ROMA TRE
UNIVERSITÀ DEGLI STUDI

Dipartimento di Ingegneria
Industriale, Elettronica e Meccanica

Collegio Didattico di Ingegneria Elettronica

Elementi di Informatica

Stringhe

Giordano Da Lozzo e Giuseppe Sansonetti

Stringa

Sequenza finita di caratteri

Esempi:

- "Ciao a tutti"
- "3-2=7"
- "mario.rossi@uniroma3.it"
- "@,<>#!"

Stringhe vs caratteri

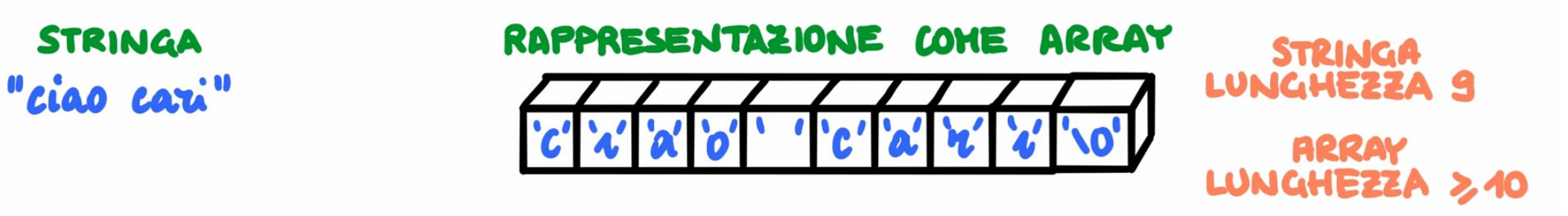
Un carattere può essere memorizzato in una variabile di tipo char.

Una stringa può essere memorizzata in un array di tipo char, ovvero un array i cui elementi sono di tipo char.

Mentre una variabile di tipo char permette di memorizzare un singolo carattere, un array di tipo char permette di memorizzare un'intera sequenza di caratteri, ovvero una stringa.

Carattere di fine stringa

Le stringhe in C sono delimitate da un carattere di fine stringa, che segue l'ultimo carattere "vero" della stringa e ne stabilisce così i limiti.



\0 è il carattere *null*, il cui codice ASCII è 0 (*è il primo carattere del codice ASCII e non il carattere '0'*).

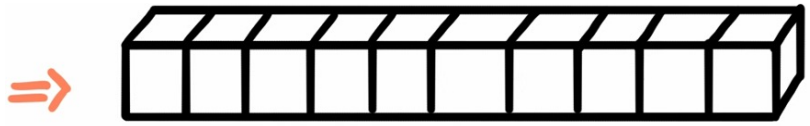
- In questo contesto funge da carattere delimitatore.

Comunemente in C si chiamano stringhe solo le sequenze di caratteri terminate dal carattere delimitatore '\0'.

- Le funzioni predefinite funzionano correttamente solo su stringhe con carattere delimitatore.

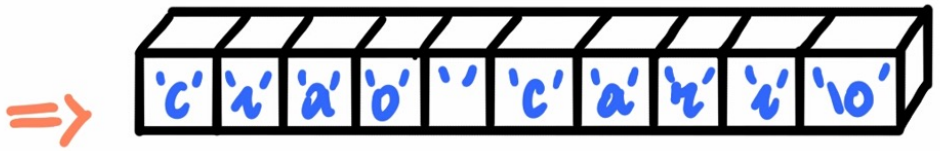
Dichiarazione: come per i "normali" array

```
char stringa[10];
```



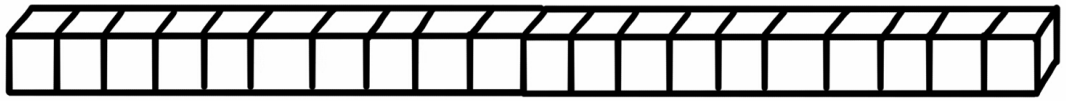
Inizializzazione: come per i "normali" array

```
stringa[0]='c';
stringa[1]='i';
stringa[2]='a';
stringa[3]='o';
stringa[4]=' ';
stringa[5]='c';
stringa[6]='a';
stringa[7]='r';
stringa[8]='i';
stringa[9]='\0';
```



Dichiarazione

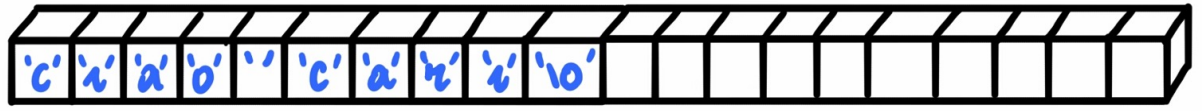
```
char stringa[20];
```

⇒ 

Inizializzazione

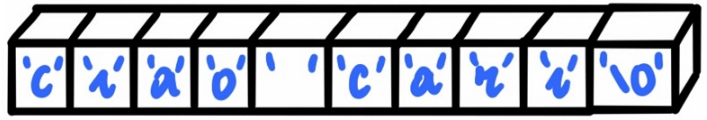
```
stringa[0]='c';
stringa[1]='i';
stringa[2]='a';
stringa[3]='o';
stringa[4]=' ';
stringa[5]='c';
stringa[6]='a';
stringa[7]='r';
stringa[8]='i';
stringa[9]='\0';
```

Spesso (ad esempio quando si legge una stringa da input) l'array che viene dichiarato per memorizzare una stringa è sovradimensionato rispetto alla dimensione della stringa stessa

⇒ 

Dichiarazione con inizializzazione

```
char stringa[] = "ciao cari";
```



STRINGA LUNGHEZZA 9

ARRAY LUNGHEZZA 10

Osserva che, a meno di voler sovradimensionare l'array, non è necessario specificarne la dimensione. Viene allocata memoria sufficiente a memorizzare la stringa (compreso il carattere delimitatore).

ATTENZIONE :

```
#include <stdio.h>

int main() {
    char stringa[10];
    stringa = "ciao cari";
}
```

error: assignment to expression with array type

Stringhe con un carattere e Stringa vuota

```
char carattere='m';
char stringa[]="m";
```

Sono istruzioni diverse!

La prima dichiara una variabile di tipo char e le assegna il valore 'm'. La seconda dichiara un array di elementi di tipo char ed assegna i valori 'm' e '\0' ai due elementi dell'array.

"" è una stringa formata da una sequenza vuota di caratteri.

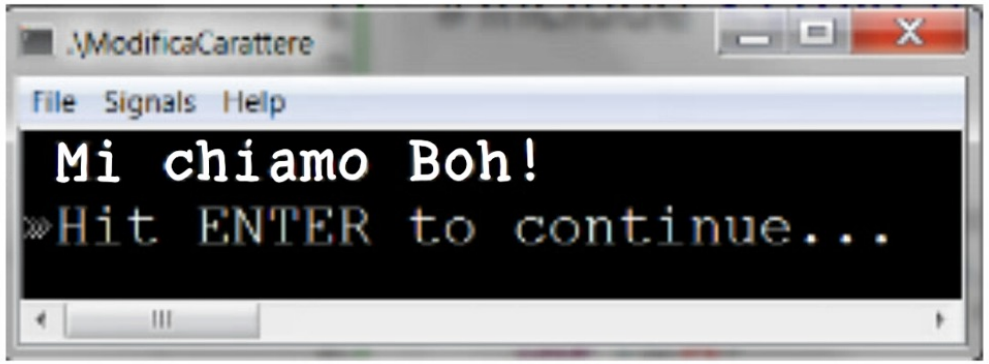
Da non confondere con " ", la stringa che contiene il solo carattere *spazio.*

La rappresentazione in memoria contiene il solo carattere null

Modifica di stringhe

E' possibile modificare la stringa memorizzata in un array di tipo char.

```
int main() {
    char stringa[] = "Mi chiamo Bob!";
    stringa[12]='h';              // modifica la stringa
    int i=0;                      // variabile contatore
    /* stampa caratteri fino a che la stringa non è terminata */
    while(stringa[i] != '\0'){
        printf("%c", stringa[i]);
        i++;
    }
}
```



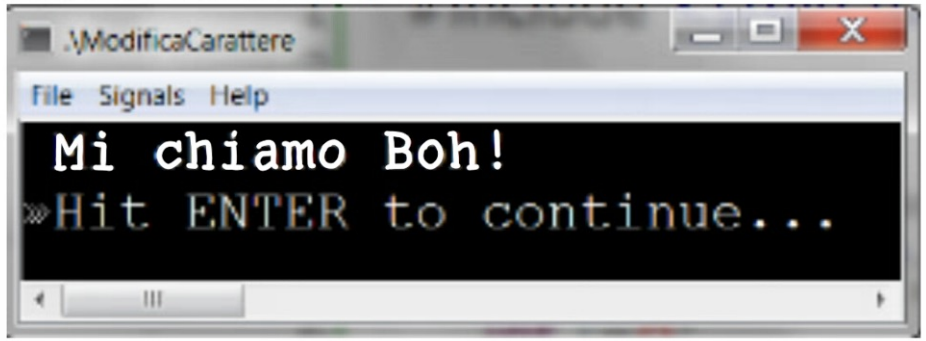
Osserva come il carattere delimitatore permetta di stampare una stringa senza conoscerne la lunghezza.

Stampa di stringhe

Oltre alla stampa carattere per carattere, utilizzando il carattere delimitatore, è possibile stampare una stringa con una sola istruzione *printf*, specificando il formato %s.

- E' allora la funzione *printf* a terminare la stampa quando viene raggiunto il carattere '\0'.

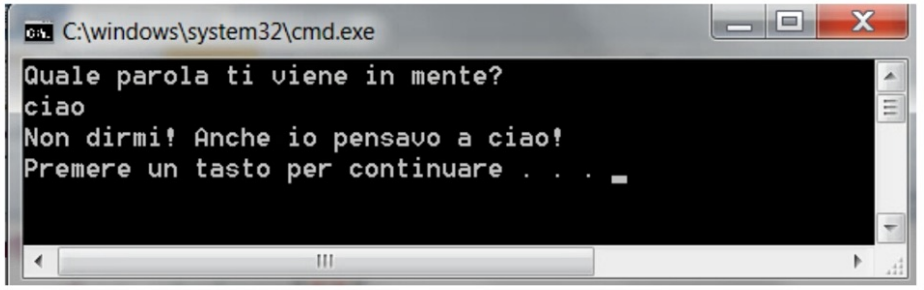
```
int main() {
   char stringa[] = "Mi chiamo Bob!";
   stringa[12]='h';              // modifica la stringa
   int i=0;                      // variabile contatore
   /* stampa caratteri fino a che la stringa non è terminata */
   printf("%s", stringa);
}
```



Lettura stringhe

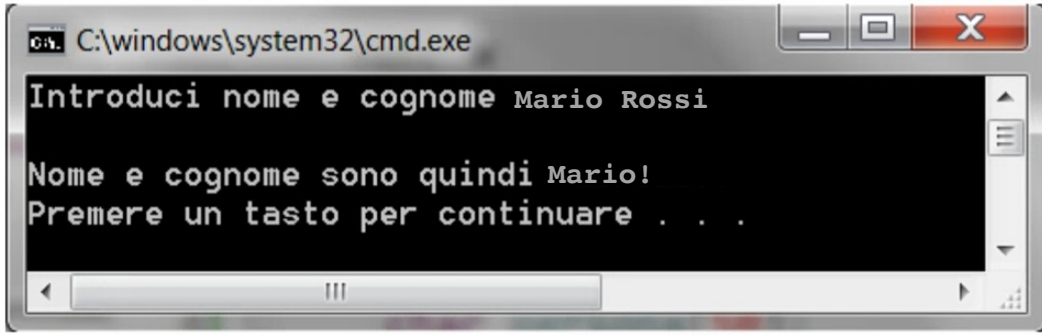
Può essere realizzata tramite la funzione *scanf* (nota: va dichiarato in precedenza un array di caratteri di dimensione fissa).

```
int main() {
    char stringa[50];
    printf("Quale parola ti viene in mente?\n");
    scanf("%s",stringa);
    printf("Non dirmi! Anche io pensavo a %s!\n", stringa);
}
```



Ma attenzione: *scanf* legge fino al primo spazio!

```
int main() {
    char persona[50];
    printf("Introduci nome e cognome\n");
    scanf("%s",persona);
    printf("Nome e cognome sono quindi %s!\n", persona);
}
```



fgets

La libreria stdio.h mette a disposizione una funzione fgets per la lettura di una sequenza di caratteri di una certa lunghezza, da un certo stream di input.

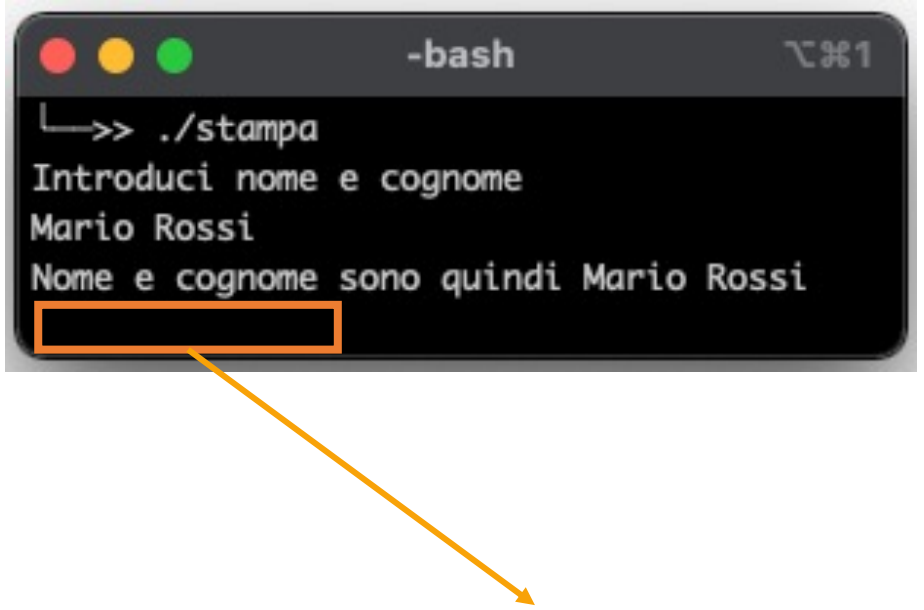
Il prototipo è char* fgets(char *str, int n, FILE *stream)

Permette di leggere una sequenza di caratteri da uno stream di input (indicare stdin per la tastiera) e di memorizzarli all'interno dell'array *str*.

- La funzione interrompe la lettura dei caratteri se è stata raggiunta la fine della linea (il carattere di ritorno a capo), o la fine del file (non si applica se lo stream di input è stdin), o sono stati letti n-1 caratteri.

```c
#include <stdio.h>


int main(){
char persona[50];
     printf("Introduci nome e cognome\n");
     fgets(persona,50,stdin);
     printf("Nome e cognome sono quindi %s\n",
     persona);
}
```



Attenzione: viene memorizzato nell'array anche il carattere newline che l'utente ha introdotto per inviare la linea.

Se vuoi escludere il carattere '\n', lo puoi fare manualmente o usare opportune funzioni nella libreria string.h

Funzioni su stringhe

Esiste una libreria, chiamata string, di funzioni predefinite per la gestione di stringhe che terminano con il carattere '\0'.

Per usare tali funzioni, bisogna includerne le dichiarazioni nel sorgente:

```
#include <string.h>
```

Funzioni su stringhe

La funzione strlen riceve una stringa come parametro e restituisce la lunghezza della stringa, ovvero il numero di caratteri che la compongono (il carattere null non viene contato).

Esempio:

```
printf("%d", strlen(""));
```

← stampa 0.

La funzione strcpy riceve due stringhe come parametri e copia la seconda nella prima

Esempio:

```
char str1[10] = "ciao cari";
char str2[10];
strcpy(str2,str1);
printf("%s\n",str2);
```

← stampa ciao cari.

Funzioni su stringhe

La funzione strlen può essere utilizzata per rimuovere eventuali caratteri newline ‘\n’ memorizzati dalla fgets.

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

int main(){
    char str[50];
    printf("Caro utente, inserisci una stringa:\n");
    fgets(str,50,stdin);
    str[strlen(str)-1] = '\0';
    printf("%s\n", str);
}
```

}

Altre risorse

- Bellini, Guidi: Linguaggio C — 13.1 – 13.3