Stringhe

- Sono sequenze di caratteri racchiusi tra doppi apici: `"abc"`
- Dal punto di vista dei tipi, una stringa è un vettore di caratteri: `char []`
- Il vettore contiene la sequenza di caratteri che forma la stringa, seguita dal *carattere speciale di fine stringa*: '\0'.
- **Esempio:**

```
char pippo[6]="pippo";
char stringa1[10] = {'p', 'i', 'p', 'p', 'o', '\0 ' };
char nonStringa1[2] = {'p', 'i'};
```

  Il terzo è un vettore di caratteri ma non una stringa (non termina con \0′).
- Attenzione: non bisogna confondere costanti di tipo stringa e costanti di tipo carattere.
  $\Longrightarrow$ `"a"` e `'a'` sono diversi. Il primo è un array di 2 elementi, i cui valori sono il carattere `'a'` ed il carattere speciale '\0'.

Le costanti di tipo stringa vengono trattate dal compilatore come puntatori a caratteri

- È possibile utilizzare *costanti di tipo stringa (stringhe* letterali) scritte tra doppio apice e senza scrivere esplicitamente il carattere di fine stringa. Ad esempio: `"pippo"`, `"corso di Informatica"`.
- **Esempio:**

```
char *p = "abc";
printf("%c%c\n", *p, *(p+1));
```

  provoca la stampa sullo schermo di `a` e `b`, poiché il puntatore `p` viene inizializzato con l'indirizzo del primo elemento dell'array *costante di* 4 caratteri utilizzato per la memorizzazione di `"abc"`.

Le stringhe costanti non possono essere modificate

Esempio:

```
char *pippo = "abc";
*pippo='x';
```

Non viene segnalato errore a tempo di compilazione, ma viene generato un errore a tempo di esecuzione! Questo è diverso da:

`char pippo[] = "pippo";` che è equivalente a
`char pippo[6] = "pippo";` che è equivalente a
`char pippo[] = {'p', 'i', 'p','p','o', '\0'}`

Nel primo caso la stringa constante `"pippo"` viene usata per inizializzare il vettore `pippo`. Quest'ultimo, tuttavia, non è un vettore costante e quindi può essere modificato:

$\Longrightarrow$ `*pippo = 'x'` (equivalente a `pippo[0]='x'`) è lecito.

La seguente dichiarazione, invece, NON è una stringa:

```
char nonStringa[] = {'a', 'b', 'c'}
```

Stampa di stringhe

Si utilizza la specifica di formato "%s", che provoca la stampa di tutti i caratteri fino al primo '\0' escluso.

Esempio:

```
printf("%s\n", pippo);
printf("%s\n", "pippo");
printf("%s\n", nonStringa);
```

Nell'ordine viene stampato:

```
pippo
pippo
abc????...
```

L'ultimo comando di stampa, in alcuni implementazioni, stampa caratteri casuali.

Lettura di stringhe

Si utilizza la specifica di formato "`%s`", che legge i caratteri fino a un carattere terminatore (spazio, tab e a capo) della stringa.

Esempio:

```
scanf("%s", stringa);
```

Se l'array di caratteri non ha elementi sufficienti, in alcune implementazioni, vengono sporcate le variabili allocate successivamente.

```
char stringa1[] = "abc";
char stringa2[4] ;
char stringa3[] = {'a', 'b', 'c'}
scanf("%s", stringa2);
```

Se inserisco una stringa molto lunga, per `stringa2` i caratteri possono essere scritti nello spazio di `stringa3`.

`Bus error` è il messaggio di errore tipico con le stringhe.

Manipolazione di stringhe

Per i motivi elencati sopra e dal momento che per la manipolazione valgono le stesse regole degli array e cioè:

- Non si può assegnare una stringa ad un'altra
- Non si possono confrontare le stringhe

È quindi più comodo e ragionevole usare le librerie:

- Libreria standard: `stdlib.h`
- Libreria per le stringhe: `string.h`
- Libreria per i caratteri: `ctype.h`