Προγραμματισμός ΙΙ

Είσοδος/Έξοδος

Κωνσταντίνος Τσερπές

(βασισμένο στις διαφάνειες του κ. Δημήτρη Μιχαήλ)

Τμήμα Πληροφορικής και Τηλεματικής

Χαροκόπειο Πανεπιστήμιο

Είσοδος/Έξοδος

Μέχρι τώρα όποτε θέλαμε να διαβάσουμε χρησιμοποιούσαμε πάντα το πληκτρολόγιο και γράφαμε πάντα στην οθόνη.

Η είσοδος και η έξοδος δεν είναι μέρος της γλώσσας C αλλά μέρος της βιβλιοθήκης της.

Επειδή τα προγράμματα σχετίζονται με το περιβάλλον τους με πολύ πιο πολύπλοκους τρόπους από ότι έχουμε δει ως τώρα, θα μελετήσουμε λίγο την βιβλιοθήκη εισόδου/εξόδου.

Βιβλιοθήκη

Το αρχείο που περιέχει τους ορισμούς των τύπων και των συναρτήσεων που είναι απαραίτητες για την συναναστροφή με το σύστημα εισόδου/εξόδου της C είναι το **stdio.h**.

Το όνομα του αρχείου σημαίνει **St**andar**d** **I**nput **O**utput.

Μέσω του αρχείου **stdio.h** ένα πρόγραμμα μπορεί να έχει επαφή με την τυπική είσοδο (συνήθως πληκτρολόγιο) και την τυπική έξοδο (συνήθως οθόνη). Οι τυπικές είσοδοι/έξοδοι παρέχονται στην C μέσω του εκάστοτε λειτουργικού συστήματος.

Ιεραρχία Δεδομένων

- bit - η μικρότερη μονάδα
  - 0 ή 1
- byte - 8 bits
  - αποθήκευση ενός χαρακτήρα (αριθμοί, γράμματα, σύμβολα, ...)
- word - 4 bytes (ή 8 bytes)
  - αποθήκευση αριθμού (int, float, long, …)
- Αρχείο - συλλογή πολλών bytes
- Βάση δεδομένων - συλλογή αρχείων (και όχι μόνο)

Αρχεία

Θα θέλαμε μέσα από την C να έχουμε πρόσβαση σε αρχεία, τα οποία δεν είναι ήδη συνδεδεμένα με το πρόγραμμα μας (όπως η τυπική είσοδος και έξοδος).

Για αυτό τον λόγο η βιβλιοθήκη της C μας παρέχει την συνάρτηση **fopen**() η οποία αναλαμβάνει την επικοινωνία με το λειτουργικό σύστημα. Το λειτουργικό σύστημα είναι υπεύθυνο για την οργάνωση των αρχείων.

Αρχεία

Η συνάρτηση **fopen**() είναι της μορφής

```
FILE * fopen ( char * name , char * mode ) ;
```

Παίρνει δύο παραμέτρους

1. μια συμβολοσειρά με το όνομα του αρχείου
2. μια συμβολοσειρά με διάφορες επιλογές για τον τρόπο πρόσβασης στο αρχείο

και επιστρέφει ένα δείκτη ενός τύπου **FILE**.

FILE

Ο τύπος **FILE** ορίζεται μέσα στην βιβλιοθήκη της C μέσω ενός typedef.

Οι λεπτομέρειες υλοποίησης δεν μας απασχολούν. Αρκεί να πούμε πως ένας δείκτης τύπου **FILE** δείχνει κάπου όπου το λειτουργικό σύστημα κρατάει πληροφορίες για το αρχείο που μας ενδιαφέρει.

Άνοιγμα Αρχείων και Ροές

Μία κλήση στην **fopen**() έχει την εξής μορφή

```
FILE * fp ;
fp = fopen ( name , mode ) ;
```

όπου **name** είναι το όνομα του αρχείου που θέλουμε να επεξεργαστούμε και **mode** είναι μια συμβολοσειρά που δηλώνει την χρήση του αρχείου.

Με την συνάρτηση **fopen**() λέμε πως **ανοίγουμε** ένα αρχείο. Για να χρησιμοποιήσουμε ένα αρχείο στην C πρέπει πρώτα να το ανοίξουμε, δηλαδή να δημιουργήσουμε μια σύνδεση μεταξύ του πραγματικού αρχείου και της γλώσσας C. Την σύνδεση αυτή, που αποκαλούμε **ροή** την φροντίζει το λειτουργικό.

Τυπικές Ροές

Όταν ξεκινήσει η εκτέλεση ενός προγράμματος ανοίγονται αυτόματα 3 ροές:

1. η τυπική είσοδος (συνήθως από το πληκτρολόγιο) με δείκτη **stdin**
2. η τυπική έξοδος (συνήθως η οθόνη) με δείκτη **stdout**
3. το τυπικό σφάλμα (συνήθως η οθόνη) με δείκτη **stderr**

Προσέξτε πως στα περισσότερα λειτουργικά συστήματα μπορούμε να ανακατευθύνουμε τις τυπικές ροές.

Αρχεία και getchar()

Η τυπική βιβλιοθήκη της C παρέχει πολλές συναρτήσεις για ανάγνωση δεδομένων από αρχεία και εγγραφή δεδομένων σε αρχεία.

Για παράδειγμα η συνάρτηση

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

διαβάζει ένα χαρακτήρα από ένα αρχείο, όπως η getchar() διαβάζει από την τυπική είσοδο.

Με άλλα λόγια η κλήση fgetc(stdin) είναι ουσιαστικά ισοδύναμη με την κλήση της getchar().

Αρχεία και putchar()

Για να γράψουμε ένα χαρακτήρα σε ένα αρχείο η τυπική βιβλιοθήκη της C μας παρέχει την συνάρτηση

```
int fputc ( int c , FILE * stream ) ;
```

η οποία γράφει τον χαρακτήρα c στο αρχείο που συνδέεται μέσω της ροής stream. Είναι δηλαδή η αντίστοιχη της συνάρτησης putchar(int c) η οποία μας επιτρέπει να γράφουμε στην τυπική έξοδο.

Με άλλα λόγια η κλήση fputc('a', stdout) είναι ουσιαστικά ισοδύναμη με την κλήση της putchar('a').

Σειριακή ή Τυχαία Προσπέλαση

Στην C μπορούμε να προσπελάσουμε ένα αρχείο με 2 τρόπους

1. σειριακή προσπέλαση
   - διαβάζουμε ή γράφουμε σειριακά από την αρχή του αρχείου
   - τα αρχεία αυτά είναι **αρχεία κειμένου**
2. τυχαία προσπέλαση
   - διαβάζουμε ή γράφουμε σε τυχαία σημεία του αρχείου
   - τα αρχεία αυτά είναι **δυαδικά αρχεία**

Δημιουργία Αρχείου Σειριακής Πρόσβασης

Η C δεν επιβάλλει κάποια δομή σε ένα αρχείο. Είναι καθήκον του προγραμματιστή να δομήσει ένα αρχείο ώστε να ικανοποιήσει τις απαιτήσεις μιας εφαρμογής.

Για να δημιουργήσουμε ένα αρχείο για γράψιμο καλούμε την συνάρτηση **fopen**() με τον παρακάτω τρόπο

```
FILE * fd = fopen ( "filename" , "w" );
```

Το "w" σημαίνει άνοιγμα του αρχείου για γράψιμο. Σε περίπτωση που το αρχείο υπάρχει ήδη, τα ήδη υπάρχοντα δεδομένα σβήνονται.

Δημιουργία Αρχείου Σειριακής Πρόσβασης

Παράδειγμα

```
1   #include
<stdio.h>
3   main ( )
4   {
5       int  AM ;
6       char  name [30] ;
7
8       FILE  * fd ;
9       if  ( ( fd=fopen ( "students.dat" ,  "w" ) )  == NULL  )
10          printf ( "File␣could␣not␣be␣opened\n" ) ;
11      else  {
12          printf ( "Enter␣AM␣and␣name,\n" ) ;
13          printf ( "Enter␣EOF␣to␣end␣input.\n" ) ;
14          printf ( "?␣" ) ;
15          scanf (  "%d%s" ,  &AM ,  name      );
16          while ( ! feof (  stdin ) ) )          {
17              fprintf (  fd ,  "%d␣%s\n" ,  AM ,  name ) ;
18              printf ( "?␣" ) ;
19              scanf (  "%d%s" ,  &AM ,  name );
20          }
21          fclose (  fd );
22      }
23  }
```

Τιμή Επιστροφής της fopen()

Όταν ανοίγουμε ένα αρχείο πρέπει πάντα να κάνουμε έλεγχο της τιμής που μας επιστράφηκε.

Εαν για κάποιο λόγο το αρχείο δεν ανοίχτηκε (π.χ δεν επιτρέπεται η πρόσβαση στον φάκελο που θέλουμε, δεν υπάρχει χώρος στον δίσκο, κ.τ.λ.) η συνάρτηση fopen() επιστρέφει NULL.

```
FILE  * fd ;
if  (( fd=fopen ( "students.dat" ,  "w" ) ) == NULL  )
        printf ( "File␣could␣not␣be␣opened\n" ) ;
```

Έλεγχος Τέλος Αρχείου
End-Of-File (EOF)

Πολλές φορές θέλουμε να δούμε εάν ένα αρχείο έφτασε στο τέλος του. Για τον σκοπό αυτό η C μας παρέχει την συνάρτηση

```
int  feof ( FILE  * stream ) ;
```

η οποία παίρνει ένα ανοικτό αρχείο ως παράμετρο και επιστρέφει αλήθεια εάν το αρχείο έχει φτάσει στο τέλος του (**EOF**), διαφορετικά επιστρέφει ψέμματα.

Με την κλήση **feof**(**stdin**) μπορούμε να ελέγξουμε για **EOF** στην τυπική είσοδο (CTRL-z στα windows, CTRL-d σε unix, κ.τ.λ.).

Γράψιμο σε Αρχείο

fprintf()

Η συνάρτηση

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

λειτουργεί ακριβώς όπως η συνάρτηση **printf**() αλλά στέλνει την έξοδο στην ροή **stream**.

Ουσιαστικά όταν γράφουμε

```
printf ( "hello␣world\n"
);
```

στην πραγματικότητα γράφουμε

```
fprintf ( stdout , "hello␣world\n" ) ;
```

Κλείσιμο Αρχείου

fclose()

Όταν τελειώσει η χρήση ενός αρχείου, ο προγραμματιστής πρέπει να κλείσει αυτό το αρχείο. Κλείνοντας το αρχείο ουσιαστικά λέμε στο σύστημα πως δεν θέλουμε να χρησιμοποιήσουμε άλλο το αρχείο αυτό. Η ροή προς το αρχείο καταστρέφεται.

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

Η fclose() φροντίζει ώστε να γραφούν ότι πληροφορίες δεν έχουν γραφτεί (βρίσκονται σε κάποιον buffer).

Κλείσιμο Αρχείου

fclose()

Σε περίπτωση που ο προγραμματιστής ξεχάσει να κλείσει ένα αρχείο, τα σύγχρονα λειτουργικά συστήματα θα κλείσουν το αρχείο αυτόματα μετά το τέλος του προγράμματος.

Είναι όμως σωστή προγραμματιστική τακτική να φροντίζουν τα προγράμματα μας να ελευθερώνουν οποιουσδήποτε πόρους του συστήματος χρησιμοποιούν.

Δημιουργία Αρχείου Σειριακής Πρόσβασης

Παράδειγμα

```
24      #include <stdio.h>
25
26  main ( )
27  {
28      int AM ;
29      char name [ 3 0 ] ;
30
31      FILE * fd ;
32      if (    ( fd=fopen ( "students.dat" , "w" ) ) == NULL )
33          printf ( "File could not be opened\n" ) ;
34      else {
35          printf ( "Enter AM and name,\n" ) ;
36          printf ( "Enter EOF to end input.\n" ) ;
37          printf ( "? " ) ;
38          scanf ( "%d%s" , &AM , name     );
39          while (! feof ( stdin ) ) {
40              fprintf ( fd , "%d %s\n" , AM , name ) ;
41              printf ( "? " ) ;
42              scanf ( "%d%s" , &AM , name     );
43          }
44          fclose ( fd );
45      }
46  }
```

Εκτελώντας το προηγούμενο πρόγραμμα

```
Enter AM and name,
Enter EOF to end input.
? 11 Kwstas
? 12 Eleni
? 13 Nikos
?
```

δημιουργείται το αρχείο με όνομα **students.dat** στον ίδιο φάκελο με το πρόγραμμα μας. Το αρχείο αυτό περιέχει:

```
11 Kwstas
12 Eleni
13 Nikos
```

Τρόποι Ανοίγματος Αρχείων

Απλές Μορφές

Απλές μορφές ανοίγματος

"w" για δημιουργία αρχείου, ή για διαγραφή των δεδομένων πριν από την εγγραφή.

"r" για διάβασμα υπαρκτού αρχείου, άνοιγμα για ανάγνωση.

"a" για προσθήκη εγγραφών στο τέλος ενός υπαρκτού αρχείου, άνοιγμα για προσάρτηση.

Τρόποι Ανοίγματος Αρχείων

Σύνθετες Μορφές Ενημέρωσης

Σύνθετες μορφές ενημέρωσης (ταυτόχρονα ανάγνωση και εγγραφή)

"r+" ανοίγει αρχείο για ανάγνωση και εγγραφή.

"w+" δημιουργεί ένα αρχείο για ανάγνωση και εγγραφή. Εαν το αρχείο υπάρχει ήδη το αρχείο ανοίγεται και χάνονται τα τρέχοντα περιεχόμενα του.

"a+" άνοιγμα για ανάγνωση και εγγραφή, η εγγραφή γίνεται στο τέλος του αρχείου. Εαν το αρχείο δεν υπάρχει, δημιουργείται.

Ανάγνωση Δεδομένων από Αρχείο Σειριακής Προσπέλασης

Διάβασμα δεδομένων από αρχείο σειριακής προσπέλασης μπορεί να γίνει με την συνάρτηση

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

Πρέπει απλά να ανοίξουμε το αρχείο στην κατάλληλη κατάσταση, π.χ "r".

Ανάγνωση Δεδομένων

Αρχείο Σειριακής Προσπέλασης

```
1   #include <stdio.h>
2
3   main ( )
4   {
5       int AM ;
6       char name [ 3 0 ] ;
7
8       FILE * fd ;
9       if ( ( fd=fopen ( "students.dat" , "r" ) ) == NULL )
10          printf ( "File could not be opened\n" ) ;
11      else {
12          printf ( "%-5s%-13s\n" , "AM" , "Name" ) ;
13          fscanf ( fd , "%d%s" , &AM , name );
14
15          while ( ! feof ( fd ) ) {
16              printf ( "%-5d%-13s\n" , AM , name ) ;
17              fscanf ( fd , "%d%s" , &AM , name );
18          }
19          fclose ( fd );
20      }
21  }
```

Ανάγνωση Δεδομένων

Αρχείο Σειριακής Προσπέλασης

Εκτελώντας το πρόγραμμα στον ίδιο φάκελο με το αρχείο "students.dat" το προηγούμενο πρόγραμμα εκτυπώνει:

```
AM    Name
11      Kwstas
12      Eleni
13      Nikos
```

Δείκτης Θέσης Αρχείου

Ο **δείκτης θέσης αρχείου** δείχνει στην επόμενη θέση αρχείου που θα χρησιμοποιηθεί για ανάγνωση ή εγγραφή.

Για παράδειγμα όταν ανοίγουμε ένα αρχείο ο δείκτης θέσης αρχείου δείχνει στην θέση 0.

Ο δείκτης θέση αρχείου είναι ένας αριθμός που δηλώνει τον αριθμό του επόμενου byte στο αρχείο για ανάγνωση ή εγγραφή.

Πολλαπλή Σειριακή Προσπέλαση

Πολλές φορές θέλουμε να διαβάσουμε τα δεδομένα ενός αρχείου περισσότερες από μια φορές.

Η C μας επιτρέπει να επανατοποθετήσουμε τον **δείκτη θέσης αρχείου** ενός προγράμματος στην αρχή του αρχείου. Η συνάρτηση λέγεται:

```
void rewind ( FILE * stream ) ;
```

Προβλήματα Σειριακής Προσπέλασης

Οι εγγραφές που χρησιμοποιήσαμε ως τώρα δεν έχουν σταθερό μέγεθος.

Για αυτόν τον λόγο δεν μπορούμε εύκολα να ενημερώσουμε εγγραφές στην ίδια θέση ενός αρχείου. Για παράδειγμα δεν μπορούμε να ανανεώσουμε την εγγραφή

11 Kwstas

με

11 Aleksandros

αφού η καινούρια εγγραφή είναι μεγαλύτερη σε μέγεθος και θα επικαλύψει την επόμενη εγγραφή.

Συνήθως επαναγράφουμε όλο το σειριακό αρχείο εαν θέλουμε να κάνουμε ενημερώσεις.

Αρχεία Τυχαίας Προσπέλασης

Η C μας παρέχει την δυνατότητα να μετακινούμε τον δείκτη θέσης αρχείου. Αυτό μας επιτρέπει να προσπελάσουμε άμεσα και χωρίς καθυστέρηση συγκεκριμένα σημεία ενός αρχείου.

Για ευκολία μπορούμε να οργανώσουμε το αρχείο μας με σταθερού μεγέθους εγγραφές και έτσι να έχουμε άμεση πρόσβαση σε κάθε εγγραφή.

Αρχεία Τυχαίας Προσπέλασης
fwrite()

Η συνάρτηση **fwrite**() μεταφέρει ένα συγκεκριμένο αριθμό bytes αρχίζοντας σε μια καθορισμένη θέση στην μνήμη, σε ένα αρχείο.

```
size_t fwrite ( const void * ptr , size_t size , size_t nmemb ,
                FILE * stream ) ;
```

Πιο συγκεκριμένα γράφει **nmemb** στοιχεία, το καθένα με μήκος **size** bytes, στη ροή **stream**. Τα στοιχεία βρίσκονται στην θέση **ptr** της μνήμης.
Η συνάρτηση επιστρέφει τον αριθμό των στοιχείων που γράφτηκαν. Εαν γίνει κάποιο λάθος τότε ο αριθμός είναι μικρότερος ή μηδέν.

Αρχεία Τυχαίας Προσπέλασης
fread()

Η συνάρτηση **fread**() μεταφέρει ένα συγκεκριμένο αριθμό bytes από ένα αρχείο σε μια καθορισμένη θέση στην μνήμη.

```
size_t fread ( void * ptr , size_t size , size_t nmemb ,
               FILE * stream );
```

Πιο συγκεκριμένα διαβάζει **nmemb** στοιχεία, το καθένα με μήκος **size** bytes, από τη ροή **stream**. Τα στοιχεία αποθηκεύονται στην θέση **ptr** της μνήμης.
Η συνάρτηση επιστρέφει τον αριθμό των στοιχείων που διαβάστηκαν. Εαν γίνει κάποιο λάθος ή συμβεί EOF τότε ο αριθμός είναι μικρότερος ή μηδέν.

Διαφορές fprintf() και fwrite()

Η συνάρτηση fwrite() χρησιμοποιεί σταθερό αριθμό από bytes ενώ η fprintf() όχι.

```
int n = 99999;
FILE * fd = fopen ( "test.dat" , "w" );
fprintf ( fd , "%d" , n );
```

Η fprintf() θα γράψει 5 bytes στην έξοδο. Με άλλα λόγια το μέγεθος της εξόδου εξαρτάται από τα ψηφία του αριθμού.

Διαφορές fprintf() και fwrite()

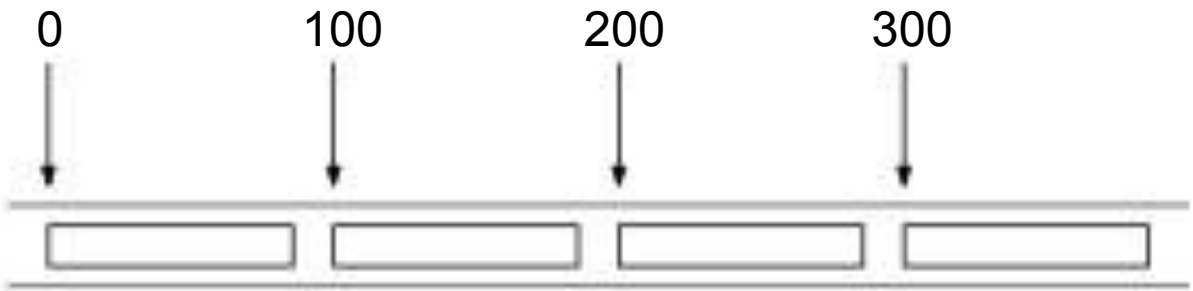
Η συνάρτηση fwrite() χρησιμοποιεί σταθερό αριθμό από bytes ενώ η fprintf() όχι.

```
int n = 99999;
FILE * fd = fopen ( "test.dat" , "w" );
fwrite ( &n , sizeof ( int ) , 1, fd );
```

Η fwrite() γράφει οποιοδήποτε ακέραιο χρησιμοποιώντας 4 bytes.

Αρχεία Τυχαίας Προσπέλασης

Συνήθως γράφουμε συγκεκριμένου μεγέθους εγγραφές στα αρχεία τυχαίας προσπέλασης, ώστε να μπορούμε άμεσα να προσπελάσουμε την *i* εγγραφή.



Δημιουργία Αρχείου Τυχαίας Προσπέλασης

Σειριακή Αρχικοποίηση

```
1   #include <stdio.h>
2
3   typedef struct {
4       int AM;
5       char name[50];
6   } student;
7
8   main() {
9       int i;
10      student blankStudent = {0, ""};
11      FILE *fd;
12
13      if ( (fd=fopen("random.dat", "w")) == NULL )
14          fprintf(stderr, "File could not be opened.\n");
15      else {
16          for ( i = 0; i < 100; i++ )
17              fwrite( &blankStudent, sizeof(student), 1, fd );
18          fclose(fd);
19      }
20  }
```

Μετακίνηση Δείκτη Θέσης Αρχείου

fseek()

Η συνάρτηση

```
int fseek ( FILE * stream , long offset , int whence ) ;
```

θέτει τον δείκτη θέσης αρχείου. Η νέα θέση (σε bytes) προκύπτει προσθέτοντας **offset** στην θέση **whence**.

Η θέση **whence** είναι είτε μια θέση σε bytes είτε μια από τις σταθερές **SEEK_SET**, **SEEK_CUR** και **SEEK_END**. Στην δεύτερη περίπτωση το **offset** είναι σχετικό με την αρχή του αρχείου, την τωρινή θέση και το τέλος του αρχείου αντίστοιχα.

Μια επιτυχής κλήση της **fseek**() σβήνει την ένδειξη **EOF**.

Εύρεση Δείκτη Θέσης Αρχείου

ftell()

Η συνάρτηση

```
long  ftell ( FILE  * stream ) ;
```

επιστρέφει το δείκτη θέσης αρχείου (σε bytes).

Εγγραφή σε Αρχείο Τυχαίας Προσπέλασης

```
1   #include <stdio.h>
2
3   typedef struct {
4       int id;
5       char name[50];
6   } student;
7
8   main()
9   {
10      FILE *fd;
11      student s;
12
13      if ( (fd=fopen("random.dat", "r+")) == NULL )
14          printf("File could not be opened\n");
15      else {
```

Εγγραφή σε Αρχείο Τυχαίας Προσπέλασης

Συνέχεια

```
16              printf ( "Enter␣id␣and␣name,\n" ) ;
17              printf ( "Enter␣EOF␣to␣end␣input.\n" ) ;
18              printf ( "?␣" ) ;
19              scanf ( "%d%s" , &s . id , s . name );
20
21              while ( ! feof ( stdin ) ) {
22                  if ( s . id >= 0 && s . id < 100 ) {
23                      fseek ( fd , s . id * sizeof ( student ) , SEEK_SET );
24                      fwrite ( &s , sizeof ( student ) , 1, fd );
25                  }
26                  else
27                      fprintf ( stderr , "0␣<=␣id␣<=␣99\n");
28                  printf ( "?␣" ) ;
29                  scanf ( "%d%s" , &s . id , s . name );
30              }
31              fclose ( fd );
32          }
33      }
```

Ανάγνωση από Αρχείο Τυχαίας Προσπέλασης

```
1  #include <stdio.h>
2
3  typedef struct {
4          int id;
5          char name[50];
6  } student       ;
7
8  main()
9  {
10         FILE *fd;
11         student s;
12         int id;
13
14         if ((fd=fopen("random.dat", "r")) == NULL)
15                 printf("File could not be opened\n");
16         else {
```

Ανάγνωση από Αρχείο Τυχαίας Προσπέλασης

Συνέχεια

```
17              printf ( "Enter␣id,\n" ) ;
18              printf ( "Enter␣EOF␣to␣end␣input.\n" ) ;
19              printf ( "?␣" ) ;
20              scanf ( "%d" , &id );
21
22              while ( ! feof ( stdin ) ) {
23                      if ( id >= 0 && id < 100 ) {
24                              fseek ( fd , id * sizeof ( student ) , SEEK_SET );
25                              fread ( &s , sizeof ( student ) 1, fd,);
26                              printf ( "Student's␣name␣is␣%s\n" , s . name ) ;
27                      }
28                      else
29                              fprintf ( stderr , "0␣<=␣id␣<=␣99\n" );
30                      printf ( "?␣" ) ;
31                      scanf ( "%d" , &id );
32              }
33              fclose ( fd );
34      }
35      return 0 ;
36  }
```