INPUT/OUTPUT

- L'immissione dei dati di un programma e l'uscita dei suoi risultati avvengono attraverso operazioni di ***lettura e scrittura***

- C ***non ha istruzioni predefinite*** per l'input/output

- In ogni versione ANSI C, esiste una ***Libreria Standard*** (**stdio**) che mette a disposizione alcune funzioni (dette *funzioni di libreria*) per effettuare l'input e l'output

INPUT/OUTPUT

- Le dichiarazioni delle funzioni messe a disposizione da tale libreria devono essere essere incluse nel programma: **`#include <stdio.h>`**
  - **`#include`** è una direttiva per il **preprocessore** C
  - nella fase precedente alla compilazione del programma ogni direttiva "**#**..." viene eseguita, provocando delle modifiche testuali al programma sorgente. Nel caso di **`#include <nomefile>`** viene sostituita l'istruzione stessa con il contenuto del file specificato

- **Dispositivi standard di input e di output:**

  per ogni macchina, sono periferiche predefinite (generalmente tastiera e video)

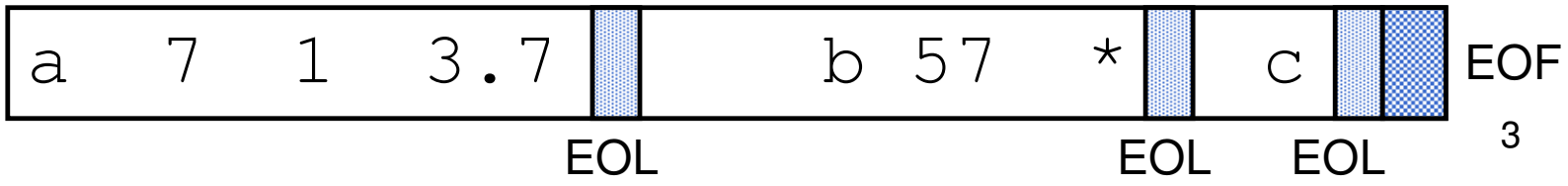
INPUT/OUTPUT

A default, C vede le informazioni lette/scritte da/verso i dispositivi standard di I/O come file *sequenziali*, cioè **sequenze di caratteri** (o stream). Vedremo più avanti la possibilità di fare anche I/O in cosiddetto formato binario...

- Gli *stream* di input/output possono contenere dei caratteri di controllo:
  - End Of File (EOF)
  - End Of Line (EOL)

Sono disponibili funzioni di libreria per:

- Input/Output a caratteri
- Input/Output a stringhe di caratteri
- Input/Output con formato



INPUT/OUTPUT CON FORMATO

Nell'I/O con formato occorre specificare il ***formato*** (*tipo*) dei dati che si vogliono leggere oppure stampare

Il ***formato*** stabilisce:

- **come interpretare** la sequenza dei caratteri immessi dal dispositivo di ingresso (nel caso della lettura)
- con quale sequenza di caratteri **rappresentare** in uscita i valori da stampare (nel caso di scrittura)

LETTURA CON FORMATO: `scanf`

È una **particolare forma di assegnamento**: scanf() assegna i valori letti alle variabili specificate come argomenti (nell'ordine di lettura)

```
scanf(<stringa-formato>, <sequenza-variabili>);
```

Ad esempio:

```
int X;
float Y;
scanf("%d%f", &X, &Y);
```

LETTURA CON FORMATO: `scanf`

scanf() legge una serie di valori in base alle specifiche contenute in *<stringa-formato>* e memorizza i valori letti nelle variabili

- restituisce il ***numero di valori letti*** e memorizzati, oppure EOF in caso di *end of file*
- gli ***identificatori*** delle variabili a cui assegnare i valori sono sempre preceduti dal ***simbolo &*** (ne parleremo diffusamente…)
- la <stringa_formato> può contenere dei caratteri qualsiasi (scartati durante la lettura), che si prevede vengano immessi dall'esterno, insieme ai dati da leggere

```
scanf("%d:%d:%d", &A, &B, &C);
```

richiede che i tre dati vengano immessi separati dal carattere ":"

SCRITTURA CON FORMATO: `printf`

- **`printf()`** viene utilizzata per fornire in uscita il valore di una variabile o, più in generale, il risultato di una espressione
- Anche in scrittura è necessario specificare (mediante una *stringa di formato*) il formato dei dati che si vogliono stampare

```
printf(<stringa-formato>,<sequenza-elementi>)
```

SCRITTURA CON FORMATO: `printf`

- **`printf`** scrive una serie di valori in base alle specifiche contenute in *<stringa-formato>*
- I valori visualizzati sono i risultati delle espressioni che compaiono come argomenti
- **`printf`** restituisce il numero di caratteri scritti
- La stringa di formato della **`printf`** può contenere sequenze costanti di caratteri da visualizzare

FORMATI COMUNI

- ***Formati più comuni***

`int`	`%d`
`float`	`%f`
`carattere singolo`	`%c`
`stringa di caratteri`	`%s`

- ***Caratteri di controllo***

`newline`	`\n`
`tab`	`\t`
`backspace`	`\b`
`form feed`	`\f`
`carriage return`	`\r`

- Per la stampa del carattere ' `%` ' si usa: `%%`

ESEMPIO

```
int main()
{
int     k;
scanf("%d",&k);
printf("Quadrato di %d: %d",k,k*k);
return 0;
}
```

Se in ingresso viene immesso il dato:

`3` viene letto tramite la scanf e assegnato a `k`

`printf()` stampa:

```
Quadrato di 3: 9
```

ESEMPIO

```
scanf("%c%c%c%d%f", &c1,&c2,&c3,&i,&x);
```

Se in ingresso vengono dati:

```
ABC 3 7.345
```

`scanf()` effettua i seguenti assegnamenti:

```
char c1          'A'
char c2          'B'
char c3          'C'
int i            3
float x          7.345
```

ESEMPIO

```
char Nome='F';
char Cognome='R';
printf("%s\n%c. %c. \n\n%s\n",
        "Programma scritto da:",
             Nome, Cognome,"Fine");
```

vengono stampate le seguenti linee

```
Programma scritto da:
F. R.

Fine
```

CHE COSA SUCCEDE?

```
a=0;
printf("ciao e %d\n", a=1);

a=0;
printf("ciao e %d\n", a==1);

a=0;
printf("ciao e %d\n", a==0);

a=0;
printf("ciao e a=%d\n", a);

a=1;
if (a=4)
  printf("ciao e a=%d\n", a);
```

CHE COSA SUCCEDE?

```
a=0;
printf("ciao e %d\n", a=1);
```
ciao e 1

```
a=0;
printf("ciao e %d\n", a==1);
```
ciao e 0

```
a=0;
printf("ciao e %d\n", a==0);
```
ciao e 1

```
a=0;
printf("ciao e a=%d\n", a);
```
ciao e a=0

```
a=1;
if (a=4)
  printf("ciao e a=%d\n", a);
```
ciao e a=4