LIBRERIE STANDARD in C

- La ***libreria standard*** del C è in realtà ***un insieme di librerie***

- Per usare una libreria, ***non occorre inserirla esplicitamente nel progetto***: ogni ambiente di sviluppo *sa già* dove cercarle

- Ogni file sorgente che ne faccia uso deve però ***includere header opportuno*** che contiene le ***dichiarazioni*** necessarie

LIBRERIE STANDARD in C

Le librerie standard

- **input/output** ***stdio.h***
- funzioni matematiche *math.h*
- gestione di stringhe *string.h*
- operazioni su caratteri *ctype.h*
- gestione dinamica della memoria *stdlib.h*
- ricerca e ordinamento *stdlib.h*
- ... e molte altre

IL MODELLO DI INPUT/OUTPUT

- **Libreria standard** `stdio`
- **Input** avviene di norma dal canale **standard di input (*stdin*)**
- **Output** avviene di norma sul canale **standard di output (*stdout*)**
- Input e output avvengono sotto forma di una ***sequenza di caratteri***
- tale sequenza di caratteri può terminare da **un *carattere speciale*** (*End-Of-File*), la cui rappresentazione può variare da un SO ad un altro

CANALI STANDARD

Di norma:

- il canale standard di input, ***stdin***, coincide con la **tastiera**
- il canale standard di output, ***stdout***, coincide con il **video**

Esiste inoltre un altro canale di output, riservato ai messaggi di errore: ***stderr***

- anch'esso di norma coincide con il **video**

MODELLO di BASE per I/O

Poiché sui canali di I/O fluiscono ***sequenze di caratteri***, il modello di I/O prevede *due operazioni base:*

- ***scrivere un carattere sul canale di output***

```
putchar(ch);
```

- ***leggere un carattere dal canale di input***

```
ch = getchar();
```

Ogni altro tipo di I/O può essere costruito a partire da queste ***operazioni primitive***

I/O A CARATTERI

```
int putchar(int ch);
```

- scrive un carattere sul canale di output
- restituisce il carattere scritto, o EOF in caso di errore

```
int getchar(void);
```

- legge un carattere dal canale di input
- restituisce il carattere letto, oppure EOF in caso la sequenza di input sia finita o in caso di errore

Entrambe le funzioni leggono/scrivono un carattere convertito in int

ESEMPIO

Ricopiare l'input standard sull'output standard, carattere per carattere

```
#include <stdio.h>

int main(){
 int c;
 while(( c=getchar() ) != EOF)
  putchar(c);
}
```

Attenzione: getchar() inizia a produrre caratteri solo *dopo* aver premuto INVIO

*Per chiudere l'input producendo un **EOF** da tastiera, **CTRL+Z** in sistemi Win, **CTRL+D** in Unix*

I/O DI TIPI PRIMITIVI

Ogni altro tipo di I/O può essere costruito sulle due primitive putchar() e getchar()

Esempi

- scrivere o leggere ***stringhe*** *di caratteri*
- scrivere o leggere *la* ***rappresentazione di un numero*** (naturale, intero, reale) sotto forma di stringa, in una base data

Queste funzionalità sono già disponibili nella libreria di I/O standard

I/O con FORMATO

La libreria standard offre due funzioni di I/O *di uso generale*, che compendiano tutte le necessità precedenti: printf() e scanf()

```
int printf(...);
```

- scrive sul canale di output una serie di valori, *effettuando le conversioni richieste ove necessario*
- restituisce il numero di ***caratteri emessi***

```
int scanf(...);
```

- legge dal canale di input una serie di *campi*, *effettuando le conversioni richieste ove necessario*
- restituisce il numero di ***campi letti con successo***

I/O con FORMATO

Le funzioni **`printf()`** e **`scanf()`** possono avere ***un numero variabile di parametri***. Inoltre, possono scrivere/leggere:

- **singoli caratteri**
- **stringhe di caratteri *formattate nel modo indicato dall'utente***
- **interi, con o senza segno, in base 8, 10, 16**
- **reali (float o double) in vari formati**

OUTPUT con FORMATO: printf()

Sintassi:

```
int printf(char frm[], e1,…, eN)
```

- la funzione scrive sul canale di output
  i risultati delle espressioni `e1, …, eN`
  nel formato specificato dalla stringa `frm[]`
- restituisce il **numero di caratteri scritti,**
  o EOF in caso di errore