Δομημένος Προγραμματισμός

Τμήμα Επιχειρηματικού Σχεδιασμού και Πληροφοριακών Συστημάτων

www.bpis.teicrete.gr

Διαχείριση Αρχείων

Υπάρχουν πέντε στοιχειώδεις συναρτήσεις

fopen( ) ανοίγει το αρχείο για να χρησιμοποιηθεί
fputc( ) γράφει ένα χαρακτήρα στο αρχείο
fgetc( ) διαβάζει ένα χαρακτήρα από το αρχείο
fclose( ) κλείνει ένα αρχείο
fseek() εκτελεί τυχαία προσπέλασης σε αρχείο.

Άνοιγμα Αρχείων – fopen()

Τύπος αρχείου

```
FILE * fp;
```

Άνοιγμα αρχείων

```
FILE * fopen (const char * filename, const char * mode);
```

Παράμετρος mode

```
"r"   ανάγνωση (read)
"w"   εγγραφή  (write)
"a"   προσθήκη (append)
"b"   δυαδικά δεδομένα (binary)
```

Άνοιγμα Αρχείων – fopen()

Η συνάρτηση fopen( ) λειτουργεί σαν δύο συναρτήσεις ανοίγει ένα αρχείο ώστε να μπορεί να χρησιμοποιηθεί, και επιστρέφει ένα δείκτη αρχείου.

Ο γενικός τύπος της fopen( ) είναι

```
FILE *fp;
   fp = fopen("όνομα αρχείου", "τύπος") ;
```

fp είναι ο δείκτης (index) του αρχείου ο οποίος στην αρχή δείχνει την πρώτη θέση του αρχείου

Παράδειγμα – fopen()

Η συνάρτηση fopen συνήθως γράφεται ως εξής:

```
FILE *fp;
fp = fopen("test.dat", "r");
if ( fp == NULL)
{
      puts("Unable to open file \n");
      exit( ) ;
 }
```

Η ταυτόχρονη ανάγνωση/εγγραφή προσδιορίζεται με το rw

Π.χ. fp = fopen("test.dat", "rw");

Παράδειγμα – fopen()

Η τιμή NULL που ισούται με το 0, χρησιμοποιείται επειδή δεν υπάρχει δείκτης αρχείου που θα μπορούσε να έχει την τιμή 0.

Αν χρησιμοποιήσουμε την fopen( ) για να ανοίξουμε ένα αρχείο για γραφή, θα διαγραφεί κάθε αρχείο με το όνομα αυτό και θα δημιουργηθεί ένα καινούργιο.

Προσοχή, γιατί τα δεδομένα του προηγούμενου αρχείου θα χαθούν οριστικά!!

Διαχείριση Αρχείων

Είσοδος-έξοδος χαρακτήρων

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

fputc()

Η συνάρτηση fputc( ) χρησιμοποιείται για να γράψουμε ένα χαρακτήρα σ' ένα αρχείο το οποίο έχει ανοίξει χρησιμοποιώντας τη συνάρτηση fopen( ) με τον τύπο w

Ο γενικός τύπος της συνάρτησης είναι:

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

```
char ch='a';
fputc (ch, fp);
```

όπου fp είναι ο δείκτης του αρχείου που επιστρέφει η fopen( ) και ch είναι ο χαρακτήρας που παράγεται δηλαδή γράφεται στο αρχείο.

Ο δείκτης του αρχείου λέει στην fputc( ) σε ποιό αρχείο και σε ποιο σημείο να γράψει. Έτσι, γράφει στην αμέσως επόμενη θέση από την τελευταία εγγραφή.

fgetc()

Η συνάρτηση fgetc( ) χρησιμοποιείται για να διαβάζει χαρακτήρες από ένα αρχείο που άνοιξε με χρήση της συνάρτησης fopen ( ) και τύπο r.

Ο γενικός τύπος της συνάρτησης είναι:

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

```
char ch;
ch = fgetc(fp);
```

όπου fp είναι ένας δείκτης (index) αρχείου τύπου FILE που έχει επιστραφεί από την fopen( )

fclose()

Η συνάρτηση fclose( ) χρησιμοποιείται για να κλείνει ένα αρχείο που έχει ανοίξει μετά από κλήση στην fopen( )

Πρέπει να κλείνουμε όλα τα αρχεία πριν τελειώσει το πρόγραμμα

Η fclose( ) κάνει κάτι περισσότερο από το να ελευθερώνει απλώς το δείκτη του αρχείου.

Γράφει οποιαδήποτε δεδομένα δεν έχουν γραφεί στο δίσκο και κάνει ένα τυπικό κλείσιμο του αρχείου στο επίπεδο του λειτουργικού συστήματος

Ο γενικός τύπος της συνάρτησης fclose( ) είναι

```
fclose(fp);
```

Άσκηση

Δημιουργήστε ένα νέο project με όνομα **files**

Γράψτε ένα πρόγραμμα που

θα ανοίγει για εγγραφή ένα αρχείο με το όνομα **data.txt**

θα διαβάζει συνεχώς χαρακτήρες από το πληκτρολόγιο με την συνάρτηση **getchar()**

θα αποθηκεύει κάθε χαρακτήρα στο αρχείο με τη συνάρτηση **fputc()**

Οι χαρακτήρες θα εισάγονται στο αρχείο μέχρι ο χρήστης να εισάγει τον χαρακτήρα #

Άσκηση - Files

```
#include "stdio.h"
void main()
{
        FILE *fp;
                char ch;
        fp = fopen("test.txt", "w" );
        if( fp == NULL)
        {
            printf("Unable to open file \n");
            exit(0);
        }

       do
       {
           ch=getchar( );
           fputc(ch,fp);
       }
       while ( ch != '#' );

          fclose( fp );
}
```

Άσκηση

Επεκτείνετε το πρόγραμμα σας, ώστε να διαβάσετε όλους τους χαρακτήρες από το αρχείο με την **fgetc()** και να τους τυπώσετε στην οθόνη με την **putchar()**

Τυχαία προσπέλαση - fseek()

Η fseek( ) χρησιμοποιείται για να προσδιορίσει την τρέχουσα θέση (συγκεκριμένο byte) σ' ένα αρχείο

Ο γενικός τύπος της συνάρτησης fseek( ) είναι:

fseek (fp, μήκος, αφετηρία) ;

όπου

fp είναι ένας δείκτης αρχείου που επιστρέφει με την κλήση στην fopen( )

μήκος είναι ο αριθμός των byte από την αφετηρία, τα οποία προσδιορίζουν την τρέχουσα θέση

αφετηρία μπορεί να είναι

το 0 για την έναρξη του αρχείου
το 1 για την τρέχουσα θέση
το 2 για το τέλος του αρχείου

Άσκηση - fseek

Επεκτείνετε το πρόγραμμα σας ώστε να μεταφερθείτε στον $10^{ο}$ χαρακτήρα και να τυπώσετε από εκεί και πέρα τους επόμενους 5 χαρακτήρες

Τυχαία προσπέλαση - fseek()

```
fseek(fp,10,0);
int count=1;
while(count<=5)
{
    ch = fgetc(fp);
    putchar(ch);
    count++;
}
```

Διαχείριση Αρχείων

Είσοδος-έξοδος συμβολοσειρών

```
int fputs (const char * s, FILE * fp);
char * fgets (char * s, int n,
                  FILE * fp);
```

Βασικές συναρτήσεις εισόδου-εξόδου

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

Έλεγχος τέλους αρχείου

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

Άσκηση

Δημιουργήστε ένα νέο project με όνομα **files2**

Γράψτε ένα πρόγραμμα που

θα ανοίγει για εγγραφή ένα αρχείο με το όνομα **data2.txt**

με χρήση της συνάρτησης **fprintf()** θα γράψετε 3 γραμμές στο αρχείο με το όνομα σας, επώνυμο και το ΑΜ σας. Χρησιμοποιήστε τις μεταβλητές firstName, lastName, AM αρχικοποιώντας τις με τα στοιχεία σας

μηδενίστε τα περιεχόμενα των μεταβλητών π.χ. strcpy(firstName,"")

στη συνέχεια, με χρήση της συνάρτησης **fscanf()** διαβάστε τα δεδομένα του αρχείου στις ίδιες μεταβλητές και τυπώστε τα στην οθόνη

Διαχείριση Αρχείων

Είσοδος-έξοδος πολλών δεδομένων

```
fwrite (const void * p,size_t size, size_t num,FILE * fp);
fread (void * p,size_t size, size_t num, FILE * fp);
```

Ο (απρόσημος) ακέραιος τύπος **`size_t`** χρησιμοποιείται για τη μέτρηση χώρου μνήμης σε bytes.

Άσκηση

Δημιουργήστε ένα νέο project με όνομα **files3**

```
size_t fwrite (const void * p, size_t size, size_t num, FILE * fp
size_t fread (void * p, size_t size, size_t num, FILE * fp);
```

Γράψτε ένα πρόγραμμα που γράφει σε ένα δυαδικό αρχείο (**data3.bin**) 100 εγγραφές μίας δομής δεδομένων, με την **fwrite()**

Στη συνέχεια, ανοίξτε το αρχείο, και διαβάστε με την **fread()** και τις 100 εγγραφές προβάλοντας τις στην οθόνη

Άσκηση

Δημιουργήστε ένα νέο project με όνομα **files4**

```
size_t fwrite (const void * p, size_t size, size_t num, FILE * fp
size_t fread (void * p, size_t size, size_t num, FILE * fp);
```

Γράψτε ένα πρόγραμμα που να αντιγράφει το δυαδικό αρχείο **binary.bin** στο δυαδικό αρχείο **data4.bin** με χρήση των **fwrite()** και **fread()**

Παράδειγμα - Αντιγραφή δυαδικών αρχείων

```
int main ()
{
   FILE * fin, * fout;
   unsigned char buffer[1000];
   size_t count;
   fin = fopen("binary.bin", "rb");
   if (fin == NULL)
      return -1;
   fout = fopen("data4.bin", "wb");
   if (fout == NULL)
      return -1;
   while ( !feof(fin) )
  {
      count = fread(buffer, 1, 1000, fin);
      fwrite(buffer, 1, count, fout);
   }
   fclose(fin);
   fclose(fout);
   return 0;
}
```

Συναρτήσεις βιβλιοθήκης

Είσοδος και έξοδος <stdio.h>

- Περιέχει όλες τις συναρτήσεις εισόδου-εξόδου
- Προκαθορισμένα αρχεία

```
FILE * stdin;    τυπική είσοδος
FILE * stdout;   τυπική έξοδος
FILE * stderr;   τυπική έξοδος σφαλμάτων
```

- Ισοδυναμίες

```
printf(...) ≡ fprintf(stdout, ...)
scanf(...)  ≡ fscanf(stdin, ...)
```

κ.λπ.

Δείκτες – Πίνακες

Το όνομα ενός πίνακα είναι μία «σταθερά τύπου δείκτη» με τιμή τη διεύθυνση του πρώτου στοιχείου του πίνακα.

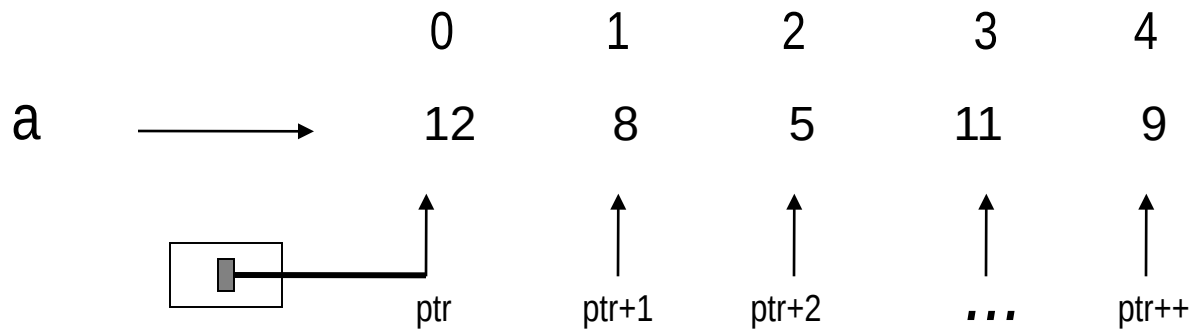
Λόγω αυτού του γεγονότος μπορούμε να ορίσουμε ένα δείκτη στον πίνακα και να χειριστούμε τον πίνακα μέσω αυτού του δείκτη.

Δεν είναι εφικτή η ανάθεση τιμής και η αριθμητική με δείκτες

Δείκτες – Πίνακες

Έστω
int a[5] = { 12, 8, 5, 11, 9};
int *ptr;

ptr = a; ή ptr = &a[0];



Δείκτες – Πίνακες

Ισοδύναμες εκφράσεις

*(a+i) ⟷ a[i]

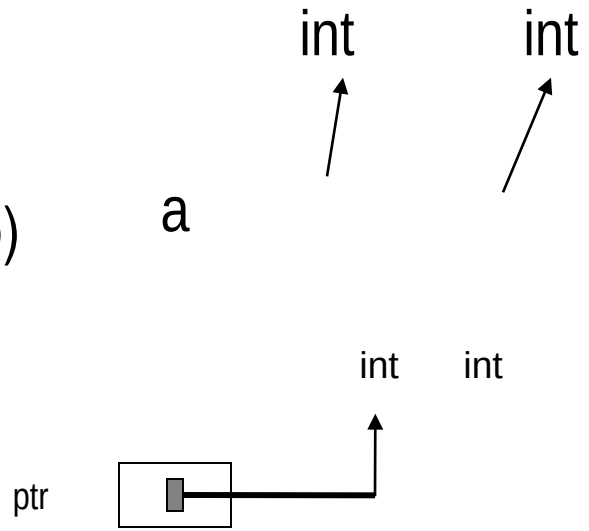
a+i ⟷ &a[i]

Δείκτης σε δείκτη

Προτεραιότητα [ ] μεγαλύτερη από *

π.χ. int *a[2] πίνακας (δύο) δεικτών

int (*ptr)[2] δείκτης σε πίνακα (δύο) ακεραίων



Δείκτες - Αλφαριθμητικά

Δήλωση δείκτη αλφαριθμητικού

char * <όνομα – δείκτη>;

π.χ. char * pmsg;

Ανάθεση σε δείκτη αλφαριθμητικού

char *<όνομα-δείκτη> = < αλφαριθμητικό >;

π.χ. char *pmsg = “Today is Thursday”;

Δείκτες - Αλφαριθμητικά

Προσοχή στις διαφορές

char msg[ ] = “Today is Thursday”;
(πίνακας χαρακτήρων)

char *msg_ptr = “Today is Thursday”;
(δείκτης σε πίνακα)

char *msg[18];
(πίνακας δεικτών χαρακτήρα)

msg[1] → 2ος δείκτης
* (msg[1]) → ο 1ος χαρακτήρας του δεύτερου δείκτη

Παράδειγμα

Υπολογισμός μήκους αλφαριθμητικού όταν μας δίνεται ο δείκτης str στο πρώτο στοιχείο του.

```
int  i=0;
while (str[i++]);
printf("%d", i);
```

```
int  i=0;
while (str[i]) i++;
printf("%d", i);
```

```
int  i=0;
while (str[i] != '\0') i++;
printf("%d", i);
```

```
char *p = str;
while (* str != '\0')
 str++;
printf("%d", str –p );
```

Παράδειγμα

```
void strcpy(char *s, char *t)
{
        int  i=0;
        while ((s[i]=t[i])!='\0')
        i++;
}
```

```
void strcpy(char *s, char *t)
{
while ((*s=*t) != '\0') { s++; t++}
}
```

```
void strcpy(char *s, char *t)
{
while ((* s++=*t++) != '\0') ;
}
```

Παράδειγμα

```
void strcmp(char *s, char *t)
{
        int  i=0;
        for (ι=0; s[i]==t[i]; i++)
            if (s[i]==‘\0’)
                return 0;
        return s[i]-t[i];
}
```

```
int strcmp(char *s, char *t)
{
        for (; *s==*t; s++, t++)
            If (*s==‘\0’)
                return 0;
        return  *s-*t;
}
```

Δυναμική Διαχείριση Μνήμης

Calloc, Malloc,

για δέσμευση μνήμης

Realloc

για αλλαγή μεγέθους δεσμευμένης μνήμης

Free

για απελευθέρωση μνήμης

Malloc()

```
#include <stdlib.h>
void *malloc(size_t size);
```

Η συνάρτηση malloc δεσμεύει χώρο μεγέθους τουλάχιστον size bytes.

Επιστρέφει ένα δείκτη στη δεσμευθείσα μνήμη ή NULL, αν δεν υπάρχει διαθέσιμη μνήμη. Τα bytes του χώρου μνήμης δεν παίρνουν αρχική τιμή.

```
float *fp;
fp = (float *)malloc(sizeof(float));
```

Calloc()

```
#include <stdlib.h>
void *calloc(size_t n,size_t size);
```

Η συνάρτηση calloc δεσμεύει χώρο για ένα πίνακα n στοιχείων, μεγέθους size το καθένα. Διασφαλίζει την αρχικοποίηση της με 0.

Επιστρέφει ένα δείκτη στη δεσμευθείσα μνήμη ή NULL, αν δεν υπάρχει διαθέσιμη μνήμη.

```
float *fp1, *fp2;
fp1 = (float *)calloc(3, sizeof(float) )
fp2 = (float *)malloc( 3*sizeof(float) );
```

Realloc()

```
#include <stdlib.h>

void *realloc(vod *buffer, size_t size);
```

Η συνάρτηση realloc μεταβάλλει το μέγεθος ενός τμήματος μνήμης που είχε προηγούμενα δεσμευτεί. Το νέο μέγεθος ορίζεται από τη size. Διασφαλίζει τα υπάρχοντα περιεχόμενα στη μνήμη. Επιστρέφει ένα δείκτη στο νέο τμήμα μνήμης που μπορεί να είναι ίδιος με τον buffer ή διαφορετικός αν το τμήμα μετακινηθεί. Αν ο buffer είναι NULL, η realloc λειτουργεί σαν τη malloc.

```
int *ptr;
ptr = calloc(5, sizeof(int));
…
ptr = realloc(ptr, 7*sizeof(int));
```

Free()

```
#include <stdlib.h>
void free(void * buffer);
```

Η συνάρτηση free αποδεσμεύει τη μνήμη η οποία σηματοδοτείται από το όρισμα buffer και η οποία είχε προηγούμενα δεσμευτεί με κλήση μίας εκ των *calloc*, *malloc*, *realloc*.

```
float *fp;
fp = (float *)malloc(sizeof(float));
….
free(fp);
```