγεθος του αρχείου, θα πρέπει πιθανώς να χρησιμοποιήσουμε δυαδικά αρχεία.

Πέραν των ζητημάτων ταχύτητας και αποθηκευτικού χώρου, η χρήση μορφοποιούμενων συναρτήσεων ανάγνωσης–εγγραφής παρουσιάζει ένα άλλο πρόβλημα: δεν υπάρχει άμεσος τρόπος ανάγνωσης και εγγραφής πολύπλοκων τύπων δεδομένων, όπως πίνακες και δομές, καθώς με τις συναρτήσεις αυτές κάθε

φορά γράφεται/ διαβάζεται ένα στοιχείο του πίνακα ή της δομής. Για ανάγνωση και εγγραφή τέτοιων τύπων δεδομένων με μία μόνο πρόταση χρησιμοποιείται το ζεύγος των συναρτήσεων **fread()**/**fwrite()**.

9.5.1 Η συνάρτηση fread

Η συνάρτηση **fread()** χρησιμοποιείται για την ανάγνωση μπλοκ δεδομένων από ένα αρχείο. Ορίζεται στο αρχείο κεφαλίδας **stdio.h** και έχει το ακόλουθο πρωτότυπο:

```
int fread(void *buffer, int length, int numItems, FILE *fp);
```

όπου

- **buffer** είναι ένας δείκτης σε μία περιοχή της μνήμης, η οποία θα δεχθεί τα δεδομένα που διαβάζονται από το αρχείο.
- **length** είναι το μέγεθος του τύπου των δεδομένων που θα αναγνωσθούν. Για τον προσδιορισμό τους χρησιμοποιείται η **sizeof**.
- **numItems** είναι ο αριθμός των στοιχείων (μήκους **length** bytes το καθένα) που θα αναγνωσθούν.
- **fp** είναι ο δείκτης του προς ανάγνωση αρχείου.

Η **fread()** επιστρέφει έναν ακέραιο, ο οποίος είναι ο αριθμός των στοιχείων (όχι των bytes) που ανεγνώσθησαν επιτυχώς.

Αξίζει να σημειωθεί ότι, για να λειτουργήσει επιτυχώς η **fread()**, θα πρέπει η περιοχή προσωρινής αποθήκευσης, που καθορίζεται από το **buffer**, αφενός μεν να αποθηκεύει ίδιου τύπου με τα προς ανάγνωση δεδομένα, αφετέρου δε να έχει επαρκή μνήμη.

9.5.1.1 Παράδειγμα

Να περιγραφεί η λειτουργία του ακόλουθου προγράμματος:

```
#include <stdio.h>
#include <assert.h>
#include <stdlib.h>

int main()
{
  FILE *fp;
  int buf[40],i,cnt,n=24;
  fp=fopen("file.dat","rb");  assert(fp!=NULL);
  cnt=fread(buf,sizeof(int),n,fp);
  if (cnt!=n)
  {
    printf( "ERROR");
    exit(-1);
  }
  printf( "Number of items read:  %d\n",cnt );
  fclose(fp);

  return 0;
}
```

Αρχικά ορίζεται ο **buffer** ως πίνακας ακεραίων με μέγεθος **40** και ο αριθμός των προς ανάγνωση δεδομένων **n**, με τιμή **24**. Ακολούθως, ανοίγει για ανάγνωση το δυαδικό αρχείο **file.dat**, το οποίο περιλαμβάνει **32** ακεραίους. Με χρήση της **fread()** διαβάζονται τα δεδομένα και η **cnt** γίνεται ίση με τη

n. Σε περίπτωση σφάλματος έχει ληφθεί πρόνοια για έξοδο από το πρόγραμμα. Στο τέλος το προγράμματος κλείνει το αρχείο **file.dat**. Η έξοδος του προγράμματος είναι: **Number of items read: 24**.

9.5.2 Η συνάρτηση fwrite

Η συνάρτηση **fwrite()** χρησιμοποιείται για την εκτύπωση μπλοκ δεδομένων σε ένα αρχείο. Ορίζεται στο αρχείο κεφαλίδας **stdio.h** και έχει το ακόλουθο πρωτότυπο:

```
int fwrite(void *buffer, int length, int num_items, FILE *fp);
```

όπου τα ορίσματα είναι ακριβώς τα ίδια με εκείνα της **fread()** (εξυπακούεται ότι στον **buffer** θα αποθηκεύονται πλέον τα προς εγγραφή δεδομένα και η έξοδος θα επιστρέφει τον αριθμό των στοιχείων που ενεγράφησαν επιτυχώς).

Παρατηρήσεις:

1. Στην περίπτωση εγγραφής αλφαριθμητικών, ένα συχνό σφάλμα που γίνεται, είναι να χρησιμοποιείται η **strlen()** για τον υπολογισμό των χαρακτήρων του αλφαριθμητικού, παραλείποντας όμως τον τερματιστή του (τον μηδενικό χαρακτήρα **'\0'**).

Για παράδειγμα, στον ακόλουθο κώδικα η εγγραφή του αλφαριθμητικού είναι ατελής:

```
int cnt;
FILE *pF;
char msg[40]="This is my song!";
pF=fopen("music.mdi","wb");
cnt=fwrite(msg,sizeof(char),strlen(msg),pF);
```

Το σφάλμα διορθώνεται με την προσθήκη μίας μονάδας στη **strlen()**, ώστε να περιληφθεί ο χαρακτήρας τερματισμού του αλφαριθμητικού:

```
cnt= fwrite(msg, sizeof(char),1+strlen(msg),pF);
```

2. Οι εντολές **fwrite**/**fwrite()** μπορούν να χρησιμοποιηθούν με τον ίδιο τρόπο και σε αρχεία κειμένου, όπου κάθε χαρακτήρας θα διαβάζεται/ εγγράφεται ως ξεχωριστό δεδομένο.

9.5.2.1 Παράδειγμα

Να περιγραφεί η λειτουργία του ακόλουθου προγράμματος:

```
#include <stdio.h>
#include <assert.h>
#include <stdlib.h>

int main()
{
  FILE *fp;
  int buf[40];
  int i,cnt,n=32;
  fp=fopen("file.dat","wb");      assert(fp!=NULL);
  for (i=1; i<=n; i++)
     buf[i]=2*i;
  cnt=fwrite(buf,sizeof(int),n,fp);
  if(cnt!=n)
  {
     printf( "ERROR");
     exit(-1);
  }
```

```
  printf( "Number of items written:  %d\n",cnt );
  fclose(pf);

  return 0;
}
```

Αρχικά, ορίζεται ο **buffer** ως πίνακας ακεραίων με μέγεθος **40** και ο αριθμός των προς εγγραφή δεδομένων **n**, με τιμή **32**. Ακολούθως, ανοίγει για εγγραφή το δυαδικό αρχείο **file.dat**. Με χρήση της **fwrite()** εκτυπώνονται τα δεδομένα και η **cnt** γίνεται ίση με τη **n**, όπως φαίνεται στην έξοδο στην οθόνη. Σε περίπτωση σφάλματος γίνεται άμεση έξοδος από το πρόγραμμα. Στο τέλος το προγράμματος κλείνει το αρχείο **file.dat**. Η έξοδος του προγράμματος είναι: **Number of items written: 32**.

9.5.2.2 Παράδειγμα

Να γραφεί κώδικας για το παράδειγμα 9.4.2.1 με χρήση δυαδικών αρχείων και των συναρτήσεων **fread()**/**fwrite()**.

```
#include <stdio.h>
#include <assert.h>

#define N 3

struct structTypeT;
{
  char nm[40];
  char srnm[40];
  char phNo[15];
};

int main()
{
  structTypeT id[N];
  FILE *f1;
  int i;
  f1=fopen("file1.dat","rb");    assert( f1!=NULL );
  for (i=0;i<N;i++)
    fread(&id[i],sizeof(id[i]),1,f1);
  fclose( f1 );
  f1=fopen("file2.dat","wb");    assert( f1!=NULL );
  for (i=0;i<N;i++)
    fwrite(&id[i],sizeof(id[i]),1,f1);
  fclose(f1);

  return 0;
}
```

9.5.2.3 Παράδειγμα

Στο πρόγραμμα που ακολουθεί διαβάζονται χαρακτήρες από το πληκτρολόγιο και εγγράφονται σε δυαδικό αρχείο, το οποίο λόγω της φύσης των περιεχομένων του (χαρακτήρες) είναι αναγνώσιμο με τους συντάκτες κειμένου. Ως συνθήκη τερματισμού θεωρείται η ανάγνωση του δολαρίου (**$**).

Ακολούθως, με χρήση της συνάρτησης

```
void readCharacter(FILE *fp, char *ps, int k);
```

διαβάζεται ο **14ος** χαρακτήρας, χρησιμοποιώντας τυχαία προσπέλαση του αρχείου.

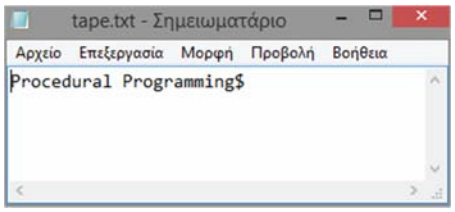
Η συνάρτηση

```
void saveCharacter(FILE *fp, char *ps, int k)
```

τοποθετεί στη **14η** θέση του αρχείου το θαυμαστικό (**!**).

```
#include <stdlib.h>
#include <stdio.h>
#define NAME "tape.txt"
void readCharacter(FILE *fp, char *ps, int k );
void saveCharacter(FILE *fp, char *ps, int k );
int main()   {
  FILE *fp;
  char charVar;
  fp=fopen(NAME,"wb");
  do
  {
     charVar=getchar();
     fwrite(&charVar,sizeof(char),1,fp);
  } while (charVar!='$');
  fclose(fp);
  fp=fopen(NAME,"rb+");
  readCharacter(fp,&charVar,13);
  printf("\n\nThe 14th character is: %c\n",charVar);
  charVar='!';
  saveCharacter(fp,&charVar,13);
  fclose(fp);
  return 0;
}
/*-----------------------------------*/
void readCharacter(FILE *fp, char *ps, int k )
{
  fseek(fp,k*sizeof(char),SEEK_SET);
  fread(ps,sizeof(char),1,fp);
}
/*-----------------------------------*/
void saveCharacter(FILE *fp, char *ps, int k )
{
  fseek(fp,k*sizeof(char),SEEK_SET);
  fwrite(ps,sizeof(char),1,fp);
}
```

Μετά την εγγραφή των δεδομένων και το πρώτο κλείσιμο του αρχείου **tape.txt**, τα περιεχόμενα του αρχείου απεικονίζονται στην **Εικόνα 9.4**, απ' όπου προκύπτει ότι ο **14ος** χαρακτήρας είναι το **'o'**.



Εικόνα 9.4 *Το αρχείο εγγραφής του προγράμματος του παραδείγματος 9.5.2.3 πριν τη μεταβολή του 14ου χαρακτήρα*

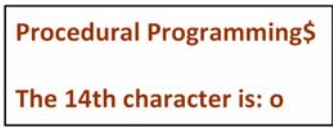
Ακολούθως, το αρχείο **tape.txt** ανοίγει εκ νέου σε κατάσταση **rb+**, επομένως μπορούν και να γραφούν και να αναγνωστούν δεδομένα. Η κλήση της συνάρτησης

```
readCharacter(fp,&charVar,13);
```

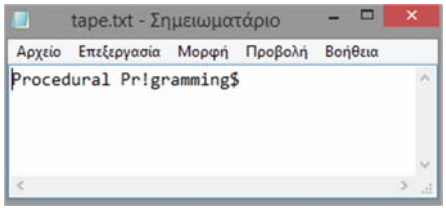
οδηγεί στην ανέρευση του **14ου** χαρακτήρα και η κλήση της συνάρτησης

```
saveCharacter(fp,&charVar,13);
```

οδηγεί στην αντικατάστση του **14ου** χαρακτήρα. Η έξοδος του προγράμματος απεικονίζεται στην **Εικόνα 9.5.α** και η τελική μορφή του αρχείου – με τον αντικατασταθέντα χαρακτήρα – παρουσιάζεται στην **Εικόνα 9.5.β**.



Εικόνα 9.5.α *Η έξοδος του προγράμματος του παραδείγματος 9.5.2.3*



Εικόνα 9.5.β *Το προκύπτον αρχείο του προγράμματος του παραδείγματος 9.5.2.3*

9.5.3 Η συνάρτηση feof

Όταν ανοίγει ένα δυαδικό αρχείο, είναι πιθανόν ο υπολογιστής να διαβάσει μία ακέραια τιμή ίση με **EOF**. Σε μία τέτοια περίπτωση θα δηλωθεί μία συνθήκη τέλους αρχείου, ακόμη κι αν ο υπολογιστής δεν έχει φτάσει στο φυσικό τέλος του αρχείου. Για να λύσει αυτό το πρόβλημα, η γλώσσα C περιλαμβάνει τη συνάρτηση **feof()**, η οποία καθορίζει πού βρίσκεται το σημάδι τέλους αρχείου, όταν διαβάζονται δυαδικά δεδομένα. Η συνάρτηση **feof()** λαμβάνει ως όρισμα έναν δείκτη αρχείου και επιστρέφει **1**, εάν ο υπολογιστής έχει φθάσει στο τέλος του αρχείου ή **0** εάν ο υπολογιστής δεν έχει φτάσει στο τέλος του αρχείου. Η **feof()** ορίζεται στο αρχείο κεφαλίδας **stdio.h**.

Έτσι, για να διαβάσουμε ένα δυαδικό αρχείο έως το σημάδι τέλους αρχείου, μπορούμε να χρησιμοποιήσουμε τον ακόλουθο κώδικα:

```
ch=getc(fp);
while (!feof(fp)) ch=getc(fp);
```

9.6 Ανάγνωση – εγγραφή χαρακτήρων με χρήση των fread/fwrite

Το ζεύγος **fread()/fwrite()** μπορεί να επιτελέσει τη λειτουργία των **getc()/putc()**, όχι μόνο για ένα αλλά για οποιονδήποτε αριθμό χαρακτήρων. Η λειτουργία θα περιγραφεί με τη βοήθεια του ακόλουθου τμήματος κώδικα:

```
FILE *fpin,*fpout;
char buf[100];
int cnt;
fpin=fopen("src.txt","r");
fpout=fopen("dest.txt","w");
if (fpin==NULL)
  exit(-1);
cnt=fread(buf,sizeof(char),100,fpin);
while(cnt==100)
{
```

```
    fwrite(buf,sizeof(char),100,fpout);
    cnt=fread(buf,sizeof(char),100,fpin);
  }
  if (cnt!=0)
    fwrite(buf,sizeof(char),cnt,fpout);
  fclose(fpout);
  fclose(fpin);
```

Αρχικά, ορίζονται οι δείκτες αρχείου **fpin** και **fpout** και ο **buffer** χαρακτήρων **100** θέσεων. Ο **fpin** χρησιμοποιείται, για να ανοίξει το αρχείο ανάγνωσης **src.txt** και ο **fpout**, για να ανοίξει το αρχείο εγγραφής **dest.txt**. Εάν υπάρξει σφάλμα στο άνοιγμα του αρχείου ανάγνωσης, το πρόγραμμα τερματίζεται (**exit(-1)**).

Με την πρόταση

```
cnt=fread(buf,sizeof(char),100,fpin);
```

ζητείται να αναγνωσθούν **100** χαρακτήρες από το **src.txt** με χρήση του **buffer**. Η **fread()** επιστρέφει στη **cnt** τον αριθμό των χαρακτήρων που ανεγνώσθησαν.

Η συνθήκη **while** ελέγχει κατά πόσον γέμισε ο **buffer**. Εάν γέμισε, γράφουμε ολόκληρο τον buffer στο αρχείο εγγραφής **dest.txt** και ακολούθως επαναλαμβάνουμε την ανάγνωση.

Η πρόταση

```
if (cnt!=0)    fwrite(buf,sizeof(char),cnt,fpout);
```

γράφει στο **dest.txt** τα περιεχόμενα του **buffer** που μπορεί να παρέμειναν (εάν ο συνολικός αριθμός των δεδομένων δεν είναι ακριβές πολλαπλάσιο του μεγέθους του **buffer**).

Το πρόγραμμα ολοκληρώνεται με κλείσιμο των αρχείων εγγραφής και ανάγνωσης.

9.7 Ανάγνωση – εγγραφή γραμμή ανά γραμμή

Η γλώσσα C δίνει τη δυνατότητα ανάγνωσης και εγγραφής γραμμή ανά γραμμή με το ζεύγος συναρτήσεων **fgets()/fputs()**. Οι συναρτήσεις ορίζονται στο αρχείο κεφαλίδας **stdio.h** και έχουν τα ακόλουθα πρωτότυπα:

```
char *fgets(char *pstr, int length, FILE *fp);
char *fputs(char *pstr, FILE *fp);
```

Η συνάρτηση **fputs()** λειτουργεί όπως ακριβώς η **puts()**, με τη διαφορά ότι η **fputs()** γράφει στο κανάλι που καθορίζεται. Η συνάρτηση **fgets()** διαβάζει ένα αλφαριθμητικό από το καθορισμένο κανάλι, έως ότου διαβάσει είτε έναν χαρακτήρα νέας γραμμής είτε αριθμό χαρακτήρων ίσο με **length-1**. Εάν η **fgets()** διαβάσει έναν χαρακτήρα νέας γραμμής, ο τελευταίος θα αποτελέσει τμήμα του αλφαριθμητικού (σε αντίθεση με τη **gets()**). Ωστόσο, μόλις τερματίσει η **fgets()**, το αλφαριθμητικό που θα προκύψει θα έχει στο τέλος του τον μηδενικό χαρακτήρα.

Στον ακόλουθο κώδικα

```
char buf[100];
FILE *fpin,*fpout;
. . . . . .
while(fgets(buf,100,fpin)!=NULL)
  fputs(buf,fpout);
```

η πρόταση **fgets(buf,100,fpin)**:

- Θα διαβάσει ένα αλφαριθμητικό από το αρχείο που καθορίζει ο δείκτης **fpin** και θα το αποδώσει στον **buf**.
- Θα σταματήσει μετά τη *νέα γραμμή* ή τον **99^{ο}** χαρακτήρα.

- Θα τοποθετήσει ακολούθως στον **buf** τον μηδενικό χαρακτήρα.
- Θα επιστρέψει τον δείκτη του **buf** σε περίπτωση επιτυχίας ή **NULL** εάν ο **fpin** είναι άδειος.

Η πρόταση **fputs(buf,fpout)** θα εγγράψει στο αρχείο που καθορίζει ο δείκτης **fpout** το περιεχόμενο του **buf**.

9.7.1 Παράδειγμα

Στον ακόλουθο κώδικα γίνεται χρήση των διαφόρων τρόπων ανάγνωσης/εγγραφής σε αρχείο.

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

#define NAME "tape.txt"

int main()
{
  FILE *fp;
  char pchar[16];
  int i;
  float x[5];
  fp=fopen(NAME,"w");
  for (i=0;i<5;i++)
    fprintf( fp,"nm%d.dat\n",i+1 );
  fclose(fp);

  fp=fopen(NAME,"r");
  for (i=0;i<5;i++)
    printf( "line %d: %s\n",i+1,fgets(pchar,15,fp) );
  fclose(fp);

  fp=fopen("data.dat","w");
  for (i=0;i<5;i++)
    x[i]=sqrt(i);
  fwrite(x,sizeof(x),1,fp);
  fclose(fp);
  fp=fopen("data.dat","r");
  for (i=0;i<5;i++)
  {
    fscanf( fp,"%f",&x[i] );
    printf( "x[%d]=%f\n",i,x[i] );
  }
  fclose(fp);

  return 0;
}
```

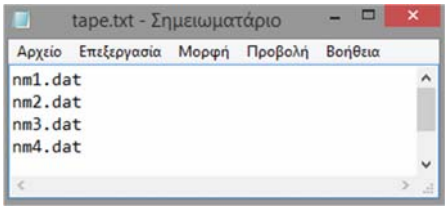
Αρχικά, ανοίγει το αρχείο κειμένου **tape.txt** για εγγραφή, στο οποίο εγγράφονται με χρήση της **fprintf()** πέντε αλφαριθμητικά που αποτελούν ονόματα αρχείων. Τα περιεχόμενα του **tape.txt** απεικονίζονται στην **Εικόνα 9.6.α**.

Μετά την εγγραφή των αλφαριθμητικών το αρχείο κλείνει και ανοίγει εκ νέου για ανάγνωση. Η ανάγνωση γίνεται με χρήση της **fgets()** και μετά το πέρας της το αρχείο κλείνει.

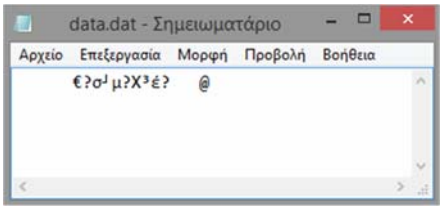
Ακολούθως το **tape.txt** ανοίγει για εγγραφή ως αρχείο κειμένου, γεγονός που σημαίνει ότι τα υφιστάμενα περιεχόμενα διαγράφονται. Η εγγραφή γίνεται με την πρόταση

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

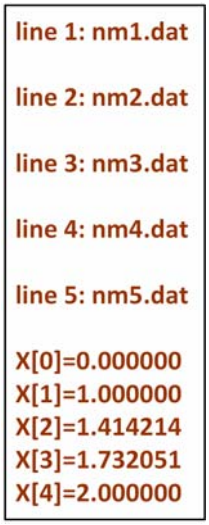
η οποία αντιγράφει ολόκληρο τον πίνακα **x** στο αρχείο. Στην **Εικόνα 9.6.β** απεικονίζονται τα περιεχόμενα του αρχείου μετά την εγγραφή, απ' όπου προκύπτει ότι τα δεδομένα είναι μη αναγνώσιμα από τον κειμενογράφο, γιατί είναι αποθηκευμένα ως δυαδικά δεδομένα. Συνεπώς, καθίσταται φανερό ότι, αν και το αρχείο **data.dat** άνοιξε ως αρχείο κειμένου, η χρήση της **fwrite()** το μετέτρεψε σε δυαδικό αρχείο. Ανοίγοντάς το εκ νέου για ανάγνωση είναι εφικτή η χρήση της **fscanf()**, μόνο εφόσον ζητηθούν επακριβώς ο αριθμός και ο τύπος των δεδομένων που είναι αποθηκευμένα, όπως φαίνεται και από τα αποτελέσματα της **Εικόνας 9.6.γ**.



Εικόνα 9.6.α *Τα περιεχόμενα του αρχείου κειμένου tape.txt του προγράμματος του παραδείγματος 9.7.1*



Εικόνα 9.6.β *Τα περιεχόμενα του δυαδικού αρχείου data.dat του προγράμματος του παραδείγματος 9.7.1*



Εικόνα 9.6.γ *Η έξοδος του προγράμματος του παραδείγματος 9.7.1*

9.8 Τυχαία προσπέλαση δυαδικού αρχείου

Στις προηγούμενες ενότητες τα αρχεία προσπελαύνονταν σειριακά, δηλαδή, για να βρεθεί ένα δεδομένο σε ένα αρχείο, θα έπρεπε να προσπελασθούν πρώτα όλα τα προηγούμενά του. Επιπρόσθετα, για να ενημερωθεί μία εγγραφή του αρχείου (π.χ. ένα όνομα), θα έπρεπε να διαβαστεί όλο το αρχείο, να γίνει η αλλαγή και κατόπιν να ξαναγραφεί.

Γίνεται φανερό ότι η σειριακή προσπέλαση δεν ενδείκνυται σε μεγάλα αρχεία ή σε αρχεία που προσπελαύνονται και τροποποιούνται συχνά. Για αυτές τις περιπτώσεις η γλώσσα C παρέχει τη δυνατότητα **τυχαίας προσπέλασης** (random access), με την οποία παρέχεται πρόσβαση σε οποιοδήποτε σημείο ενός αρχείου. Η τυχαία προσπέλαση στηρίζεται στο γεγονός ότι κάθε ανοικτό αρχείο έχει έναν δείκτη θέσης αρχείου, ο οποίος καθορίζει σε ποιο σημείο του αρχείου θα γίνει ανάγνωση ή εγγραφή. Η θέση αυτή δίνεται ως αριθμός bytes από την αρχή του αρχείου και είναι μία μεταβλητή–μέλος της δομής **FILE**. Όταν το αρχείο ανοίγει για ανάγνωση, η θέση αυτή είναι **0**. Όταν ανοίγει για προσάρτηση, είναι το τέλος του αρχείου. Καθορίζοντας τον δείκτη θέσης αρχείου μπορούμε να έχουμε προσπέλαση σε οποιοδήποτε σημείο του αρχείου.

9.8.1 Συναρτήσεις διαχείρισης της θέσης σε αρχείο

1. **`fseek()`**

Η συνάρτηση **`fseek()`** αποτελεί το εργαλείο για την εκτέλεση λειτουργιών τυχαίας ανάγνωσης και εγγραφής. Ορίζεται στο αρχείο κεφαλίδας **`stdio.h`** και έχει το ακόλουθο πρωτότυπο:

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

όπου

- **`fptr`** είναι ένας δείκτης αρχείου, που επιστρέφεται από την **`fopen()`**.
- **`offset`** είναι ο αριθμός των bytes, που δηλώνουν την απόσταση της νέας θέσης από το όρισμα **`origin`**.
- **`origin`** είναι το σημείο αφετηρίας και μπορεί να έχει μία από τις τρεις ακόλουθες τιμές:
  - *(i)* *Αφετηρία*: αρχή του αρχείου, *τιμή*: **SEEK_SET(0)**
  - *(ii)* *Αφετηρία*: τρέχουσα θέση, *τιμή*: **SEEK_CUR(1)**
  - *(iii)* *Αφετηρία*: τέλος του αρχείου, *τιμή*: **SEEK_END(2)**

Έτσι, για να βρεθεί π.χ. το **`offset`** από την τρέχουσα θέση, το **`origin`** θα πρέπει να λάβει την τιμή **SEEK_CUR**. Σε περίπτωση επιτυχίας η **`fseek()`** επιστρέφει **0**, ενώ εάν αποτύχει επιστρέφει **μη μηδενική** τιμή.

Θα πρέπει να σημειωθεί ότι ο δείκτης θέσης αρχείου επανατοποθετείται στην αρχή με τη συνάρτηση **`rewind(fptr);`**

2. **`ftell()`**

Η συνάρτηση **`ftell()`** ορίζεται στο αρχείο κεφαλίδας **`stdio.h`** και έχει το ακόλουθο πρωτότυπο:

```
long ftell(FILE *fptr)
```

όπου **`fptr`** είναι ένας δείκτης αρχείου, που επιστρέφεται από την **`fopen()`**. Επιστρέφει την τιμή ενός **`long`** ακεραίου, η οποία αντιστοιχεί στην τρέχουσα θέση στο αρχείο. Δηλαδή, η επιστρεφόμενη τιμή διαδραματίζει τον ρόλο του **`offset`** στην **`fseek()`**, όταν ως **`origin`** τεθεί η αρχή του αρχείου.

9.8.1.1 Παράδειγμα

Για τη διαχείριση των στοιχείων των φοιτητών ορίζεται ο πίνακας **`studentList[size]`** με στοιχεία τύπου δομής **`StudentT`**:

```
struct StudentT
{
  int AM,year;
  char firstname[20],lastname[40];
};
```

Για τη διαχείριση μεμονωμένων φοιτητών ορίζεται η μεταβλητή **`svar`**, επίσης τύπου δομής **`StudentT`**.

Για τη διαχείριση των στοιχείων των φοιτητούν θα αναπτυχθούν οι ακόλουθες συναρτήσεις:

1. **`saveData()`**: Αποθήκευση των δεδομένων του πίνακα **`studentList`** στο αρχείο **students.dat**.

2. **`readData()`**: Ανάγνωση των δεδομένων από το αρχείο και αποθήκευσή τους στον πίνακα **`studentList`**.

3. **`readStudent()`**: Προσπέλαση ενός συγκεκριμένου φοιτητή στο αρχείο και αποθήκευση των στοιχείων του σε μία μεταβλητή τύπου **`StudentT`**.

4. **`saveStudent()`**: Αποθήκευση των στοιχείων ενός συγκεκριμένου φοιτητή στο αρχείο.

Υποθέτουμε ότι το δυαδικό αρχείο έχει ανοίξει κανονικά στη **main()**, υπάρχει ένας έγκυρος δείκτης **fptr** σε αυτό και το **size** έχει καθοριστεί με την εντολή προεπεξεργαστή **#define**.

1. saveData()

Η συνάρτηση καλείται από τη **main()** ως εξής:

```
saveData(fptr, studentList);
```

και έχει το ακόλουθο σώμα:

```
void saveData(FILE *fp, StudentT *plist)
{
  int i;
  rewind(fp);
  for (i=0;i<size;i++)
     fwrite(&plist[i],sizeof(StudentT),1,fp);
}
```

Αντί του βρόχου **for** θα μπορούσε να γραφεί:

```
fwrite(plist,sizeof(StudentT),size,fp);
```

2. readData()

Η συνάρτηση καλείται από τη **main()** ως εξής:

```
readData(fptr, studentList);
```

και έχει το ακόλουθο σώμα:

```
void saveData( FILE *fp, StudentT *plist )
{
  rewind(fp);
  fread(plist,sizeof(StudentT),size,fp);
}
```

3. readStudent()

Η συνάρτηση καλείται από τη **main()** ως εξής:

```
read_student(fptr, &svar, i);
```

όπου **svar** είναι μία οποιαδήποτε μεταβλητή τύπου **StudentT** (στη θέση της θα μπορούσε να είναι η **&studentList[i]**), **i** είναι η θέση ενός στοιχείου του πίνακα **studentList**. Το σώμα της είναι το ακόλουθο:

```
void read_student(FILE *fp, Student *ps, int k)
{
  fseek(fp,k*sizeof(StudentT),SEEK_SET);
  fread(ps,sizeof(StudentT),1,fp);
}
```

4. saveStudent()

Η συνάρτηση καλείται από τη **main()** ως εξής:

```
saveStudent(fptr, &studentList[i], i);
```

όπου **i** είναι ο αύξων αριθμός του στον πίνακα **studentList**. Το σώμα της είναι το ακόλουθο:

```
void save_student(FILE *fp, Student *ps, int k)
{
  fseek(fp,k*sizeof(StudentT),SEEK_SET);
  fwrite(ps,sizeof(Student),1,fp);
}
```

Η συνάρτηση **saveStudent()** χρησιμοποιείται, όταν έχουμε κάνει αλλαγές στα στοιχεία ενός φοιτητή και θέλουμε να ενημερώσουμε την εγγραφή του στο αρχείο.

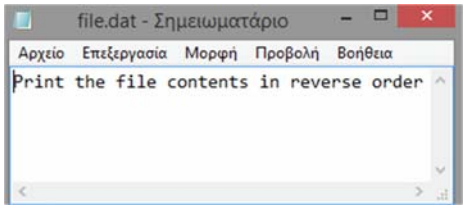
Η συνάρτηση **readStudent()** χρησιμοποιείται, για να αντλήσουμε τα στοιχεία του του φοιτητή από το αρχείο και πιθανώς να τα τροποποιήσουμε.

9.8.1.2 Παράδειγμα

Το πρόγραμμα που ακολουθεί, εμφανίζει τα περιεχόμενα ενός αρχείου με αντίστροφη σειρά.

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

int main(void)
{
   char ch;
   FILE *fp;
   long i,last;
   fp=fopen("file.dat","rb");      assert(fp!=NULL);
   fseek(fp,0,SEEK_END); /* μετακίνηση το τέλος του αρχείου */
   last=ftell(fp);
   for (i=1;i<=last;i++)
   {
      /* μετακίνηση από το τέλος προς τα πίσω */
      fseek(fp,-i,SEEK_END);
      ch=getc(fp);
      if (ch!=EOF)
         putchar(ch);
   }
   putchar('\n');
   fclose(fp);

   return 0;
}
```

Στην **Εικόνα 9.6.α** απεικονίζονται τα περιεχόμενα του αρχείου και στην **Εικόνα 9.6.β** η έξοδος του προγράμματος, από την οποία συνάγεται ότι εκτυπώθηκε αντεστραμμένη η πρόταση, που βρισκόταν αποθηκευμένη στο **file.dat**.



Εικόνα 9.7.α *Το αρχείο του προγράμματος του παραδείγματος 9.8.1.2*

Εικόνα 9.7.β *Η έξοδος του προγράμματος του παραδείγματος 9.8.1.2*

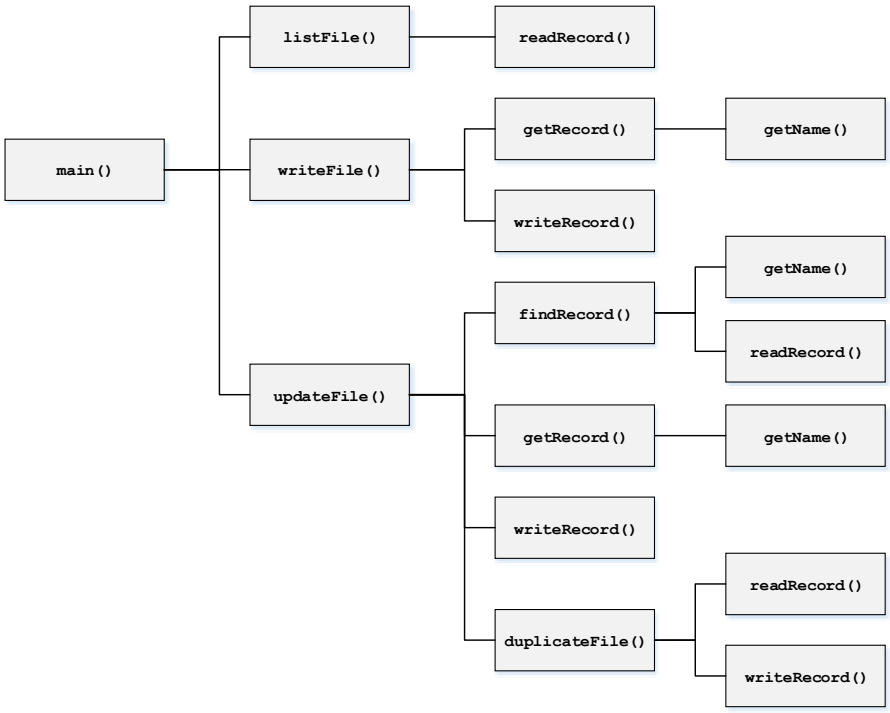
9.9 Παράδειγμα ανάπτυξης προγράμματος

Στο πρόγραμμα που ακολουθεί, θα χρησιμοποιηθούν σύνθετα δεδομένα τύπου δομής (εγγραφές), για να υλοποιηθούν ορισμένες διαδικασίες επεξεργασίας αρχείων, όπως η ενημέρωση των εγγραφών, η προσθήκη

και η διαγραφή εγγραφών. Στο πλαίσιο αυτό, οι λειτουργίες του προγράμματος θα μεριστούν σε δέκα συναρτήσεις:

- **`main()`**: Ελέγχει τη συνολική λειτουργία του προγράμματος και παρέχει στον χρήστη τη δυνατότητα επιλογής από ένα πλήθος λειτουργιών του αρχείου.
- **`updateFile()`**: Τροποποιεί τα περιεχόμενα του αρχείου.
- **`listFile()`**: Εμφανίζει τα περιεχόμενα του αρχείου στην οθόνη (**`stdout`**).
- **`writeFile()`**: Έχει διπλή λειτουργία: την εγγραφή σε ένα νέο αρχείο και την προσάρτηση σε ένα υφιστάμενο αρχείο.
- **`getRecord()`**: Λαμβάνει δεδομένα (εγγραφές) από το πληκτρολόγιο (**`stdin`**).
- **`getName()`**: Διαβάζει ένα όνομα από το πληκτρολόγιο.
- **`writeRecord()`**: Γράφει δεδομένα (εγγραφές) στο αρχείο.
- **`readRecord()`**:Λαμβάνει δεδομένα (εγγραφές) από το αρχείο.
- **`findRecord()`**: Αναζητά μία εγγραφή στο αρχείο, βάσει μίας μεταβλητής–μέλος της εγγραφής.
- **`duplicateFile()`**: Αναπαράγει το αρχείο αντικαθιστώντας μία εγγραφή, όταν αυτή έχει διαφορετικό μήκος από την αντικαθιστώμενη.

Η ιεραρχία κλήσεων των συναρτήσεων απεικονίζεται στο **Σχήμα 9.1**. Οι τρεις συναρτήσεις που καλούνται από τη **`main()`**, υλοποιούν την κύρια λειτουργικότητα του προγράμματος. Οι συναρτήσεις που βρίσκονται στα δεξιά τους απλοποιούν τις λειτουργίες των τριών κυρίων συναρτήσεων.



Σχήμα 9.1 *Η ιεραρχία των κλήσεων των συναρτήσεων του προγράμματος*

Τα δεδομένα του αρχείου (εγγραφές) είναι τύπου δομής. Χάριν ευκολίας θα χρησιμοποιηθεί μία στοιχειώδης δομή, αποτελούμενη από δύο μέλη:

```
struct RecordT
{
  char name[MAXLENGTH];
  int age;
```

```
};
```

1. Ανάγνωση εγγραφής από το πληκτρολόγιο

Το πρωτότυπο της συνάρτησης που θα λαμβάνει δεδομένα από το πληκτρολόγιο είναι το εξής:

```
struct RecordT *getRecord(struct RecordT *precord);
```

Η συνάρτηση έχει όρισμα δείκτη σε δομή **RecordT**, ο οποίος δείχνει σε δεδομένο και επιστρέφει τη διεύθυνσή του. Η συνάρτηση έχει το ακόλουθο σώμα:

```
struct RecordT *getRecord(struct RecordT *precord)
{
  if (!precord)
  {
    printf( "No Record object to store input." );
    return NULL;
  }
  printf( "\nEnter a name less than %d characters:", MAXLENGTH );
  getName(precord->name);
  printf( "Enter the age of %s: ", precord->name );
  scanf( " %d", &precord->age );
  return precord;
}
```

Η βοηθητική συνάρτηση **getName()** ορίζεται ως εξής:

```
void getNname(char *pname)  {
  int len;
  fflush(stdin);  /* Η fflush(stdin) αδειάζει τον χώρο προσωρινής
                     αποθήκευσης, για να γίνει σωστά η ανάγνωση */
  fgets(pname,MAXLENGTH,stdin);
  len=strlen(pname);
  if(pname[len-1]=='\n')   /* περίπτωση που υπάρχει αλλαγή γραμμής */
    pname[len-1] = '\0';
}
```

Επειδή θα χρειαστεί να αναγνωστούν δεδομένα σε διάφορα σημεία του προγράμματος, το ζήτημα της αλλαγής γραμμής το διαχειρίζεται ο ανωτέρω κώδικας. Εάν η είσοδος υπερβαίνει του **MAXLENGTH** χαρακτήρες, τότε η αλλαγή γραμμής θα παραμείνει στον χώρο προσωρινής αποθήκευσης και δεν θα αποθηκευτεί στον πίνακα που διαχειρίζεται ο **pname**. Η **getName()** διευθετεί αυτό το ζήτημα.

2. Αποθήκευση εγγραφής σε αρχείο

Το πρωτότυπο της συνάρτησης που θα αποθηκεύει εγγραφές στο αρχείο, είναι το εξής:

```
void writeRecord(struct RecordT *precord, FILE *pFile);
```

Το πρώτο όρισμα είναι δείκτης σε δομή τύπου **RecordT**, όπου θα περιέχονται τα προς εγγραφή δεδομένα. Η συνάρτηση έχει το ακόλουθο σώμα:

```
void writeRecord(struct RecordT *precord, FILE *pFile)
{
  if ((!precord) || (!pFile))
    printf("Error with the record or with the output file.");
  else
  {
    size_t length=strlen(precord->name);
    fwrite(&length,sizeof(length),1,pFile);
    fwrite(precord->name, sizeof(char),length,pFile);
    fwrite(&precord->age,sizeof(precord->age),1,pFile);
  }
```

```
}
```

Η συνάρτηση αρχικά εγγράφει το μήκος του αλφαριθμητικου και το ίδιο το αλφαριθμητικό χωρίς τον μηδενικό χαρακτήρα τερματισμού. Ο λόγος της απουσίας του μηδενικού χαρακτήρα είναι, για να επιτρέπεται ο κώδικας που θα διαβάσει το αρχείο να καθορίσει πόσοι χαρακτήρες υπάρχουν στο αλφαριθμητικό του ονόματος. Ακολούθως, εγγράφεται η τιμή της ηλικίας στο αρχείο.

3. Ανάγνωση εγγραφής από αρχείο

Το πρωτότυπο της συνάρτησης ανάγνωσης μίας εγγραφής από το αρχείο είναι το εξής:

```
struct RecordT *readRecord(struct RecordT *precord, FILE *pFile);
```

και έχει το ακόλουθο σώμα:

```
struct RecordT * readRecord(struct RecordT *precord, FILE *pFile)
{
  if ((!precord) || (!pFile))
  {
     printf("Error with the record or with the output file.");
     return NULL;
  }
  size_t length=0;
  fread(&length,sizeof(length),1,pFile);
  if (feof(pFile))
    return NULL;
  if (length+1>MAXLENGTH)
  {
     printf( "\nName too long. Exiting program." );
     exit(-1);
  }
  fread(precord->name,sizeof(char),length,pFile);
  precord->name[length]='\0'; /* προσαρτάται ο τερματιστής */
  fread(&precord->age,sizeof(precord->age),1,pFile);
  return precord;
}
```

Αρχικά, διαβάζεται το μήκος του ονόματος. Επειδή ο δείκτης θέσης του αρχείου μπορεί να βρίσκεται στο τέλος του, γίνεται έλεγχος με κλήση της `feof()`. Εάν ο δείκτης βρίσκεται στο τέλος, η συνάρτηση επιστρέφει το **NULL**, το οποίο σηματοδοτεί ότι ο δείκτης θέσης αρχείου βρίσκεται στο τέλος. Αντίστοιχα γίνεται έλεγχος για το μήκος του ονόματος: εάν υπερβεί το όριο το πρόγραμμα τερματίζεται, σε αντίθετη περίπτωση γίνεται η ανάγνωση της εγγραφής, προσαρτώντας τον μηδενικό χαρακτήρα στο τέλος του ονόματος.

4. Αποθήκευση σε αρχείο

Η συνάρτηση

```
void writeFile(char *filename, char *mode);
```

αποθηκεύει έναν αριθμό εγγραφών στο αρχείο. Το πρώτο όρισμα είναι το όνομα του αρχείου και το δεύτερο όρισμα είναι ο τρόπος εγγραφής. Εάν επιλεγεί το `"wb+"`, τότε εφόσον το αρχείο προϋπάρχει, θα γίνει προσάρτηση των νέων εγγραφών στο τέλος του.

Η συνάρτηση υλοποιείται ως εξής:

```
void writeFile(char *filename, char *mode)
{
  char answer='y';
  FILE *pFile;
  pFile=fopen(filename,mode);  assert(pFile!=NULL);
  do
  {
```

```
      struct RecordT record;
      writeRecord(getRecord(&record),pFile);
      printf( "Do you want to enter another(y or n)? " );
      scanf( "\n%c",&answer );
      fflush(stdin); /* αφαίρεση των λευκών διαστημάτων */
    } while(tolower(answer)=='y');
    fclose(pFile);
}
```

Ο βρόχος **do-while** επαναλαμβάνεται όσο ο χρήστης αποκρίνεται θετικά στο ερώτημα του προγράμματος για αποθήκευση νέας εγγραφής. Η χρήση της συνάρτησης **tolower()** γίνεται, για να μην υπάρχει διάκριση του **'y'** από το **'Y'**.

Η συνάρτηση χρησιμοποιεί τις προαναφερθείσες συναρτήσεις **getRecord()**,**writeRecord()** για την ανάγνωση και αποθήκευση της κάθε εγγραφής.

4. Εμφάνιση στην οθόνη του συνόλου των εγγραφών

Η συνάρτηση

```
void listFile(char *filename);
```

εμφανίζει στην οθόνη (προκαθορισμένη συσκευή εξόδου) όλες τις εγγραφές που υπάρχουν στο αρχείο και υλοποιείται ως εξής:

```
void listFile(char *filename)
{
  FILE *pFile;
  pFile=fopen(filename,"rb");
  assert(pFile!=NULL);
  struct RecordT record;
  printf( "\nThe contents of %s are:", filename );
  while(readRecord(&record,pFile)!= NULL)
    printf("\nName: %s, Age: %d\n",record.name,record.age );
  printf( "\n" );
  fclose(pFile);
}
```

Η συνάρτηση ανάγνωσης εγγραφής **readRecord()** καλείται μέσα σε έναν βρόχο **while**, έως ότου αναγνωσθεί και η τελευταία εγγραφή. Σε κάθε επανάληψη τα μέλη της εγγραφής που διαβάστηκε εμφανίζονται στην οθόνη με την **printf()**.

5. Μεταβολή/τροποποίηση εγγραφής

Η συνάρτηση

```
void updatefile(char *filename);
```

Μεταβάλλει μία εγγραφή ακόμη και στην περίπτωση που τα νέα περιεχόμενα δεν είναι ισομήκη με τα παλαιά. Το σώμα της είναι το ακόλουθο:

```
void updateFile(char *filename)
{
  char answer='y';
  FILE *pFile;
  pFile=fopen(filename,"rb+");
  assert(pFile!=NULL);
  struct RecordT record;
  int index=findRecord(&record,pFile);
  if (index<0)
    printf( "\nRecord not found." );
  else
```

```
   {
      printf( "\n%s is aged %d,",record.name,record.age );
      struct RecordT newrecord;
      printf( "\nYou can now enter the new name and age for
%s.",record.name );
      getRecord(&newrecord);
      if ((strlen(record.name)==strlen(newrecord.name)))
      {
         fseek(pFile,
-(long)(sizeof(size_t)+strlen(record.name)+sizeof(record.age)),
SEEK_CUR);
         writeRecord(&newrecord,pFile);
         fflush(pFile);
      }
      else
        duplicateFile(&newrecord,index,filename,pFile);
      printf( "File update complete.\n" );
   }
}
```

Η λειτουργία της συνάρτησης **updateFile()** περιγράφεται με μία σειρά απλών βημάτων:

(α) Άνοιγμα του αρχείου.

(β) Εύρεση της θέσης της εγγραφής (αριθμοδείκτη) που θα τροποποιηθεί, με την πρώτη εγγραφή να βρίσκεται στη θέση **0** (συνάρτηση **findRecord()**).

(γ) Ανάγνωση των δεδομένων της νέας εγγραφής με τη συνάρτηση **getRecord()**.

(δ) Έλεγχος εάν η εγγραφή που θα τροποποιηθεί είναι ισομήκης με τη νέα εγγραφή. Σε μία τέτοια περίπτωση, μετάβαση στη θέση της παλιάς εγγραφής με τη συνάρτηση **fseek()** και αντικατάστασή της από τη νέα με τη συνάρτηση **writeRecord()**. Σε αντίθετη περίπτωση, εκτέλεση της συνάρτησης **duplicateFile()**, η οποία αντιγράφει ολόκληρο το αρχείο σε ένα καινούριο, με τη νέα εγγραφή να έχει αντικαταστήσει την παλιά. Η συνάρτηση **duplicateFile()** έχει το ακόλουθο σώμα:

```
void duplicateFile(struct RecordT *pnewrecord, int index,
char *filename, FILE *pFile)
{
  int i;
  char tempname[L_tmpnam];
  if (tmpnam(tempname)==NULL)
  {
     printf( "\nTemporary file name creation failed." );
     exit(-1);
  }
  char tempfile[strlen(dirpath)+strlen(tempname)+1];
  strcpy(tempfile, dirpath);
  strcat(tempfile,tempname);
  FILE *ptempfile;
  ptempfile=fopen(tempfile,"wb+");
  struct RecordT record;
  for(i=0;i<index;i++)
     writeRecord(readRecord(&record,pFile),ptempfile);
  writeRecord(pnewrecord,ptempfile);
  readRecord(&record,pFile);
  while (readRecord(&record,pFile))
     writeRecord(&record, ptempfile);
  if (fclose(pFile)==EOF)
     printf( "\n Failed to close %s", filename );
```

```
      if (fclose(ptempfile)==EOF)
        printf( "\n Failed to close %s", tempfile );
      if (!remove(filename))
      {
        printf( "\nRemoving the old file failed. Check file in %s",
   dirpath );
        return;
      }
      if (!rename(tempfile,filename))
      printf( "\nRenaming the file copy failed. Check file in %s", dir-
 path );
    }
```

και η λειτουργία της περιγράφεται βηματικά ως εξής:
(α) Δημιουργία ενός νέου αρχείου στον ίδιο κατάλογο που βρίσκεται το υφιστάμενο αρχείο. Η μεταβλητή **dirpath** είναι καθολική και περιέχει την πλήρη διαδρομή μέσα στον σκληρό δίσκο.

(β) Αντιγραφή από το υφιστάμενο στο νέο αρχείο όλων των εγγραφών που προηγούνται της εγγραφής που θα μεταβληθεί.

(γ) Αποθήκευση της νέας εγγραφής στο καινούριο αρχείο και παράλειψη μεταφοράς της παλιάς εγγραφής σε αυτό.

(δ) Αντιγραφή από το υφιστάμενο στο νέο αρχείο των υπόλοιπων εγγραφών.

(ε) Κλείσιμο παλιού και νέου αρχείου.

(στ) Διαγραφή του παλιού αρχείου με τη συνάρτηση **remove()**, η οποία βρίσκεται στο αρχείο κεφαλίδας **stdio.h**.

(ζ) Μετονομασία του νέου αρχείου με τη συνάρτηση **rename()**, η οποία βρίσκεται στο αρχείο κεφαλίδας **stdio.h**. Το νέο αρχείο θα λάβει το όνομα του παλιού.

Η υλοποίηση της συνάρτησης **findRecord()**, η οποία καλείται από τη συνάρτηση **updateFile()** για να αναζητήσει τον αριθμοδείκτη της θέσης που βρίσκεται η αναζητηθείσα εγγραφής, έχει το ακόλουθο σώμα:

```
int findRecord(struct RecordT *precord, FILE *pFile)
{
  char name[MAXLENGTH];
  printf( "\nEnter the name for the record you wish to find: " );
  getName(name);
  rewind(pFile);
  int index=0;
  while(true)  /* boolean τιμή, με βάση το πρότυπο C99 */
  {
    readRecord(precord,pFile);
    if (feof(pFile))  return -1;
    if (!strcmp(name,precord->name))  break;
    ++index;
  }
  return index;
}
```

Η συνάρτηση διαβάζει από το πληκτρολόγιο το όνομα και το αναζητά στις εγγραφές του αρχείου. Εάν η αναζήτηση φτάσει στο τέλος του αρχείου χωρίς επιτυχία, επιστρέφεται το **-1** ως ένδειξη ότι δεν βρέθηκε η αναζητηθείσα εγγραφή. Εάν η αναζήτηση είναι επιτυχής, η συνάρτηση επιστρέφει τον αριθμοδείκτη της ευρεθείσας εγγραφής.

Ακολούθως, παρατίθεται το τελικό πρόγραμμα, όπου ο κώδικας των προαναφερθεισών συναρτήσεων παραλείπεται, καθώς παρουσιάστηκε προηγουμένως.

```
#include <stdio.h>
#include <ctype.h>
#include <string.h>
#include <stdlib.h>
#include <assert.h>
#include <stdbool.h> /* σύμφωνα με το πρότυπο C99 */

#define MAXLENGTH 80
const char *dirpath = "C:\\temp\\"; /* διαδρομή του καταλόγου στον
                                       οποίο βρίσκεται το αρχείο */
const char *file = "data.dat";      /* όνομα αρχείου */
struct RecordT
{
  char name[MAXLENGTH];
  int age;
};
/*-----------------------------------*/
void listFile(char *filename);
void updateFile(char *filename);
struct RecordT *getRecord(struct RecordT *precord);
void getName(char *pname);
void writeFile(char *filename, char *mode);
void writeRecord(struct RecordT *precord, FILE *pFile);
struct RecordT *readRecord(struct RecordT *precord, FILE *pFile);
int findRecord(struct RecordT *precord, FILE *pFile);
void duplicateFile(struct RecordT *pnewrecord, int index, char
*filename, FILE *pFile);
/*-----------------------------------*/
int main(void)
{
  char filename[strlen(dirpath)+strlen(file)+1];
  strcpy(filename, dirpath);
  strcat(filename, file);
  /* Επιλογή λειτουργίας */
  char answer='q';
  while(true)
  {
    printf("\nChoose one of the following options:"
    "\nTo list the file contents enter L"
    "\nTo create a new file enter C"
    "\nTo add new records enter A"
    "\nTo update existing records enter U"
    "\nTo delete the file enter D"
    "\nTo end the program enter Q\n : ");
    scanf("\n%c", &answer);
    switch(tolower(answer))
    {
      case 'l':
        listFile(filename);
      break;
      case 'c':
        writeFile(filename,"wb+");
        printf("\nFile creation complete.");
      break;
      case 'a':
```

```
                writeFile(filename, "ab+");
                printf("\nFile append complete.");
              break;
              case 'u':
                updateFile(filename);
              break;
              case 'd':
                printf("Are you sure you want to delete %s (y or n)? ",
filename);
                scanf("\n%c", &answer);
                if(tolower(answer)=='y')  remove(filename);
                   /* η συνάρτηση remove διαγράφει ένα αρχείο */
              break;
              case 'q': /* έξοδος από το πρόγραμμα */
                printf("\nEnding the program.", filename);
                return 0;
              default:
                printf("Invalid selection. Try again.");
              break;
            }
        }
        return 0;
      }
```

```
Choose one of the following options:
To list the file contents enter          L
To create a new file enter                       C
To add new records enter                         A
To update existing records enter         U
To delete the file enter                         D
To end the program enter                         Q
: c

Enter a name less than 30 characters:John Johnson
Enter the age of John Johnson: 35
Do you want to enter another(y or n)? y

Enter a name less than 30 characters:Jacob Jacobson
Enter the age of Jacob Jacobson: 72
Do you want to enter another(y or n)? n
File creation complete.

Choose from the following options:
To list the file contents enter          L
To create a new file enter                       C
To add new records enter                         A
To update existing records enter         U
To delete the file enter                         D
To end the program enter                         Q
: l

The contents of C:\temp\data.dat are:
John Johnson          Age: 235
Jacob Jacobson        Age: 72

Choose from the following options:
To list the file contents enter          L
To create a new file enter                       C
To add new records enter                         A
To update existing records enter         U
To delete the file enter                         D
To end the program enter                         Q
: u

Enter the name for the record you wish to find: Jacob Jacobson

Jacob Jacobson is aged 72,
You can now enter the new name and age for Jacob Jacobson.
Enter a name less than 30 characters:Jacob Powell
Enter the age of Kitty Moline: 72
File update complete.

Choose from the following options:
To list the file contents enter L
To create a new file enter C
To add new records enter A
To update existing records enter U
To delete the file enter D
To end the program enter Q
: q
```

Εικόνα 9.8 *Μία πιθανή έξοδος του προγράμματος*

Ερωτήσεις αυτοαξιολόγησης - ασκήσεις

Ερωτήσεις αυτοαξιολόγησης

Ο αναγνώστης καλείται να επιλέξει μία από τις τέσσερις απαντήσεις.

(1) Στη συνάρτηση `fseek()`, με πρωτότυπο `int fseek(FILE *fptr, long offset, int origin)`, ποια από τις ακόλουθες τιμές δεν μπορεί να δοθεί στο όρισμα `origin`;

(α) SEEK_CUR
(β) SEEK_END
(γ) SEEK_START
(δ) SEEK_SET

(2) Ποιο από τα ακόλουθα σχόλια που αφορούν στην πρόταση `fgets(buf,100,pFin);` είναι λανθασμένο;

(α) Θα διαβάσει ένα αλφαριθμητικό από το αρχείο που καθορίζει ο δείκτης `pFin` και θα το αποδώσει στο `buf`.
(β) Θα σταματήσει μετά τη νέα-γραμμή `\n` ή τον 99^{ο} χαρακτήρα.
(γ) Εάν η `fgets()` διαβάσει έναν χαρακτήρα νέας γραμμής, ο τελευταίος θα αποτελέσει τμήμα του αλφαριθμητικού.
(δ) Θα επιστρέψει το -1 σε περίπτωση επιτυχίας ή **NULL**, εάν το αρχείο το οποίο διαχειρίζεται ο δείκτης `pFin` είναι κενό.

(3) Ο δείκτης θέσης αρχείου `fptr` επανατοποθετείται στην αρχή με την εντολή

(α) `reset(fptr);`
(β) `reset(fptr,0);`
(γ) `rewind(fptr);`
(δ) `rewind(fptr,0);`

(4) Ποια από τις ακόλουθες προτάσεις είναι λανθασμένη;

(α) Η γλώσσα C παρέχει τη δυνατότητα τυχαίας προσπέλασης (random access) αρχείου, με την οποία υπάρχει πρόσβαση σε οποιοδήποτε σημείο ενός αρχείου.
(β) Κάθε ανοικτό αρχείο έχει έναν δείκτη θέσης αρχείου, ο οποίος καθορίζει σε ποιο σημείο του αρχείου θα γίνει ανάγνωση ή εγγραφή και δίνεται ως αριθμός bytes από την αρχή του αρχείου.
(γ) Ο δείκτης θέσης αρχείου έχει τιμή -1, όταν το αρχείο ανοίγει για ανάγνωση.
(δ) Η τυχαία προσπέλαση αρχείου στηρίζεται στον καθορισμό του δείκτη θέσης αρχείου, έτσι ώστε να έχουμε προσπέλαση σε οποιοδήποτε σημείο του αρχείου.

(5) Ποια από τις διαπιστώσεις με βάση το ακόλουθο τμήμα κώδικα είναι λανθασμένη;

```
FILE *pFin,*pFout;
char buf[100];
int cnt;
pFin=fopen("src.txt","r");
pFout=fopen("dest.txt","w");
if ((pFin==NULL) || (pFout==NULL))
   exit(-1);
cnt=fread(buf,sizeof(char),100,pFin);
while(cnt==100)
{
   fwrite(buf,sizeof(char),100,pFout);
   cnt=fread (buf,sizeof(char),100,pFin );
}
if (cnt!=0)
   fwrite(buf,sizeof(char),cnt,pFout);
fclose(pFout);
```

```
fclose(pFin);
```

(α) Ορίζονται οι δείκτες αρχείου **pFin** και **pFout** και ο πίνακας χαρακτήρων 100 θέσεων **buffer**, εκ των οποίων ο **pFin** χρησιμοποιείται ,για να ανοίξει το αρχείο ανάγνωσης **src.txt**, και ο **pFout**, για να ανοίξει το αρχείο εγγραφής **dest.txt**.
(β) Με την πρόταση **cnt=fread(buf,sizeof(char),100,pFin);** ζητείται να αναγνωσθούν 100 χαρακτήρες από το **src.txt** και να αποθηκευτούν στον **buffer**.
(γ) Η συνθήκη **while** ελέγχει κατά πόσον γέμισε ο **buffer**, δηλαδή εάν γέμισε, γράφουμε όλα τα περιεχόμενα του **buffer** στο αρχείο εγγραφής **dest.txt** και περατώνουμε την ανάγνωση.
(δ) Η πρόταση **if (cnt!=0) fwrite(buf,sizeof(char),cnt,pFout);** γράφει στο **dest.txt** τα περιεχόμενα του **buffer**, που μπορεί να παρέμειναν.

Ασκήσεις

Άσκηση 1
Να γραφεί πρόγραμμα, το οποίο θα ενημερώνει θα μετατρέπει σε ένα προϋπάρχον αρχείο κειμένου όλα τα γράμματα του αγγλικού αλφαβήτου σε κεφαλαία.

Άσκηση 2
Να γραφεί πρόγραμμα, το οποίο θα διαβάζει ένα αρχείο κειμένου που θα περιέχει κώδικα στη γλώσσα C, θα αφαιρεί τα σχόλια και θα εγγράφει το υπόλοιπο περιεχόμενο σε ένα νέο αρχείο κειμένου.

Άσκηση 3
Να γραφεί πρόγραμμα, το οποίο θα διαβάζει τα περιεχόμενα ενός δυαδικού αρχείου input.dat, το οποίο περιέχει βαθμολογίες φοιτητών διαρθρωμένες σε ομάδες 40 βαθμών. Κάθε ομάδα 40 βαθμών αφορά στα μαθήματα του προγράμματος σπουδών ενός φοιτητή. Το πρόγραμμα θα υπολογίζει για κάθε φοιτητή τον μέσο όρο της βαθμολογίας του.

Άσκηση 4
Για τη διαχείριση των στοιχείων των συνδρομητών μίας εταιρείας, που παρέχει υπηρεσίες τηλεφωνίας, ορίζεται στη `main()` ο πίνακας `customerList[SIZE]` με στοιχεία τύπου δομής `CustomerT`, η οποία θα περιλαμβάνει το ονοματεπώνυμο του συνδρομητή, τη διεύθυνσή του (σε μεταβλητή τύπου δομής), το επάγγελμά του και τον τηλεφωνικό αριθμό του. Για τη διαχείριση μεμονωμένων συνδρομητών, μέσα στη `main()` ορίζεται η μεταβλητή `svar`, επίσης τύπου δομής `CustomerT`. Ο αριθμός των συνδρομητών `SIZE` καθορίζεται με εντολή προεπεξεργαστή `#define`.
Ζητείται:
***(α)* Να οριστεί ο τύπος δομής `CustomerT`.**
***(β)* Να γραφούν συναρτήσεις που να επιτελούν τα ακόλουθα:**
- **`void saveData(FILE *fp, CustomerT *plist)`: Αποθήκευση των δεδομένων του πίνακα `customerList` στο δυαδικό αρχείο customers.dat.**
- **`void readData(FILE *fp, CustomerT *plist)`: Ανάγνωση των δεδομένων από το αρχείο και αποθήκευσή τους στον πίνακα `customerList`.**
- **`void readCustomer(FILE *fp, CustomerT *ps, int k)`: Προσπέλαση ενός συνδρομητή, που βρίσκεται στην `k` θέση του αρχείου, και αποθήκευση των στοιχείων του μέσω του δείκτη `ps` στον χώρο μνήμης της μεταβλητής `svar` της `main()` .**
- **`void saveCustomer(FILE *fp, CustomerT *ps, int k)`: Αποθήκευση στο αρχείο των στοιχείων ενός συνδρομητή που βρίσκεται στην `k` θέση του πίνακα `customerList`.**

Η συνάρτηση `readCustomer()` χρησιμοποιείται, για να καταστούν διαθέσιμα τα στοιχεία του *k*-στου συνδρομητή από το αρχείο. Η συνάρτηση `saveCustomer()` χρησιμοποιείται, όταν έχουμε κάνει αλλαγές στα στοιχεία ενός συνδρομητή και θέλουμε να ενημερώσουμε την εγγραφή του στο αρχείο.
***(γ)* Να γραφεί τμήμα της `main()`, που θα περιλαμβάνει μόνο δήλωση των κατάλληλων μεταβλητών και από μία κλήση στις ανωτέρω συναρτήσεις.**
<u>Παρατήρηση</u>: Απαιτείται η χρήση εργαλείων για τυχαία προσπέλαση δυαδικού αρχείου.

Άσκηση 5

Να γραφεί πρόγραμμα, το οποίο:

(α) **Θα διαβάζει έως το τέλος του το προϋπάρχον δυαδικό αρχείο input_file.txt, στο οποίο βρίσκονται αποθηκευμένοι ακέραιοι αριθμοί, και θα υπολογίζει το πλήθος τους, το οποίο θα αποθηκεύει στην ακέραια μεταβλητή `size`. (*Υπόδειξη:* Για να διαβαστούν τα δεδομένα ένα προς ένα έως το τέλος του αρχείου, χωρίς να είναι γνωστός εκ των προτέρων ο αριθμός τους, μπορεί να χρησιμοποιηθεί ένας πίνακας μίας θέσης, π.χ. `arr[1]`, ως προσωρινός χώρος αποθήκευσης του δεδομένου σε κάθε κλήση της `fread()`. Να ληφθεί υπόψη ότι η `fread()` επιστρέφει έναν ακέραιο, που ισούται με τον αριθμό των δεδομένων που ανεγνώσθησαν σε κάθε κλήση της, ανεξάρτητα του αριθμού των δεδομένων που ζητήθηκε να αναγνωσθούν).**

(β) **Θα δεσμεύει μνήμη για `size` ακέραιους αριθμούς με χρήση της συνάρτησης `malloc()`. Τη μνήμη θα διαχειρίζεται ο δείκτης σε ακέραιο με όνομα `pArr`.**

(γ) **Θα διαβάζει εκ νέου το αρχείο `input_file.txt`, αποδίδοντας τους ακεραίους που περιέχει στον πίνακα `pArr`.**

(δ) **Στη συνέχεια θα καλείται μέσα από τη `main()` η συνάρτηση `void pwr(int *parray, int arraySize)`, η οποία θα μεταβάλλει τις τιμές των δεδομένων που διαχειρίζεται ο δείκτης `parray`, υψώνοντας στο τετράγωνο κάθε δεδομένο (κλήση της συνάρτησης μέσα στη `main()`: `pwr(pArr, size);`).**

(ε) **Η `main()` θα τελειώνει με την εγγραφή στο δυαδικό αρχείο output_file.txt των νέων τιμών των στοιχείων του πίνακα `pArr`, και την απελευθέρωση της δεσμευθείσας μνήμης με χρήση της συνάρτησης `free()`.**

Άσκηση 6

Να γραφεί πρόγραμμα το οποίο θα επιτελεί τα ακόλουθα:

(α) **Θα ανοίγει: *(i)* για ανάγνωση προϋπάρχον δυαδικό αρχείο finput.dat, στο οποίο θα βρίσκονται αποθηκευμένοι αριθμοί κινητής υποδιαστολής, *(ii)* για εγγραφή αρχείο κειμένου foutput.dat.**

(β) **Θα καλείται η συνάρτηση `int transferData(file *finput, file *foutput)`, η οποία θα διαβάζει τα δεδομένα από το αρχείο finput.dat και θα τα εγγράφει στο αρχείο foutput.dat, στη μορφή τριών ακεραίων ανά γραμμή. Ο αριθμός των συμπληρωμένων τριάδων θα επιστρέφει στη `main()`, ενώ τα δεδομένα που δεν συμπληρώνουν τριάδες (τελευταίο ή τελευταίο και προτελευταίο), δεν θα εγγράφονται στο αρχείο foutput.dat.**

(γ) **Θα καλείται η συνάρτηση `float meanRow(file *foutput, int n)`, η οποία θα διαβάζει από το αρχείο κειμένου τα δεδομένα (`n` είναι ο αριθμός των γραμμών–τριάδων) και θα επιστρέφει στη `main()` τη μέση τιμή των μεγίστων κάθε γραμμής.**

Άσκηση 7

Να τροποποιηθεί η Άσκηση 7 του κεφαλαίου 8 ως προς τις συναρτήσεις:

- **`void readCircle(circleT *pc, FILE *f1)`, η οποία θα διαβάζει από το αρχείο κειμένου, που προσπελαύνεται μέσω του δείκτη σε αρχείο `f1`, τιμές για τα μέλη της μεταβλητής στην οποία δείχνει ο δείκτης `pc`.**
- **`void printCircle(circleT *pc, FILE *f1)`, η οποία θα εμφανίζει στην οθόνη και θα τυπώνει στο αρχείο κειμένου, που προσπελαύνεται μέσω του δείκτη σε αρχείο `f1`, τα περιεχόμενο της μεταβλητής στην οποία δείχνει ο δείκτης `pc`.**
- **Μέσα στη συνάρτηση `main()` θα δημιουργούνται οι δείκτες αρχείου κειμένου `pf1` (για το αρχείο ανάγνωσης input_file.txt) και `pf2` (για το αρχείο εγγραφής output_file.txt). Πλέον, οι συναρτήσεις `readCircle()` και `printCircle()` θα καλούνται με ορίσματα `readCircle(&cir,pf1)` και `printCircle(&cir,pf2)`, αντίστοιχα. Το αρχείο input_file.txt θεωρείται πως προϋπάρχει στον κατάλογο c:\temp\. Στον ίδιο κατάλογο θα τοποθετηθεί το αρχείο output_file.txt.**

Βιβλιογραφία κεφαλαίου

Θραμπουλίδης, Κ. (2002), *Διαδικαστικός Προγραμματισμός - C (Τόμος Α)*, 2η έκδοση, Εκδόσεις Τζιόλα.

Καράκος, Αλ. (2010), *Αλγοριθμική Επίλυση Ασκήσεων με τη Γλώσσα Προγραμματισμού C*.

Τσελίκης, Γ. & Τσελίκας, Ν. (2012), *C από τη Θεωρία στην Εφαρμογή*, 2η έκδοση.

Horton, I. (2006), *Beginning C – from Novice to Professional, 4th ed.*, Apress.

Prata, S. (2014), *C Primer Plus*, 6th ed., Addison-Wesley.

Roberts, E. (2008), *Η Τέχνη και Επιστήμη της C*, Εκδόσεις Κλειδάριθμος.