ΑΡΧΕΙΑ (Files)

- 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

- 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 **ανακατεύθυνση 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

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

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

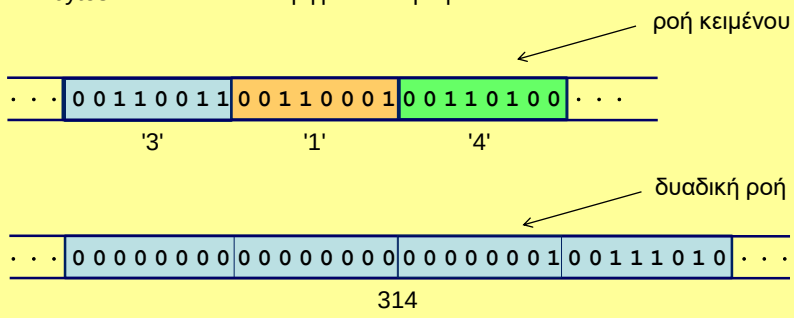
- Ένα **κανάλι επικοινωνίας** (ή **ροή**) είναι μια αλληλουχία από bytes δεδομένων που προέρχονται ή αποστέλλονται σε κάποια περιφερειακή συσκευή.
- Μια **ροή εισόδου** αναφέρεται σε bytes δεδομένων που προέρχονται από μια συσκευή εισόδου (π,χ, πληκτρολόγιο, αρχείο, τηλέτυπο, κ.λπ.). Μια **ροή εξόδου** 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: α) η **stdin** (τυπική είσοδος) που λαμβάνει δεδομένα (bytes) από το πληκτρολόγιο, β) η **stdout** (τυπική έξοδος) που στέλνει δεδομένα (bytes) στην οθόνη, και γ) η **stderr** (τυπικό σφάλμα) που εμφανίζει μηνύματα λάθους στην οθόνη.

ΚΑΝΑΛΙΑ ΕΠΙΚΟΙΝΩΝΙΑΣ

- 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 (η ANSI C υποστηρίζει γραμμές τουλάχιστον 254 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

- 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

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 */
```

Παράδειγμα:

```
/* 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;
}
```

Η **fopen()** είναι υπεύθυνη για το άνοιγμα ενός αρχείου. Η διακήρυξη 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.

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)**.

Σύνταξη συναρτήσεων εισόδου/εξόδου:

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

Επιστρέφει τον χαρακτήρα που διαβάστηκε ή το End Of File (EOF) σε τύπο int.

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

Ίδια συμπεριφορά με fgetc αλλά πιό γρήγορη λόγω υλοποίησης ως μακροεντολής.

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

Εγγραφή του χαρακτήρα ch στο αρχείο με δείκτη fp. Επιστρέφει τον χαρακτήρα ή EOF.

```
char *fgets(char *str, int size, FILE *fp);
```

Διάβασμα (size-1) χαρακτήρων ή μέχρι να συναντήσει το '\n' όποιο έρθει πρώτο και αποθήκευση στο str. Αποθηκεύει και το '\n' κσι προσθέτει στο τέλος και το '\0'. Επιστρέφει δείκτη στη συμβολοσειρά που διαβάστηκε. Αν αποτύχει, επιστρέφει NULL.

```
int fputs(const char *str, FILE *fp);
```

Γράφει τη συμβολοσειρά str στο αρχείο fp και επιστρέφει ακέραιο ή, σε αποτυχία, το EOF.

```
int fprintf(FILE *fp, const char *format, ...);
```

```
int fscanf(FILE *fp, const char *format, ...);
```

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;
}
```

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: α) δείκτη στον πίνακα χαρακτήρων στον οποίο θα αποθηκευθεί η συμβολοσειρά που θα διαβάσουμε, β) το μέγιστο μήκος της συμβολοσειράς που θα διαβασθεί, και γ) 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 (μετά το newline) τοποθετείται ο τερματικός '\0' χαρακτήρας. Η fgets() επιστρέφει την τιμή NULL (μηδενικός 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 χωρίς να προσθέτει το newline στο τέλος.

Συμπληρώστε το πρόγραμμα ώστε να δεσμεύει δυναμικά μνήμη και να αντιγράφει το αρχείο σε πίνακα δομών με την δομή (struct) να ομαδοποιεί τα στοιχεία κάθε φοιτητή.

```
#include <stdio.h>
#define MAXLEN 256
#define ERROR -1

struct foititis {       /* Εξωτερική διακήρυξη δομής */
   char   AM[8];
   char   onoma[30];
   char   epwnymo[30];
   float  bathmos;
   int    apousies;
};

int main()
{
  int  i, N;
  char junk[MAXLEN];
  struct foititis *data;
  FILE *ifp;
```

```
  if ((ifp = fopen("lab_data.txt","r")) == NULL)
  {  printf("Can't open file \"lab_data.txt\" for reading.\n");
     exit(ERROR); }

  fscanf(ifp,"%s %s %d\n",junk,junk,&N);
   /* Βάζουμε \n ή \t ή [space] στο τέλος του format string της fscanf για να
      μη μείνουν σκουπίδια στο buffer (newline) και τα διαβάσει η fgets() */
  fgets(junk,MAXLEN,ifp);    /* Προσπέρασε αυτή τη γραμμή … */
  fgets(junk,MAXLEN,ifp);    /* … και αυτή τη γραμμή */

  data = (struct foititis *) malloc(N*sizeof(struct foititis));

  for(i=0; i<N; i++)
    fscanf(ifp,"%s %s %s %f %d",data[i].AM,data[i].onoma,
              data[i].epwnymo,&data[i].bathmos,&data[i].apousies);

  fclose(ifp);

  printf("Just read file \"lab_data.txt\".\n");
  . . . περισσότερος κώδικας . . . .
  free(data);                /* Αποδέσμευσε τη μνήμη */
  return 0;
}
```

Γράψτε συνάρτηση που να εμφανίζει τα στοιχεία των φοιτητών που το επώνυμό τους αρχίζει από συγκεκριμένο γράμμα. Η συνάρτηση αυτή θα δέχεται ως είσοδο το γράμμα και τον πίνακα δομών, θα εμφανίζει τα στοιχεία και θα επιστρέφει ακέραιο ίσο με το πλήθος των φοιτητών που θα εμφανιστούν στην οθόνη.

```
int emfanisi_stoixeiwn(char ch, struct foititis pin[], int N)
{
  int  i,num=0;

  for(i=0; i<N; i++)
    if(pin[i].epwnymo[0] == ch)
    {
      printf("%s %s %s %f %d\n", pin[i].AM, pin[i].onoma,
              pin [i].epwnymo, pin[i].bathmos, pin[i].apousies);
      num++;
    }

  return num;
}
```