



# **Università degli Studi di Cassino e del Lazio Meridionale**

## **Corso di Fondamenti di Informatica**

### ***Tipi strutturati: Stringhe***

Anno Accademico 2016/2017

Francesco Tortorella

# Stringhe di caratteri

- La stringa è il tipo strutturato con cui vengono rappresentati gruppi di caratteri quali parole, nomi, frasi, ecc.
- La stringa è una sequenza di caratteri, di lunghezza variabile, ma limitata.
- Quindi, diversamente dall'array, la stringa non ha una lunghezza fissa.

# Array di caratteri e stringhe

- Il C++ implementa le stringhe come array di caratteri di cardinalità prefissata. In questo modo la lunghezza massima della stringa coincide con la cardinalità specificata nella definizione.
- La lunghezza variabile viene gestita attraverso un carattere speciale (carattere nullo '\0') che indica il termine della stringa.

```
char Stringa2[9] = "Mortimer"
```

**Definizione di una variabile stringa con  
inizializzazione**

Stringa2 [0]	M	
[1]	o	
[2]	r	
[3]	t	
[4]	i	
[5]	m	
[6]	e	
[7]	r	
[8]	\0	carattere nullo

(b)

# Definizione di una stringa

- Per definire una variabile stringa, è necessario specificare:
  - il nome della variabile stringa
  - il tipo degli elementi (**char** in questo caso)
  - il numero degli elementi presenti (cardinalità della stringa)

# Definizione di una stringa

- Nel decidere la dimensione massima della stringa (la cardinalità dell'array) è quindi necessario ricordare che un elemento deve essere riservato ad ospitare `\0`.
- Nell'inizializzazione è il compilatore a inserire automaticamente il carattere nullo al termine della stringa. Eventuali caratteri presenti nell'array dopo `'\0'` non vengono considerati.

# Accesso agli elementi di una stringa

- Come per gli altri array, è possibile accedere ai singoli caratteri di una stringa tramite indice:

```
char nome []="Pippo";  
int i;  
i=0;  
while (nome [i] != '\0' )  
    cout << nome [i++] << endl;
```

# Assegnazione tra stringhe

- Come per gli array non è possibile fare assegnazioni dirette tra variabili stringhe.
- Una possibilità per assegnare i caratteri di una stringa ad un'altra stringa è quella di fare una serie di assegnazioni tra elementi corrispondenti.
- Tuttavia, vista la frequenza di un'operazione del genere, esiste una funzione di libreria che la realizza direttamente:

```
char nome1[6]="Pippo";
```

```
char nome2[6];
```

```
strcpy(nome2, nome1);
```

**destinazione**



**sorgente**



# Input/Output di stringhe

- Contrariamente a quanto visto per gli array generici, è possibile fare operazioni di I/O direttamente sulle stringhe.
- È definito l'output di una stringa tramite l'operatore di inserimento <<:
  - i caratteri sono inseriti in ordine sullo stream di output finché non si incontra il carattere '\0'.
- È definito l'input di una stringa tramite l'operatore di estrazione >>:
  - sono estratti caratteri dallo stream di input e memorizzati nella stringa finché non viene incontrato un qualunque carattere *blank*: spazio, tab, newline.

# Input/Output di stringhe

```
#include <iostream>
using namespace std;
void main()
{
    char nome[30];
    cout << "stringa: "; cin >> nome;
    cout << "stringa fornita: " << nome << endl;
}
```

Esempio

```
stringa: pippo
stringa fornita: pippo

stringa: paolino paperino
stringa fornita: paolino
```

# Letture di stringhe con `cin.getline()`

- È possibile leggere stringhe contenenti spazi usando la funzione `cin.getline()` (funzione membro della classe `iostream` applicata all'oggetto `cin`).

- Sintassi:

`cin.getline(var_str, max_num, delim)`

dove:

`var_str` è la variabile stringa in cui inserire i caratteri estratti dall'input;

`max_num` è il numero massimo di caratteri da leggere;

`delim` è il carattere delimitatore da considerare per l'input

# Lettura di stringhe con `cin.getline()`

- La funzione estrae caratteri dallo stream di input finché o arriva al `(max_num-1)`-mo carattere o incontra il carattere `delim` (la prima delle due condizioni).
- Si assume che la dimensione della stringa sia `max_num`, comprendente anche il carattere `'\0'`: perciò si leggono `max_num-1` caratteri.
- Se viene omesso `delim`, viene assunto uguale a `'\n'`.

# Letture di stringhe con `cin.getline()`

```
#include <iostream>
using namespace std;
```

```
int main(int argc, char** argv) {
    char s[32], t[64];

    cin.getline(s, 32);

    cin.getline(t, 64, '*');
}
```

Legge al più 31  
caratteri, fermandosi  
eventualmente al '\n'



Legge al più 63  
caratteri, fermandosi  
eventualmente al '\*'



# Lettura di stringhe con `gets ()`

- Per leggere una stringa che contiene spazi, un'altra possibilità è utilizzare la funzione `gets ()` disponibile nella libreria C e dichiarata in `cstdio`.
- Sintassi:  
`gets (str)`  
`str` è la variabile stringa in cui inserire i caratteri estratti dall'input
- Limitazioni:  
`gets` non controlla la lunghezza massima della stringa.

# Letture di stringhe con gets ()

```
#include <iostream>
#include <cstdio>

using namespace std;

int main(int argc, char** argv) {
    /* gets example */
    char string [32];

    cout << "Fornire nome e cognome: ";
    gets(string);
    cout << "Tu sei : " << string << endl;

    return (EXIT_SUCCESS);
}
```

# Stringhe come parametri

- Come gli array generici, anche le stringhe sono passate per riferimento.
- Come parametro formale, una stringa si indica con tipo, identificatore e parentesi quadre: `char s[]`. Non si inserisce l'&.
- Come parametro effettivo va fornito il nome della stringa, senza specificare altro.

# La libreria <cstring>

- Molte funzioni per la elaborazione e manipolazione di stringhe sono disponibili nella libreria C e dichiarate in <cstring>. Per utilizzarle, va inserito `#include <cstring>`
- Alcuni esempi di funzioni:

**strcpy()** `char*` strcpy(**char** dest[], **char** sorg[]);  
Copia la stringa sorg nella stringa dest. Restituisce dest.

**strlen()** `size_t` strlen (**char** s[]);  
Restituisce la lunghezza della stringa s.

**strcat()** `char*` strcat(**char** dest[], **char** sorg[]); Aggiunge una copia della stringa sorg alla fine di dest. Restituisce dest.

**strcmp()** `int` strcmp(**char** s[], **char** t[]);  
Confronta le stringa s e t e restituisce:  
0 se s == t  
<0 se s < t  
>0 se s > t