POLITECNICO DI TORINO – I ANNO

INFORMATICA

Anno Accademico 2012-2013

Erica Raviola

02/05/2014

STRINGHE

Nota: definito `char str[]` allora `char *str` è equivalente a `char str[]` (ovvero usare il puntatore al primo elemento di un array o richiamare il vettore senza specificare la posizione è del tutto analogo).

Definizione di stringa: *array di caratteri terminati dal carattere '/0'.*
Scorrendole è possibile ricavare la loro lunghezza grazie al terminatore. Sono vettori che posso anche non riempire completamente, posso tranquillamente sovradimensionarle. Non c'è un operatore come il più (+) che permette di concatenare due stringhe, non c'è un operatore di maggiore o minore, per lavorare sulle stringhe bisogna basarsi sulle funzioni della libreria <string.h>.

SSCANF E SPRINTF
Sono analoghe a printf e scanf, ma orientate a lavorare sulle stringhe. Utilizzano lo stesso formato e identificatori delle "sorelle" per la stampa/acquisizione da video (e.g. %d per interi, %s per stringhe, %c per un singolo carattere etc.), dovrò passare semplicemente la variabile per la sprintf (passage by value) mentre per la sscanf sarà necessario fornire l'indirizzo della variabile su cui andare a scrivere (passage by reference) e il puntatore alla stringa su cui leggere/scrivere. Cambia la sorgente dei dati (stringa anziché video), entrambe le funzioni tornano il numero di parametri letti/scritti oppure EOF in caso d'errore.

```
//============ ESEMPIO 01 =================//
#define L_MAX       1000

int esempio1() {
charstr[L_MAX];
char data[]="13/05/2013";
int gg, mese, anno;

    /* come faccio a interpretare data e ricavare gg mm anno? */
    sscanf(data, "%d/%d/%d", &gg, &mese, &anno);
    /* passando data così passo il puntatore al primo elemento..*/
    /* ATTENZIONE: str deve essere grande a sufficienza!!! */
    sprintf(str,"%d", gg);
    printf("GG= %d mese=%d anno=%d\n", gg, mese, anno);

    return 0;
}
```

FUNZIONI SULLE STRINGHE (in string.h)

Prototipo	*Descrizione*
char * strcat(char *s1, char *s2)	Concatenazione di s1 con s2, aggiunge s2 dove finisce s1.
char * strchr (char *s, int c)	Cerca il carattere c nella stringa, torna il puntatore alla prima occorrenza del carattere nella stringa,torna direttamente l'indirizzo di memoria (e non l'indice alla posizione) .Posso però ricavare la posizione facendopos=p-s (differenza fra il valore tornato e l'inizio della stringa). Torna NULL se non ha trovato il carattere.
intstrlen(char *s)	Torna la lunghezza di una stringa.
intstrcmp(char *s1, char *s2)	Torna 0 se le due stringhe sono uguali.
char * strcpy(char *s1, char *s2)	Copia s2 in s1, carattere per carattere. Visto che s1=s2 non posso farlo, copia solo la stringa contenuta in s2(si ferma al '/0').

```
int esempio2() {
char str[]="Ciao mondo";
char c='a';
int pos;
char *p;

       p=strchr(str, c);
       if(p==NULL) {
              printf("Non ho trovato il carattere…\n");
       } else {
              pos=p-str;
              printf("Ho trovato il carattere <%c> in posizione %d.\n", c, pos);

       }

}
```

LEZIONE DEL 05/16/2013

Alcune funzioni sulle stringhe, per esempio la strcpy, lavorano sulla stringa fino al '/0' incluso (ignorando il resto). Utilizzando la strcat e volendo concatenare str2="mondo" e str1="ciao", è necessario che str1 sia abbastanza lunga (devo avere spazio sufficiente per poter aggiungere " mondo"), inoltre questa funzione non concatena i due vettori ma il loro contenuto fino al terminatore (concatenazione di stringhe).
Quindi quando parliamo di vettore di caratteri intendiamo tutto l'array (dalla prima all'ultima cella allocata), quando invece usiamo il termine stringa intendiamo l'array che parte dalla posizione 0 fino al terminatore (incluso). I caratteri dal terminatore alla fine effettiva dell'array non sono da considerare. Da questo si ha che la lunghezza di una stringa non è la lunghezza del vettori di caratteri, ma è il numero di caratteri fino al '/0'.
Con la strcat può essere chiesto di immettere la seconda stringa facendo un cast a const, in questo modo viene reso esplicito che la seconda stringa non verrà modificata dalla funzione strcat.

```
/*
    ESERCIZIO DI LABORATORIO ...
    VERIFICARE CHE I DUE VETTORI CONTENGONO GLI STESSI ELEMENTI CON
RIPETIZIONI....
        v1= 1   7   5   1
        v2= 5   7   7   1
In v1 potrei avere due volte 1, in v2 solo una volta...In questo caso non me ne
accorgerei se usassi il metodo senza ripetizioni...
    Soluzioni:
- Potrei controllare v1 con v2 e poi vice versa, ma non mi risolve il problema.
- Potrei prendere i due vettori, ordinarli, e confrontarli posizione per
posizione. Va bene.
- Potrei confrontare le occorrenze dei numeri del primo vettore con quelle del
secondo vettore.
- Usare un vettore di appoggio (di flag)=struttura dati di appoggio per contenere
le informazioni.
*/

#define     N       4

int esempio3(){
int v1[N]={1, 5, 7, 1};
int v2[N]={1, 7, 1, 5};
int f[N]={0};             // vettore di flag... 0=libero, 1=già contato
int i,uguali,j;
/*
    vettore di appoggio        |_|_|_|_|
    cerco 1 in v2              |X|_|_|_|
    cerco 5 in v2              |X|_|_|X|
*/

    uguali=1;

       for(i=0; i<N; i++) {        // scorro elemento per elemento v1...
        // -------- ricerca di v1[i] in v2.....
       trovato=0;
       for(j=0; j<N;j++) { // fissato i-esimo elemento di v1, scorro l'altro.
       if(v2[j]==v1[i] && !f[j]) {  // se sono uguali e se non l'ho ancora contato
              trovato++;
               break;
                f[j]++;            // me lo segno nel vettore di flag...
     }
       }
        //--------- se non l'ho trovato posso uscire...
if(!trovato) {
            uguali=0;
            break;          // esco dal for principale...
        }
    }

if(!uguali) {
printf("I due vettori NON sono uguali. \n");
    } else {

    }


}
```

```
//================================================//
//== MI COSTRUISCO LA STRCHR...             ========//
//===IN: come la strchr..                   ========//
```

```
//===OUT: posizione int in cui ho trovato carattere //
//================================================//
int myStrchr(char *s, int c) {
int trovato, len;

/* non mi serve la lunghezza del vettore, mi serve lungh stringa =>strlen...
posso capire quante celle ho usato, non di quanto è lungo il vettore di
caratteri.. */
       len=strlen(s);
       trovato=0;
    for(i=0;i<len;i++) {
        if(s[i]==c) {
             trovato++;
            break;
        }
    }

    if(trovato) return i;   else return -1;
}


#define LEN 1000
int main() {
char str1[LEN+1], str2[LEN+1];      //len +1 perchè c'è anche il terminatore
int result;

    gets(str1);
    gets(str2);

    puts(str1);
    puts(str2);
    result=strcmp(str1, str2);
    /* strcmp torna 0 se le due stringhe sono uguali, torna positivo quando la
I stringa viene dopo in ordine alfabetico, torna negativo quando la I stringa
viene prima in ordine alfabetico. */

    printf("result: %d\n", result);

    return 0;

}
```

Come posso confrontare due date? Se sono scritte nelle stesso formato es. 20130516 posso confrontare se le due stringhe sono in ordine alfabetico mentre confrontando direttamente anno, mese e giorno (anno1>anno2 &&.... &&...) non funzionerebbe. Oppure posso fare data=anno*1000000+mese*100+gg*10 o contare i giorni da una data fissata (un po' incasinata se ho anni diversi).

VETTORI A PIU' DIMENSIONI

Def. Matrici: *vettori a due dimensioni.*

Per dichiarare una matrice la sintassi è la seguente:

```
      <tipo>  <nome_matrice> [# righe] [# colonne];
```

Esempio:

```
int a[3][5];      //  matrice di interi formata da 3 righe e 5 colonne.

// I          II          III       IV           V
```

```
a[0][0]     a[0][1]     a[0][2]     a[0][3]     a[0][4]
a[1][0]     a[1][1]     a[1][2]     a[1][3]     a[1][4]
a[2][0]     a[2][1]     a[2][2]     a[2][3]     a[2][4]
```

Per specificare i valori degli elementi della matrice durante la sua dichiarazione:

```
int v[3][2]= {(1,1), (2,2), (3,3)};    // specifico i valori riga per riga...
int w[3][2]= {1,1,2,2,3,3};      /* stessa cosa della riga sopra.. in memoria
questa matrice è in realtà un vettore lineare di 6 elementi, ovvero viene salvato
lo sviluppo sulle righe.*/
```

Nota:

	cos'è	***tipo***
a	**matrice**	**int[][]**
a[0]	**vettore(riga 0)**	**int []**
a[2][2]	**elemento**	**int**

Per stampare l'intera matrice uso due for annidati (uno per le righe e uno per le colonne).

```
//===== ESEMPIO 4  ==========//
int esempio4(){
int idx;
int m[3][5]={0};            // matrice a 3 righe x 5 colonne...
                            // ogni riga di una matrice è un vettore...

    m[1][2]=7;    // devo specificare riga e colonna quando assegno un valore.

       for(idx=0; idx<5;idx++){
              printf("%d\t", m[1][idx]);
       }
}
```

LEZIONE 05/17/2013

La matrici possono essere viste come schiere di vettori, un caso interessante è quando le matrici sono di tipo char. In particolare posso interpretare le righe della matrice come le righe di una pagina di testo e le singole colonne come i singoli caratteri che compongono la riga (quindi il massimo numero di caratteri per riga corrisponde anche al numero di colonne).

Esempio:
Matrice 13 x 4.

Lorem ipsum dolor sit amet,
consectetur adipisicing elit…

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	l	o	r	e	m		i	p	s	u	m		s
1	i	t		a	m	e	t	,		c	o	n	s
2	e	c	t	e	t	u	r		a	d	i	p	i
3	s	i	c	i	n	g		e	l	i	t	,	

```
/**
    ESERCIZIO 1- MATRICI.
    - leggere dei numeri da tastiera
    - stampare la matrice
    - somma e media dei valori per colonna
**/
#define     NR      1000   /* numero di righe*/
```

```
#define     NC      1000   /* numero di colonne*/

int esercizio1(){
int m[NR][NC];
int somme[NC];     /* la somma delle colonne di m */
int idx,jdx;

    /* leggiamo 5 volte tre interi da tastiera */
    for(idx=0; idx<NR; idx++) {

#if 0        // istruzione preprocessore, ignoro il codice tra #if e #endif
        printf("Inserisci tre numeri: ");
        scanf("%d %d %d",&m[idx][0], &m[idx][1],&m[idx][2] );
        /* nella scanf posso o no mettere gli spazi...*/
#endif

        /* oppure posso fare cosi, indipendentemente dal numero di colonne...*/
        for(jdx=0; jdx<NC; jdx++) {
            printf("\n[%d][%d]: ", idx, jdx);
            scanf("%d", &m[idx][jdx]);
        }
    }

    /* stampo tutta la matrice  */
    for(idx=0; idx<NR;idx++) {
        for(jdx=0; jdx<NC; jdx++){
            printf("%3d ", m[idx][jdx]);
        }
        printf("\n");
    }

    /* somme per colonne */
    for(jdx=0; jdx<NC; jdx++) {
        for(idx=0; idx<NR; idx++) {
           somme[jdx]+=m[idx][jdx];
        }
    }

    /* stampo il vettore con le somme delle colonne della matrice */
    for(idx=0; idx<NC; idx++) {
        printf("%d\t", somme[idx]);
    }

    return 0;
}
```

ok, il mio programma interagisce con l'esterno.. nel ms-dos basta mettere < per sparare nel mio *.exe
un file con i dati...

MATRICI DI CARATTERI

```
char m[NR][NC];
```

Ogni cella è un char, per stampare tutta la matrice uso due for e "%c".

```
___c
_t__
____
```

Se invece scrivo dei caratteri terminati da “/0”, una riga passa dall’essere un vettore di

caratteri a una stringa.

c i o /0

_ _ _ _

m[idx] => è la stringa corrispondente alla i-esima riga.
Quindi posso scrivere `printf("%s", m[idx]);` la quale stampa la riga come stringa di caratteri. Inoltre, trattando le righe come stringhe, è anche possibile utilizzare funzioni come la `puts(m[idx]);` per la stampa a video.

Con un po' di fantasia, è possibile vedere `char m[#righe]` ***come un vettore formato da stringe*** (ovvero da un array di caratteri terminati da '/0'), così facendo la matrice diventa un vettore "verticale" e non più "orizzontale", dove ogni cella non è più un intero o un singolo char, ma un'intera stringa. In pratica non c'è più la divisione in colonne ma tutte quante vengono unite in una singola stringa.

m[0] |_str0_|
m[1] |_str1_|
m[2] |_str2_|
m[3] |_str3_|

	C0	C1	C2	C3	C4	C5
m[0]=riga 0	C	i	a	o	/0	
m[1]=riga 1	m	o	n	d	o	/0
m[2]=riga 2	h	e	l	l	o	/0
…						
m[k]=riga k	g	u	y	s	/0	

Come faccio a scrivere nella i-esima riga/stringa? Utilizzando le funzioni sprint o gets.

ARGV E ARGC

È possibile che il nostro programma ricevi informazioni dall'esterno. Quando ad esempio lancio l'eseguibile attraverso linea di comando, posso specificare dei parametri di "funzionamento".La funzione main() a cui siamo abituati non è tutto l'eseguibile ma solo una parte. A questa si deve infatti aggiunge un'altra parte che vive nel sistema operativo e che può interagire con questo. I parametri che vengono passati dal mondo esterno al nostro programma vengono passati direttamente alla funzione main attraverso argc e argv:

```
int main(int argc, char* argv) { }
```

argv è un vettore di stringhe e argc (numero intero) indica di quante stringhe è composto argv.

es. argv[0]=> contiene sempre la path del *.exe

Da riga di comando basta passare tutti gli argomenti che voglio dopo *.exe i quali vengono trattati come argv[indice].

Nota:da linea di comado uso " " per inserire un unica stringa contenente spazi.

Tramite argv e argc posso dare dei comandi al main, per convenzione il comandi hanno - se lettere, -- se parole; le stringhe senza il trattino rappresentano i dati su cui applicare i comandi.

Attenzione **:**prima di leggere argv[idx] con idx!=0, devo controllare se argc>1 (ovvero

se ho altri parametri oltre la path del *.exe).

Tutto questo perché vorrei dire all'eseguibile cosa fare direttamente da linea di comando (e senza chiederlo direttamente all'utente).

ATTENZIONE: tutto quello che c'è in argv[] è UNA STRINGA!!!!! no int, float etc.

Quindi per convertire una stringa in un alto tipo (float, int, etc) devo usare delle funzioni (presenti in stdlib.h):

```
- int atoi (const char * str);      // converte stringa in intero.
- double atof (const char* str);    // converte stringa in double.
```

LEZIONE 05/23/2013

Per passare una matrice es. `matrix[N][M]` devo fornirgliela come `matrix[][M]` (ovvero devo SEMPRE specificare di quante colonne è composta).

```
/**
    -------  CONTEGGIO DELLE PAROLE IN UNA STRINGA.  --------------
    -------  CONTEGGIO DELLE PAROLE IN UNA STRINGA.  --------------

    Quando trovo uno spazio sono fuori dalla parola, altrimenti sono dentro.
    Uso una variabile "stato" per identificarlo...
    ^^^^--- informazione di STATO DEL PROGRAMMA.
    Dovrò leggere una stringa carattere per carattere.
    Per ogni carattere, se non è uno spazio stato=IN, se è uno spazio stato=OUT;
    Quando incontro una nuova parola? quando stato passa da OUT a IN.
    Per il primo carattere? stato inizialmente deve essere OUT.
**/

#if 0
#define IN      1
#define OUT     0
#define LEN     100

int main(){
int stato,idx,cntParole;              /* stato può essere in o out... */
charstr[LEN+1]="Ciao mondo 1234";     /* stringa su cui lavoro*/
char c;

    printf("%s\n",str);
    idx=0;
    cntParole=0;
    stato=OUT;                      /* così becco anche la prima parola...*/

    /* per ogni carattere della stringa fino al terminatore */
    while((c=str[idx]) != '\0') {

        /* se il carattere corrente è uno spazio..*/
        if(isspace(c)) {
            stato=OUT;
            printf(".");
        /* se il carattere corrente NON è uno spazio..*/
        } else {
            printf("*");
            /* se è il primo carattere di una parola... */
            if(stato==OUT)  cntParole++;
```

```
            stato=IN;

            /* sono dentro una parola.. lo faccio adesso per non intoccare
stato prima dell'if precedente... */
        }
      idx++;
    }
    printf("\n\nNumero parole: %d\n", cntParole);


     return 0;

}
```

WARNING:

```
char str[100]="Ciao mondo 123";
char parola[100];

      sscanf(str, "%s", parola);       <-- finisce "Ciao"
      sscanf(str, "%s", parola);       <-- finisce "Ciao"
```

La sscanf() legge sempre da inizio stringa, non ha un buffer interno come la scanf().

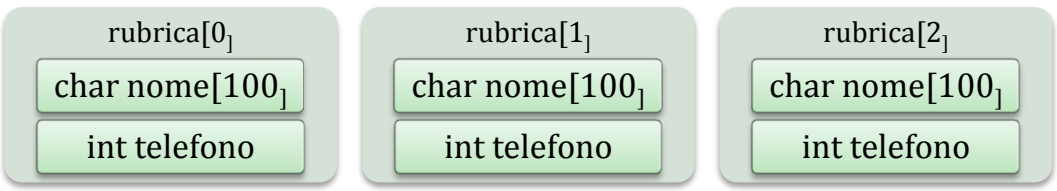
STRUTTURE

Cosa ci permettono di fare le struct? Prendere più variabili e compattarle in un nuovo tipo di dato. Hanno un'utilità concettuale, ***raggruppano fisicamente e logicamente variabili che devono sempre viaggiare insieme***. L'alternativa è avere tanti vettori paralleli.

Con le strutture posso anche fare vettori di strutture. Un esempio classico è la rubrica:

```
struct Contatto { // def. della struct (sto solo definendo il tipo di dato, non
       char nome[100];           // sto ancora occupando spazio in memoria.
       int telefono;
}

int main() {
struct Contatto rubrica[3];
// definisco un vettore lungo 3 di "oggetti" di tipo Contatto.
    ...
}
```



Con l'operatore typedef posso definire un "soprannome" da dare alla struct.

```
      typedef struct <nome_struttura> {
          <variabili contenute dalla struttura>
      } <nome_variabile_struttura>;
```

```
int complexCmp(struct complex num1, struct complex num2);

struct Complex {           // definisco la struttura…
    double re;
    double im;
}cplx;                     // e creo variabile globale (visibile da tutte
funzioni di questo file) di tipo Complex.

/**
    --------- STRUTTURE  ------------------
    --------- STRUTTURE  ------------------
**/
int main(){
struct complex num1, num2;
struct complex num[3];

       memset(&cplx, 0, sizeof(cplx));
       printf("%lld %lld\n", cplx.re, cplx.im);
       return 0;
}

//==================================//
//==================================//
//==================================//
int complexCmp(struct complex num1, struct complex num2) {
    if(num1.re==num2.re && num1.im==num2.im) return 0;  else return -1;
}
```

FILE

I file di testo sono quelli che contengono solo caratteri ASCII, escluse le lettere accentate. Servono per memorizzare, passare, riutilizzare informazioni. I nostri programmi in C possono leggere dei file per avere un modo alternativo alla tastiera d'interfacciarsi con il mondo esterno. Di che tipo sono i file? Per ora noi tratteremo solo file di testo e **SEQUENZIALI**, ovvero che si possono leggere soltanto in ordine, dal primo all'ultimo carattere, non si possono saltare dei pezzi (non posso leggere all'indietro né fare dei jump). Sono costituiti da caratteri (codificati su 1 byte) e sono terminati da EOF. *Il C vede i file come un flusso sequenziale di dati, è compito del programmatore dargli una struttura.*

In C, prima di leggere i file, devo aprire i file (in genere sono chiusi), sul file aperto posso leggere, quando lo chiudo non posso più leggerlo né modificarlo.

I file possono essere sia letti che scritti, a ogni file è associato un FILE* (un cartellino, FILE è una typedef struct, è già definita in stdio.h). Devo definire un FILE *<nome_file>.
Ci sono tre file predefiniti che vengono aperti e chiusi automaticamente all'inizio e alla fine del mio programma:

- **stdin**: input da tastiera
- **stdput**: output su suschermo
- **stderr**:output errori su schermo.

APERTURA FILE

```
FILE * fopen(char * <nomeFile>, char * <modo>);
```

Mi servono:

- il nome del file (dato come puntatore alla stringa)
- modo: r=lettura, w=scrittura (e altri).
- il puntatore al file che mi verrà restituito.

Questa funzione setta il puntatore e me lo restituisce.
Es. ho file chiamato dati.txt, come faccio?

```
FILE *fp;

    fp=fopen("dati.txt", "r");    --> voglio aprire il file in modalità read.
    if(fp==NULL) {
        printf("Il file non puo' essere aperto!\n);
        exit(1);// esco dal programma, anzichè 1 posso scrivere EXIT_FAILURE.
// l'exit esce dal programma in qualunque punto io sono, il return esce solo dalla
//funzione corrente.
    }
```

Dove devo mettere il file per riuscire a leggerlo? Devo metterlo non insieme all'.exe, ma insieme al main.c. In alternativa posso passare come nomeFile l'intera path ("C:\\dati.txt...”). (Nota: devo raddoppiare le \\ per esprimere il carattere '\', in quanto dopo l’\ si aspetterebbe un carattere di controllo).

CASI PARTICOLARI:

- restituisce NULL in caso di errore (il file non può essere aperto).

CHIUSURA FILE

```
int fclose (FILE * <file>);
```

LEZIONE 05/27/2013

Dobbiamo ricordarci che accediamo ai file in modo sequenziale, per cui dovrò sempre leggerlo tutto anche se molte informazioni saranno buttate via. Spesso il numero di righe del file non è noto a priori, non posso memorizzare tutti i dati in memoria e poi utilizzarli (a meno che non usi l’allocazione dinamica per noi off limits), ma devo aggiornarle in tempo reale, devo processare una riga per volta e aggiornare quello che mi serve. Quando voglio leggere un file, lo apro, lo leggo e dopo aver letto l'ultimo carattere ricevo **EOF** (End Of File, è il ctr-z da utente). La lettura è qualcosa del tipo "finchè non è EOF fai qualcosa". Quando scrivo del contenuto non ho questo problema.

LETTURA A CARATTERI

```
int fgetc(FILE *<file>);
```

Legge un carattere per volta, restituisce il carattere letto o EOF se fine file o errore. Le passo il puntatore al file da leggere, torna il carattere letto. E' simile alla `getchar()` (la quale invece legge da tastiera), in più questa deve sapere da quale file leggere (se le passo stdin, questo è il puntatore già settato automaticamente all'input da tastiera).

Quindi `getchar() = getc(stdin);`

SCRITTURA A CARATTERI

```
int fputc(int c, FILE *<file>);
```

Scrive un carattere alla volta nel file, restituisce il carattere scritto o EOF in caso di errore. Poiché questo è un file sequenziale, scrivo subito dopo l'ultima posizione scritta dal mio programma (se l'ho appena aperta allora il cursore è all'inizio). Non posso specificare da dove deve iniziare a scrivere, il cursore va avanti in modo automatico man mano che scrivo sul file. Come la printf() stampa a video sempre in modo sequenziale (cioè senza fare dei salti) così la fputc() scrive carattere per carattere. Anche qui si ha che `putchar(...)=putc(..., stdout);`

LETTURA A RIGHE

```
char* fgets(char*<s>, int<n>, FILE *<file>);
```

La fgets è l'analogo della gets(); la fgets deve sapere da dove leggere la stringa e dove scriverla, devo inoltre specificare quanti caratteri deve al massimo leggere(introdotto per motivi di sicurezza). In pratica n=lunghezza massima di s.
L'eventuale '\n' non viene eliminato(al contrario di gets). Restituisce NULL in caso di errore o EOF.

SCRITTURA A RIGHE

```
int fputs(char *<s>, FILE *<file>);
```

LETTURA FORMATTA

```
int fscanf(FILE *<file>, char *<formato>, ...);
```

Attenzione! deve contenere un numero finito di campi.
Memorizzo il risultato in una variabile per capire quando ho letto tutto il file.

STAMPA FORMATTATA

```
int fprintf(FILE *<file>, char *<formato>, ...);
```

ALTRE FUNZIONI

fflush (tra l'altro con questa posso svuotare il buffer della scanf).

```
int eof(FILE * <file>);
```

restituisce 0 se il puntatore è alla fine del file, !0 se non è alla fine.

SCHEMA GENERALE LETTURA FILE
legge un dato dal file;
finche(non è finito il file) {

elabora;
leggi un dato;
}
oppure
while(fscanf(fp, "...", ...) !=EOF) {
elabora();
}

LEZIONE 05/30/2013

La funzione strcmp confronta il contenuto delle due stringhe e ci permette anche di capire quale delle due viene prima in ordine alfabetico. Torna 0 se sono uguali, 1 o -1 a seconda se la prima stringa viene prima o dopo in ordine alfabetico.

Come faccio a confrontare due stringhe? La strncmp va bene per confrontare "porto" con "bello"... ma in questo caso?

```
char s1="portobello";
char s2="bello";
```

Se scrivo `printf("%s\n", s1[5]);` si pianta perché `s1[5]` è solo un carattere, per trasformarlo nell'indirizzo da dove voglio che parta la stringa dovrò scrivere:
`printf("%s\n", &s1[5]);` così gli passo l'indirizzo del V carattere nella stringa e lui inizia di lì.

Nota: se ho in una scanf() in cui chiedo un intero e subito dopo ho una gets(), la scanf() legge il numero ma il '\n' rimane nel buffer, il '\n' viene letto dalla gets() che lo interpreta come "fine lettura" ed esce subito (senza dare all'utente la possibilità di inserire la stringa). Per ovviare a questo inconveniente utilizzare la funzione fflush.

ESEMPIO:prendo un file e lo carico tutto in memoria. Il nome del file viene dato da linea di comando

```
C:\>program.exe  acquisti-vendite.txt
```

argv[0] argv[1]

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

int main(int argc, char *argv[]) {
FILE *fp;

    if(argc!=2) {
        exit(1);
    }
    fp=fopen(argv[1], "r");
    if(fp==NULL)    exit(1);

    fscanf();
    while(!feof(fp)) {
        fscanf();
```

```
    }

    fclose(fp);
    return 0;

}
```

LEZIONE 06/03/2013

Quando apro un file in "w", perdo completamente il contenuto iniziale e il cursore di scrittura viene posizionato all'inizio. Se invece apro un file in "a" (append), il contenuto viene mantenuto e inizio a scrivere dall'ultimo carattere.

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

/* ho una matrice 4x4.. ma voglio leggere solo una matrice 2x2...*/
#define N       4
#define M       2

//==============================//
int main() {
FILE *pf;
int area[M][M];
inti,j, val;


    pf=fopen("mappa.dat", "r");

    /** SOLUZIONE SBAGLIATA!!!!!!
        ATTENZIONE: nel file le informazioni sono tutte sequenziali, il \n serve
a me per visualizzarlo meglio **/

    for(i=0; i<M; i++) {            /* leggo solo due righe e due colonne... */
        for(j=0; j<M; j++) {
       /* vengono messi correttamente nella matrice ma quali valori prendo?
       prendo quelli sequenziali...*/
              fscanf(pf, "%d", area[i][j]);
        }
    }

    /** SOLUZIONE GIUSTA: è meglio leggere tutta la matrice e prendere solo i
dati che mi interessano... devo leggere tutti e accorgermi di quando capito negli
estremi (x e Y) della mia area. **/

    for(i=0; i<N; i++){      /* in questo for ext potrei fermarmi a M...*/
        for(j=0; j<N;j++) {
            fscanf(pf, "%d", &val);
            if((i>=X1 && i<=X2 )&& (j>=Y1 && j<=Y2))  area[i][j]=val;
        /* x1, x2, y1, y2 sn gli estremi del mio rettangolo..
        può anche non partire    nell'origine....*/
        }
    }

    fclose(pf);
    return 0;
}


/** e se non tutte le righe hanno la stessa dimensione????
es. 20   70  80  90
    56   89  77
    99   66
```

```
    00   55   77   88

    Conosco i valori massimi per le righe e per le colonne...
    in questo caso uso N e N...

    Fin quando non trovo un \n sono sulla stessa riga...
**/
int main() {
FILE *pf;
int area[N][N];
inti,j, val;
char sep;

    pf=fopen("mappa.dat", "r");
    i=0;
    j=0;

    while(feof(fp)!=EOF) {                  /* per tutto il file....*/
        fscanf(fp, "%d%c", &val, &sep);     /* leggo il valore e il carattere
successivo=separatore...*/
        area[i][j]=val;
        if(sep=='\n') {
            i++;    j=0;    /* vado a capo e torno alla colonna */
        } else {
            j++;        /* vado sulla colonna successiva... */
        }
    }

    fclose(pf);
    return 0;
}
```