UNIVERSITÀ
degli STUDI
di CATANIA

Input e output di base in C: stdin, stdout, stderr

Corso di programmazione I (A-E / O-Z) AA 2023/24

Corso di Laurea Triennale in Informatica

Fabrizio Messina
fabrizio.messina@unict.it

Dipartimento di Matematica e Informatica

Libreria standard C per IO

Documentazione: https://en.cppreference.com/w/c/io

Libreria di IO basata sui flussi o (**stream**).

Un flusso è una astrazione utile a modellare le **comunicazioni** tra le applicazioni (i processi) e l'ambiente in cui operano (Sistema Operativo e/o hardware).

Le applicazioni usano gli stream per scambiare dati attraverso la rete, inviare output a video, ricevere input da tastiera, etc.

In C possiamo usare gli stream per l'input e l'output di base mediante lo header `stdio.h`.

Libreria standard C per IO

Gli stream si possono **collegare** di volta in volta ad eventuali **periferiche** come tastiera o video, si usano per operare in un file system locale o remoto e cosi via.

Flussi standard o predefiniti

Header `stdio.h`

- Standard output, rappresentato dal simbolo o macro `stdout`, generalmente associato al **video**.
- Standard input, rappresentato dal simbolo o macro `stdin`, generalmente associato alla **tastiera**.
- Standard error, rappresentato dal simbolo o macro `stderr`, generalmente associato al **video**.

Usare i canali di IO predefiniti

```c
#include <stdio.h>
float y = 10.2;
printf("Hello world! y= %f\n", y);
// oppure
fprintf(stdout, "Hello world! y=%f\n", y);
```

fprintf() / printf() esempio di funzioni con numero di argomenti variabile..

fprintf consente di specificare lo stream di output

Usare i canali di IO predefiniti

```c
#include <stdio.h>
float y = 10.2;
printf("Hello world! y= %f\n", y);
// oppure
fprintf(stdout, "Hello world! y=%f\n", y);
```

Primo argomento printf() / secondo argomento fprintf() è detto "stringa di formato" (o stringa di controllo formato)

Essa può contenere una o più **specifiche di conversione**.

Usare i canali di IO predefiniti

```
1 #include <stdio.h>
2 float y = 10.2;
3 printf("Hello world! y= %f\n", y);
```

Argomenti dopo la stringa di formato devono essere in numero uguale al numero di specifiche di conversione contenute nella stringa di formato:

- %f specifica di conversione (indica che va stampato un numero in virgola mobile)
- y variabile contenente il numero in virgola mobile da stampare

Usare i canali di IO predefiniti

Specifica di conversione:

- inizia con '%';
- zero o più elementi detti **flag**;
- un modificatore di (minima) **larghezza di campo**;
- un modificatore di **precisione** (opzionale) ;
- un **modificatore di lunghezza** (opzionale);
- uno **specificatore di conversione** (necessario);

Usare i canali di IO predefiniti

```
1 printf("%+10d", 148);
```

'+': flag che indica che va posto un segno prima del numero

'10': larghezza di campo minima (10), eventuale riempimento con spazi

d: specificatore di conversione che indica di stampare un intero con segno

Usare i canali di IO predefiniti

```
1 printf("%+10.5f", 148.1234567);
```

'+': flag che indica che va posto un segno prima del numero

'10': larghezza di campo minima (10), eventuale riempimento con zeri (0)

'.5': precisione, ovvero 5 cifre dopo la virgola

f: specificatore di conversione che indica di stampare un numero in virgola mobile

Usare i canali di IO predefiniti

```
1 long k = 10e10; // notazione esponenziale
2 printf("%ld", k);
```

'l': modificatore di lunghezza che indica che il numero e' un long

d: specificatore di conversione che indica di stampare un numero intero

Usare i canali di IO predefiniti

```
1 const char *s = "Hello world!";
2 printf("%s", s);
```

s: specificatore di conversione che indica di stampare uno o più caratteri corrispondenti all'argomento (puntatore a carattere)

Usare i canali di IO predefiniti

Caratteri speciali o “escaped”:

- '\t' rappresenta tabulazione orizzontale (equivalente ad un certo numero di spazi);
- '\n' rappresenta un ritorno a capo (newline);
- '\\' permette di stampare il carattere backslash '\';
- '\r' “ritorno carrello”, il cursore viene posizionato all'inizio della linea di testo;
- %% stampa il carattere '%'.
- '\" ' permette di stampare le virgolette

Esempi svolti

Esempi svolti

8_01_stream.c

8_02_conversion.c

8_03_escaped_characters.c

Input di base: famiglia di funzioni scanf

```
1    #include <stdio.h>
2
3    int scanf(const char *format, ...);
```

scanf() legge caratteri dallo **standard input**, negli argomenti specificati nella stringa di formato;

i caratteri verranno copiati negli argomenti specificati dopo la stringa di formato;

leggendo uno spazio (se non ci sono altri argomenti da riempire) oppure dopo aver letto il new line.

Input di base: famiglia di funzioni scanf

```c
#include <stdio.h>

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

- fscanf() si comporta come scanf(), ma legge dallo stream specificato nel primo argomento;
- sscanf() legge i caratteri dal buffer specificato (è un puntatore a carattere) passata come primo argomento e li pone quindi negli argomenti, sulla base della stringa di formato;

Input di base: famiglia di funzioni scanf

Stringa di formato:

- simile alla stringa di formato della funzione printf(), ma specifica quali dati sono “attesi” in input
- una sequenza di specifiche di conversione, eventualmente separate da spazi (che saranno scartati)

Input di base: famiglia di funzioni scanf

Specifica di conversione:

- Inizia con il carattere '%'
- un modificatore di ampiezza di campo (max numero di caratteri da leggere)
- un modificatore di lunghezza (ad esempio 'l' per i long)
- uno specificatore di conversione (es: d o f)

Esempi svolti

Esempi svolti

8_04_scanf.c

8_04_scanf_b.c

8_04_scanf_chars.c

8_05_fgets.c

8_05_scanf_numbers.c

8_06_scanf.c

Riferimenti e approfondimenti

[1] → Capitoli 2 e 3.

---

[1] Paul J. Deitel and Harvey M. Deitel.
C Fondamenti e tecniche di programmazione.
Pearson, 2022.