Δομές & Αρχεία στην C

ΔΟΜΕΣ (structures) [1/2]

- Ο τρόπος αναπαράστασης πολύπλοκων ή/και σύνθετων δεδομένων στη C είναι μέσω μιας ευέλικτης μορφής που ονομάζεται δομή. Οι δομές είναι πεπερασμένες συλλογές μεταβλητών, διαφόρων τύπων στη γενική περίπτωση, και οι οποίες ομαδοποιούνται υπό ένα και μοναδικό όνομα. Η πρόσβαση στις μεταβλητές αυτές γίνεται μέσω του ονόματος της δομής. Η γενική μορφή μιας δομής στη C είναι η εξής:

```
struct όνομα δομής {
        τύπος στοιχείο1;
        τύπος στοιχείο2;
        . . .
        τύπος στοιχείοN;
};
```

όπου τα "στοιχεία" μπορεί να είναι απλές μεταβλητές, κατασκευές (π.χ. πίνακες, συμβολοσειρές), δείκτες ή δομές.

ΔΟΜΕΣ (structures) [2/2]

- Στο παρακάτω παράδειγμα, η δομή book,

```
struct book {
        int serial_nr; /* 1ο πεδίο */
        char title[SIZE]; /* 2ο πεδίο */
        char author[SIZE]; /* 3ο πεδίο */
        float value; /* 4ο πεδίο */
};
```

- προσδιορίζει τέσσερα πεδία για τις μεταβλητές και συμβολοσειρές: serial_nr (τύπου int), title και author (τύπου πίνακα από char) και value (τύπου float). Το πρώτο πεδίο είναι για τον αύξοντα αριθμό, τα επόμενα δύο πεδία είναι για τον τίτλο του βιβλίου και το όνομα του πρώτου συγγραφέα αντίστοιχα και το τέταρτο πεδίο είναι για την τιμή κοστολόγησης.
- Μπορούμε να ερμηνεύσουμε τον ορισμό μιας δομής με τον καθορισμό ενός δικού μας (σύνθετου ή μικτού) τύπου δεδομένων. Έτσι, μπορούμε να ορίσουμε και μεταβλητές αυτού του τύπου, όπως, για παράδειγμα, με τη δήλωση:

struct book book1, book2, oldbooks[100], *bkptr;

τύπος μεταβλητές δομής πίνακας δομών pointer σε δομή

ΔΕΣΜΕΥΣΗ ΜΝΗΜΗΣ

- Κατά τη δήλωση των μεταβλητών, δεσμεύεται και κατάλληλος χώρος στη μνήμη του υπολογιστή ανάλογα με τον ορισμό των πεδίων της δομής. Έτσι, υποθέτοντας ότι ο τύπος int απαιτεί 4 bytes μνήμης, για κάθε μια από τις **book1** και **book2**, θα δεσμευθούν

  1 x sizeof (int) + 2 x SIZE x sizeof (char) + 1 x sizeof (float) = 2 x (SIZE + 4) bytes.
- Αντίστοιχα, για τον πίνακα **oldbooks** θα δεσμευθούν **200 x (SIZE + 4) bytes** και για τον pointer **bkptr** θα δεσμευθούν **4 bytes**.
- Επίσης, μπορείτε να δείτε τον χώρο μνήμης που δεσμεύεται μέσω του τελεστή **sizeof** (π.χ., η **sizeof (book)** δίνει το μέγεθος της **book** σε bytes).

- *Αν δεν συμφωνεί με τον παραπάνω αναλυτικό υπολογισμό, αυτό θα οφείλεται στο ότι εμείς υποθέσαμε byteaddressable σύστημα ενώ στην ουσία πρόκειται για word-addressable σύστημα. Π.χ. για words των 4 bytes και SIZE = 27, στη μεν περίπτωση byte addressable συστήματος οι book1 και oldbooks δεσμεύουν 62 bytes και 6200 bytes αντίστοιχα ενώ στην περίπτωση word addressable συστήματος δεσμεύουν 64 bytes και 6400 bytes αντίστοιχα (ο συνολικός χώρος που δεσμεύεται για κάθε μεταβλητή ή στοιχείο πίνακα πρέπει να είναι σε πολλαπλάσιο των 4 bytes).*

ΠΡΟΣΠΕΛΑΣΗ ΣΤΑ ΠΕΔΙΑ ΔΟΜΗΣ [1/3]

- Η προσπέλαση στα πεδία μιας δομής γίνεται μέσω του τελεστή μέλους δομής '.'.
- Για παράδειγμα, η εκχώρηση τιμών στα πεδία της **book1** με βάση τα στοιχεία του βιβλίου

1. "The C programming Language", "Kernighan-Ritchie", €30.00
μπορεί να γίνει με τον εξής τρόπο:

```
book1.serial_nr = 1;
strcpy(book1.title, " The C programming Language ");
/* Αν ο δοθεί από το πληκτρολόγιο τότε: gets(book1.title); */
strcpy(book1.author, " Kernighan-Ritchie ");
book1.value = 30.00;
```

ΠΡΟΣΠΕΛΑΣΗ ΣΤΑ ΠΕΔΙΑ ΔΟΜΗΣ [2/3]

```
...
int main() {
        int count = 0;
        char answer[5];

        struct book {
                int serial_nr;
                char title[81];
                char author[81];
                float value;
        } oldbooks[100]; /* Ταυτόχρονος ορισμός δομής και πίνακα */

        printf("Do you want to enter a new book? (yes/no) : ");
        while( gets(answer) != NULL && !strcmp(answer,"yes") && count < 100 ) {
                oldbooks[count].serial_nr = count + 1;
                printf("Titlos bibliou : ");
                gets(oldbooks[count].title);
                printf("Prwtos syggrafeas : \n");
                gets(oldbooks[count].author);
                printf("Axia bibliou se Euro : \n");
                scanf("%f",&oldbooks[count].value);
                fflush(stdin);  /* Άδειασε τον buffer */
                count++;
                printf("Do you have another book? (yes/no) :\n");
        }
        return 0;
}
```

ενώ το διάβασμα των στοιχείων 100 βιβλίων στον πίνακα oldbooks θα μπορούσε, για παράδειγμα, να γίνει με το δίπλα πρόγραμμα:

ΠΡΟΣΠΕΛΑΣΗ ΣΤΑ ΠΕΔΙΑ ΔΟΜΗΣ [3/3]

- Στο παραπάνω πρόγραμμα fflush(stdin) χρησιμοποιείται για να αδειάζουμε τον buffer από χαρακτήρες που άφησε η scanf() (ό,τι ακολουθεί στη γραμμή εισόδου μέχρι και τον χαρακτήρα '\n') πριν διαβάσουμε την επόμενη είσοδο με gets().
- Επίσης, παρατηρήστε ότι αφού η strcmp() επιστρέφει την τιμή 0 όταν οι συμβολοσειρές είναι ίδιες, η !strcmp(answer,"yes") θα επιστρέψει 1 αν πληκτρολογήσαμε "yes".
- Ο πίνακας **oldbooks** έχει σαν στοιχεία τα **oldbooks[i]** (i = 0, 1, ..., 99). Το κάθε ένα από αυτά τα στοιχεία είναι μια δομή τύπου **struct book**. Κατά συνέπεια, μπορούμε να προσπελάσουμε τα πεδία (ή μέλη) της **oldbooks[i]** με τον τελεστή '.' όπως και για τις απλές μεταβλητές δομής.
- Έτσι, για παράδειγμα, η αξία του 15ου βιβλίου θα βρίσκεται στο **oldbooks[14].value** (***Προσοχή****: όχι oldbooks.value[14]*) με διεύθυνση στη μνήμη την &oldbooks[14].value. Αντίστοιχα, η διεύθυνση μνήμης στην οποία βρίσκεται αποθηκευμένη η δομή oldbooks[14] θα είναι η &oldbooks[14]. Οι ομοιότητες με τους κλασικούς πίνακες είναι προφανείς. Ο πίνακας δομών είναι απλώς ένας γενικευμένος πίνακας στον οποίο αποθηκεύουμε διαφόρων τύπων δεδομένα.

ΟΡΙΣΜΟΣ ΔΟΜΗΣ: ΜΕ ΟΝΟΜΑ Ή ΧΩΡΙΣ ΟΝΟΜΑ

- Συνήθως, παράλληλα με τη δήλωση μιας δομής της δίνουμε και ένα όνομα (βλ. Για παράδειγμα τη δήλωση της δομής **struct book**). Έχοντας δώσει όνομα στη δομή, μπορούμε στη συνέχεια να διακηρύξουμε μεταβλητές δομής με δηλώσεις της μορφής:

```
struct book book1;
```

- Πολλές φορές, δηλώνουμε μια δομή εξωτερικά ώστε να φαίνεται από όλες τις συναρτήσεις ενώ ορίζουμε τοπικές μεταβλητές αυτής της δομής στις συναρτήσεις που τις χρησιμοποιούν. Αν ο ορισμός της δομής και των μεταβλητών γίνεται στο ίδιο σημείο του προγράμματος, τότε μπορούμε να αποφύγουμε να δώσουμε όνομα στη δομή. Στην περίπτωση αυτή έχουμε ταυτόχρονα ορισμό δομής (χωρίς όνομα) και μεταβλητών. Παράδειγμα:

```
struct { /* Δεν υπάρχει όνομα δομής */
        int serial_nr;
        char title[SIZE];
        char author[SIZE];
        float value;
} oldbooks[100], book1, book2, *bkptr;
```

ΔΕΙΚΤΕΣ ΣΕ ΔΟΜΕΣ [1/2]

```
#include <stdio.h>
#define LEN 80
struct CD {
                char band[LEN];
                char album[LEN];
                int year;
};

int main() {
                struct CD rock[2] = {
                                { "Joe Cocker", "Greatest Hits", 1998 },
                                { "Bruce Springsteen", "The Rising", 2002 }
                };

struct CD *cdptr; /* Pointer σε δομή */
printf("&rock[0] = %u, &rock[1] = %u\n",&rock[0],&rock[1]);

cdptr = &rock[0];
printf("cdptr = %u, cdptr+1 = %u\n",cdptr,cdptr+1);
printf("cdptr->year is %d, (*cdptr).year is %d\n",
cdptr->year, (*cdptr).year);
cdptr++;
printf("BAND: %s - ALBUM: %s – YEAR: %d\n",
cdptr->band, cdptr->album, cdptr->year);
return 0;
}
```

```
Έξοδος προγράμματος:
&rock[0] = 4235424, &rock[1] = 4235592
cdptr = 4235424, cdptr+1 = 4235592
cdptr->year is 1998, (*cdptr).year is 1998
BAND: Bruce Springsteen - ALBUM: The Rising – YEAR: 2002
```

```
#include <stdio.h>
#define LEN 80
struct CD {
                char band[LEN];
                char album[LEN];
                int year;
};

int main() {
                struct CD rock[2] = {
                                { "Joe Cocker", "Greatest
Hits", 1998 },
                                { "Bruce Springsteen", "The
Rising", 2002 }
                };

struct CD *cdptr; /* Pointer σε δομή */
printf("&rock[0] = %u, &rock[1] = %u\n",&rock[0],&rock[1]);

cdptr = &rock[0];
printf("cdptr = %u, cdptr+1 = %u\n",cdptr,cdptr+1);
printf("cdptr->year is %d, (*cdptr).year is %d\n",
cdptr->year, (*cdptr).year);
cdptr++;
printf("BAND: %s - ALBUM: %s – YEAR: %d\n",
cdptr->band, cdptr->album, cdptr->year);
return 0;
}
```

```
Έξοδος προγράμματος:
&rock[0] = 4235424, &rock[1] = 4235592
cdptr = 4235424, cdptr+1 = 4235592
cdptr->year is 1998, (*cdptr).year is 1998
BAND: Bruce Springsteen - ALBUM: The Rising – YEAR: 2002
```

ΔΕΙΚΤΕΣ ΣΕ ΔΟΜΕΣ [2/2]

- Όταν αυξάνουμε κατά ένα τον δείκτη, η διεύθυνσή του αυξάνεται κατά το πλήθος των bytes που απαιτούνται για μια δομή. Έτσι, επειδή τα στοιχεία του πίνακα είναι δομές που η κάθε μία απαιτεί 80 + 80 + 4 + 4 = 168 bytes, η διεύθυνση του cdptr+1 θα είναι: 4235424 +168 = 4235592.
- Η C παρέχει δυνατότητα προσπέλασης των μελών μια δομής απ'ευθείας από δείκτη στη δομή. Αυτό γίνεται μέσω του τελεστή '->'. Με άλλα λόγια, ένας δείκτης σε δομή που ακολουθείται από τον τελεστή '->' λειτουργεί με τον ίδιο ακριβώς τρόπο που μια μεταβλητή δομής ακολουθείται από τον τελεστή '.'. Στο παραπάνω πρόγραμμα διαπιστώσαμε ότι

  cdptr->year == **(*cdptr).year**
- κάτι που μπορεί να εξηγηθεί από το γεγονός ότι ενώ ο cdptr είναι ένας δείκτης σε δομή, το *cdptr θα είναι στην ουσία η ίδια η (σύνθετη) θέση μνήμης των 168 bytes που αντιστοιχεί στη δομή, κατ'αναλογία με τους δείκτες σε μεταβλητές.
- Οι παρενθέσεις στο **(*cdptr).year** είναι απαραίτητες καθώς ο τελεστής '.' έχει μεγαλύτερη προτεραιότητα από τον '*'. Συνεπώς, το ***cdptr.year** ερμηνεύεται ως ***(cdptr.year)** με το τελευταίο να μην έχει νόημα.

ΔΟΜΕΣ ΚΑΙ ΣΥΝΑΡΤΗΣΕΙΣ

- Γνωρίζουμε ήδη ότι από τα ορίσματα των συναρτήσεων περνάμε τιμές στη συνάρτηση. Οι τιμές αυτές μπορεί να είναι, για παράδειγμα, τύπου int, float ή ακόμη και διευθύνσεις. Λόγω, όμως, της πολυπλοκότητας μιας δομής, σε κάποιους μεταγλωττιστές δεν επιτρέπεται οι δομές να χρησιμοποιούνται ως ορίσματα σε συναρτήσεις (ορισμένοι compilers μπορεί να δέχονται δομές ως ορίσματα), κάτι που άλλωστε συμβαίνει και με τους πίνακες. Πάντως, ο Dev-C++ επιτρέπει ορίσματα συναρτήσεων τύπου δομής. Προκειμένου ο πηγαίος κώδικας να είναι μεταφέρσιμος (δηλαδή να μην εξαρτάται από τον μεταγλωττιστή) συστήνεται η χρήση δεικτών σε δομές ως ο ενδεδειγμένος τρόπος επικοινωνίας με συναρτήσεις (εννοείται ότι οι τιμές στα πεδία μιας δομής μπορούν να δοθούν ως ορίσματα συναρτήσεων κατά τον συνήθη τρόπο).

ΑΡΧΕΙΑ (Files) [1/2]

- Ως αρχείο θεωρούμε μια πλήρη συλλογή πληροφοριών στην οποία έχει δοθεί κάποιο όνομα, όπως ένα πρόγραμμα, ένα σύνολο δεδομένων που χρησιμοποιείται από ένα πρόγραμμα, ή ένα έγγραφο δημιουργημένο από το χρήστη. Το αρχείο είναι η βασική μονάδα αποθήκευσης, συνήθως στον δίσκο, που επιτρέπει στον υπολογιστή να διακρίνει ένα σύνολο πληροφοριών από ένα άλλο.
- Τυπικά, ένα C-πρόγραμμα δέχεται είσοδο από τη βασική συσκευή εισόδου (standard input). Αν και ως βασική συσκευή εισόδου μπορεί να προεπιλεγεί, για παράδειγμα, μαγνητική ταινία, διάτρητες κάρτες, τηλέτυπο, κ.λπ., η πιο συνήθης επιλογή είναι, φυσικά, το πληκτρολόγιο.
- Εναλλακτικά, η είσοδος σε ένα πρόγραμμα μπορεί να προέρχεται από κάποιο αρχείο.

ΑΡΧΕΙΑ (Files) [2/2]

- Υπάρχουν δύο τρόποι για να κάνουμε ένα πρόγραμμα να χρησιμοποιήσει αρχεία. Ο πρώτος (και πιο εύκολος τρόπος) είναι να χρησιμοποιήσουμε ένα πρόγραμμα που δέχεται είσοδο από το πληκτρολόγιο (standard input) και στέλνει την έξοδό του στην οθόνη (standard output) αλλά να το υποχρεώσουμε να ανακατευθύνει την είσοδο και την έξοδο σε διαφορετικά κανάλια, π.χ. σε αρχεία. Η μέθοδος αυτή ονομάζεται α**νακατεύθυνση εισόδου/εξόδου** (I/O redirection).
- Ο δεύτερος τρόπος είναι να χρησιμοποιήσουμε ειδικές συναρτήσεις που μας επιτρέπουν να ανοίγουμε αρχεία, να κλείνουμε αρχεία, να διαβάζουμε αρχεία, να αποθηκεύουμε σε αρχεία, να τροποποιούμε αρχεία, κ.λπ. Άλλες είναι οι συναρτήσεις εισόδου/εξόδου που αφορούν τα ASCII ή text αρχεία και άλλες αυτές που αφορούν τα δυαδικά (binary) αρχεία.

ΑΝΑΚΑΤΕΥΘΥΝΣΗ ΕΙΣΟΔΟΥ/ΕΞΟΔΟΥ [1/2]

Ας υποθέσουμε ότι έχουμε μεταγλωττίσει ένα πρόγραμμα progr1.c και ότι τοποθετήσαμε τον εκτελέσιμο κώδικα στο αρχείο progr1.exe. Αν τώρα πληκτρολογήσουμε

A:\> progr1

όταν θα απαιτείται είσοδος αυτή θα δίνεται από το πληκτρολόγιο ενώ όταν παρέχεται κάποια έξοδος, αυτή θα εμφανίζεται στην οθόνη του υπολογιστή. Στην περίπτωση που θα θέλαμε η είσοδος να διαβάζεται από ένα ASCII ή text αρχείο (δηλαδή αρχείο χαρακτήρων), τότε μπορούμε να χρησιμοποιήσουμε το σύμβολο '**<**' για ανακατεύθυνση της εισόδου.

Για παράδειγμα, αν η είσοδος είναι στο text αρχείο words.txt τότε μπορούμε να πληκτρολογήσουμε

A:\> progr1 < words.txt

ΑΝΑΚΑΤΕΥΘΥΝΣΗ ΕΙΣΟΔΟΥ/ΕΞΟΔΟΥ [2/2]

Αντίστοιχα, αν θέλουμε να ανακατευθύνουμε την έξοδο προς το αρχείο outwords (αντί της οθόνης):

A:\> progr1 > outwords

ενώ μπορούμε να έχουμε ταυτόχρονη ανακατεύθυνση εισόδου και εξόδου όπως για παράδειγμα με το

A:\> progr1 < words.txt > outwords

Όταν γράφουμε τα αποτελέσματα ενός προγράμματος σε ένα αρχείο (π.χ. στο outwords) τότε αυτό το αρχείο είτε δημιουργείται εκείνη τη στιγμή (αν δεν υπήρχε) είτε σβήνεται και ετοιμάζεται για γράψιμο (αν προϋπήρχε).

Αν θέλουμε να διατηρήσουμε το περιεχόμενο ενός αρχείου και απλώς να προσθέσουμε τη νέα έξοδο στο τέλος του αρχείου τότε χρησιμοποιούμε το σύμβολο '**>>**':

A:\> progr1 < morewords.txt >> outwords

ΚΑΝΑΛΙΑ ΕΠΙΚΟΙΝΩΝΙΑΣ– ΡΟΕΣ ΔΕΔΟΜΕΝΩΝ [1/3]

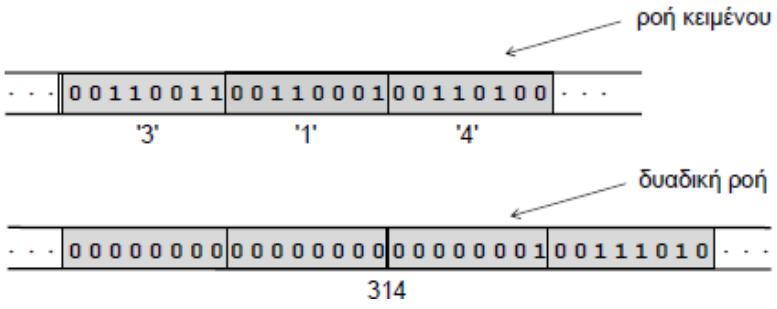
- Ένα **κανάλι επικοινωνίας** (ή **ροή**) είναι μια αλληλουχία από bytes δεδομένων που προέρχονται ή αποστέλλονται σε κάποια περιφερειακή συσκευή.
- Μια **ροή εισόδου** αναφέρεται σε bytes δεδομένων που προέρχονται από μια συσκευή εισόδου (π,χ, πληκτρολόγιο, αρχείο, τηλέτυπο, κ.λπ.). Μια **ροή εξόδου** αναφέρεται σε bytes δεδομένων που αποστέλλονται σε μια συσκευή εξόδου (π,χ, οθόνη, αρχείο, εκτυπωτή, κ.λπ.). Μια **ροή εισόδου/ εξόδου** μπορεί να στέλνει αλλά και να λαμβάνει δεδομένα από μια περιφερειακή συσκευή, όπως ένα αρχείο στο δίσκο ή ένα modem.
- Στο πρότυπο ANSI της C υπάρχουν τρια προκαθορισμένα κανάλια – ροές που αναφέρονται επίσης και ως τυπικές ροές εισόδου/εξόδου. Αυτές οι ροές είναι οι εξής:
  - η **stdin** (τυπική είσοδος) που λαμβάνει δεδομένα (bytes) από το πληκτρολόγιο,
  - η **stdout** (τυπική έξοδος) που στέλνει δεδομένα (bytes) στην οθόνη, και
  - η **stderr** (τυπικό σφάλμα) που εμφανίζει μηνύματα λάθους στην οθόνη.

ΚΑΝΑΛΙΑ ΕΠΙΚΟΙΝΩΝΙΑΣ– ΡΟΕΣ ΔΕΔΟΜΕΝΩΝ [2/3]

- Μια ροή ονομάζεται **ροή κειμένου** αν αποτελείται μόνο από χαρακτήρες, δηλαδή τα bytes αντιστοιχούν σε ASCII κωδικούς. Για παράδειγμα, σε μια τέτοια ροή, τα αριθμητικά σύμβολα 314 ερμηνεύονται ως οι τρεις αριθμητικοί χαρακτήρες '3', '1' και '4' που καταλαμβάνουν τρια διαδοχικά bytes.
- Οι ροές κειμένου οργανώνονται σε γραμμές έως 255 χαρακτήρων οι οποίες τερματίζονται με τον χαρακτήρα νέας γραμμής (newline ή '\n'). Ο χαρακτήρας αυτός κατά την εγγραφή σε αρχείο μετατρέπεται σε δύο χαρακτήρες: τον χαρακτήρα επιστροφής του δρομέα (carriage return με ASCII κωδικό 13) και τον χαρακτήρα προώθησης γραμμής (line feed με ASCII κωδικό 10).
- Αντιθέτως, όταν κατά το διάβασμα από ένα αρχείο εντοπιστούν οι δύο αυτοί χαρακτήρες, η C τους μετατρέπει στον χαρακτήρα newline.
- Τα προκαθορισμένα κανάλια stdin, stdout και stderr είναι ροές κειμένου που "ανοίγουν" αυτόματα με την έναρξη εκτέλεσης ενός προγράμματος.

ΚΑΝΑΛΙΑ ΕΠΙΚΟΙΝΩΝΙΑΣ– ΡΟΕΣ ΔΕΔΟΜΕΝΩΝ [3/3]

- Μια ροή ονομάζεται **δυαδική ροή** αν δεν περιορίζεται μόνο σε δεδομένα κειμένου αλλά σε οποιοδήποτε είδος δεδομένων. Τα bytes δεδομένων διαβάζονται και γράφονται από/σε **δυαδικά αρχεία** ακριβώς όπως είναι. Για παράδειγμα, σε μια τέτοια ροή, ο κάθε ακέραιος (π.χ. ο 314) έχει αριθμητική (δυαδική) αναπαράσταση τεσσάρων bytes ενώ σε ροή κειμένου το πλήθος των bytes είναι όσα και τα ψηφία του αριθμού.



ΣΥΝΑΡΤΗΣΕΙΣ ΡΟΩΝ ΔΕΔΟΜΕΝΩΝ

- Οι περισσότερες από τις συναρτήσεις που χρησιμοποιούνται στις τυπικές ροές εισόδου και εξόδου έχουν και τις αντίστοιχές τους για είσοδο/έξοδο δεδομένων από οποιαδήποτε ροή απαιτώντας από τον προγραμματιστή να προσδιορίσει την ροή.
- Για παράδειγμα, οι συναρτήσεις printf(), scanf(), puts(), gets(), putchar() και getchar(), που έχουμε χρησιμοποιήσει επανειλημμένα στις τυπικές ροές, έχουν και τις αντίστοιχες συναρτήσεις fprintf(), fscanf(), fputs(), fgets(), putc(), fputc(), getc() και fgetc() με τις οποίες μπορούμε να διαβάσουμε ή να γράψουμε δεδομένα σε ένα αρχείο κειμένου. Ένα από τα ορίσματα των συναρτήσεων αυτών προσδιορίζει τη συγκεκριμένη ροή δεδομένων από ή προς το αρχείο.
- Όπως με τις τυπικές ροές έτσι και κατά το διάβασμα ή γράψιμο σε αρχείο χρησιμοποιείται buffer για προσωρινή αποθήκευση των δεδομένων.

ΑΡΧΕΙΑ [1/6]

Στην C τα αρχεία αναπαρίστανται με μια δομή, ο ορισμός της οποίας είναι στη βιβλιοθήκη <stdio.h>. Αν και δεν θα εξετάσουμε τις λεπτομέρειες, παραθέτουμε ένα παράδειγμα ορισμού της δομής **FILE** που χρησιμοποιούν όλες οι συναρτήσεις που χειρίζονται αρχεία στην C:

```
struct _iobuf {
        char *_ptr; /* current buffer pointer */
        int _cnt; /* current byte count */
        char *_base; /* base address of I/O buffer */
        int _flag; /* control flags */
        int _file; /* file number */
};

#define FILE struct _iobuf /* shorthand */
```

ΑΡΧΕΙΑ [2/6]

```
/* Εμφανίζει το περιεχόμενο του αρχείου "test" στην οθόνη */
#include <stdio.h>
#define ERROR -1
int main() {
        int ch;
        FILE *fp; /* Δείκτης σε αρχείο */
        if ( (fp = fopen("test","r"))  ==  NULL ) {
                /* Άνοιξε το αρχείο test για διάβασμα ελέγχοντας αν υπάρχει. Αν δεν υπάρχει επιστρέφει NULL
*/
                printf("Cannot open file \"test\" for reading.\n");
                exit(ERROR);
        }
        while( (ch = getc(fp))  !=  EOF ) /*Διάβασε char από fp...*/
                putc(ch,stdout); /*...και εμφάνισέ τον στην οθόνη */
        fclose(fp); /* Κλείσε το αρχείο */
        return 0;
}
```

ΑΡΧΕΙΑ [3/6]

Η **fopen()** είναι υπεύθυνη για το άνοιγμα ενός αρχείου. Η δήλωση τόσο της fopen() όσο και των υπόλοιπων συναρτήσεων που χειρίζονται αρχεία, βρίσκεται στη βιβλιοθήκη <stdio.h> (π.χ. FILE *fopen();) και, συνεπώς, δεν χρειάζεται να μεριμνήσουμε εμείς γι' αυτό. Η fopen() έχει δύο ορίσματα εισόδου: το όνομα του αρχείου (συμβολοσειρά) και μια παράμετρο (επίσης συμβολοσειρά) που αν είναι **"r"** (από το read) ανοίγει το αρχείο για διάβασμα, αν είναι **"w"** (από το write) ανοίγει το αρχείο για γράψιμο (αν δεν υπήρχε τότε δημιουργείται, αν υπήρχε τότε σβήνεται και ξεκινάει το γράψιμο σε άδειο αρχείο), και αν είναι **"a"** (από το append) ανοίγει το αρχείο για να προσθέσουμε στο τέλος του, χωρίς να σβηστούν τα περιεχόμενά του όπως θα έκανε η επιλογή "w". Η fopen() επιστρέφει pointer στο αρχείο που ανοίγει (file pointer), όπως φαίνεται και στο προηγούμενο πρόγραμμα:

```
fp = fopen("test","r");
```

Τέλος, αν η fopen() αποτύχει να ανοίξει το αρχείο (π.χ. αν δεν υπήρχε το αρχείο ή αν ο δίσκος ήταν γεμάτος ή αν το όνομα του αρχείου δεν ήταν νόμιμο, κ.λπ.), τότε επιστρέφει την τιμή **NULL** (μηδενική διεύθυνση). Αυτό μας επιτρέπει να ελέγχουμε αν το άνοιγμα ήταν επιτυχές πριν αποπειραθούμε να διαβάσουμε από το αρχείο. Μόλις τελειώσουμε με το διάβασμα, γράψιμο ή τροποποίηση του αρχείου, πρέπει να κλείσουμε το αρχείο με την

```
fclose(fp);
```

όπου το μόνο όρισμα της fclose() είναι ο FILE pointer. Η fclose() επιστρέφει 0 αν κλείσει με επιτυχία το αρχείο και –1 στην αντίθετη περίπτωση.

ΑΡΧΕΙΑ [4/6]

Οι συναρτήσεις **getc()** και **putc()** (ή **fgetc()**, **fputc()**) είναι αντίστοιχες των **getchar()** και **putchar()** και μπορούν να χρησιμοποιηθούν για διάβασμα (γράψιμο) χαρακτήρων από (σε) αρχείο. Η getc() έχει ως όρισμα εισόδου τον FILE pointer του αρχείου που ανοίχθηκε για διάβασμα και επιστρέφει τον χαρακτήρα που διάβασε από το αρχείο ενώ η putc() έχει δύο ορίσματα: τον χαρακτήρα και τον FILE pointer του αρχείου που ανοίχθηκε για γράψιμο. Έτσι, αν θέλουμε να διαβάζουμε χαρακτήρες από το αρχείο "infile.txt" και να γράφουμε χαρακτήρες στο αρχείο "outfile.txt", θα πρέπει να ανοίξουμε τα δύο αρχεία (το ένα για διάβασμα και το άλλο για γράψιμο) και να χρησιμοποιήσουμε τους δείκτες σ'αυτά τα αρχεία μαζί με τις getc(), putc():

```
FILE *fpin, *fpout;
. . .
fpin = fopen("infile.txt","r"); /* Open file for reading */
fpout = fopen("outfile.txt","w"); /* Open file for writing */
. . .
ch = getc(fpin);
. . .
putc(ch,fpout);
. . .
fclose(fpin);
fclose(fpout);
```

Οι getc() και putc() είναι πιο γενικές των getchar() και putchar().
Η getchar() αντιστοιχεί στην **getc(stdin)** και η putchar(ch) αντιστοιχεί στην **putc(ch,stdout)**.

ΑΡΧΕΙΑ [5/6]

Στο παρακάτω πρόγραμμα θα επεξηγηθούν οι συναρτήσεις fopen(), fclose(), feof(), fgets() και fputs() μέσω ενός προγράμματος που αντιγράφει ένα αρχείο κειμένου σε ένα άλλο. Το παρακάτω παράδειγμα αρχείου εξυπηρετεί και τα επόμενα παραδείγματα.

ΑΡΧΕΙΟ lab_data.txt

```
Plithos Foithtwn: 5
A.M.        Onoma     Epwnymo    Bathmos  Apousies
-------------------------------------------------
2010017     Spyros    Pappas       8.5        2
2010112     Trine     Sulfa        6          3
2009003     ELENI     Antoniou     9          1
2010078     Michalis  Zafeiratos   7          0
2008056     Dimitris  Pappas       2.5        0
```

```
#include <stdio.h>
#define MAXLEN 256
int main()
{
  char buffer[MAXLEN];
  FILE *ifp,*ofp;
  ifp = fopen("lab_data.txt","r");
  ofp = fopen("copy.txt","w");
  while ( !feof(ifp) )    /* όσο δεν έχουμε φθάσει στο EOF .. */
  { fgets(buffer,MAXLEN,ifp);
    fputs(buffer,ofp); }
  fclose(ifp);  fclose(ofp);   return 0;
}
```

ΑΡΧΕΙΑ [6/6]

- Στο προηγούμενο πρόγραμμα χρησιμοποιήθηκε η **feof()** για έλεγχο αν φθάσαμε στο τέλος του αρχείου. Η συνάρτηση feof() έχει ως μοναδικό όρισμα έναν δείκτη σε αρχείο. Επιστρέφει 0 όσο δεν έχουμε φθάσει στο τέλος του αρχείου και μη μηδενική τιμή στην αντίθετη περίπτωση. Η feof() χρησιμοποιείται τόσο με αρχεία κειμένου όσο και με δυαδικά αρχεία.
- Η συνάρτηση **fgets()** διαφέρει κατά τι από την gets(). Συγκεκριμένα, η fgets() έχει τρία ορίσματα:
  - δείκτη στον πίνακα χαρακτήρων στον οποίο θα αποθηκευθεί η συμβολοσειρά που θα διαβάσουμε,
  - το μέγιστο μήκος της συμβολοσειράς που θα διαβασθεί, και
  - pointer στο αρχείο από το οποίο διαβάζουμε.
- Στο προηγούμενο παράδειγμα, θα διαβασθούν χαρακτήρες είτε μέχρι να συναντήσουμε το newline '\n' είτε όταν διαβασθούν MAXLEN-1 χαρακτήρες – όποιο συμβεί πρώτο. Και στις δύο περιπτώσεις, στο τέλος τοποθετείται ο τερματικός '\0' χαρακτήρας.
- Η fgets() επιστρέφει την τιμή NULL (κενός δείκτης) σε περίπτωση σφάλματος ή όταν συναντήσουμε τον χαρακτήρα EOF (τέλος αρχείου). Ειδάλλως, επιστρέφει δείκτη στη συμβολοσειρά που διαβάστηκε.
- Όσον αφορά την **fputs()**, η μόνη διαφορά από την puts() είναι ότι στέλνει μια ακολουθία χαρακτήρων σε μια καθορισμένη ροή. Κατά τα άλλα είναι πανομοιότυπη με την puts().