

Titolo unità didattica: Stringhe ed elaborazione di testi [09]

Titolo modulo : Stringhe in C [03-C]

Proprietà delle stringhe C

Argomenti trattati:

- ✓ rappresentazione di stringhe in C
- ✓ function C per lettura e visualizzazione di stringhe
- ✓ function in C per operazioni di base su stringhe
- ✓ array di puntatori al tipo **char**

Prerequisiti richiesti: AP-03-05-C, AP-05-04-C, AP-07-08-C

stringhe di caratteri in C

in C il tipo stringa non è direttamente definito

- ✓ il tipo **stringa** in C è realizzato come un **array 1D** di elementi di tipo **char**
- ✓ la **fine** della sequenza di caratteri della stringa è specificata esplicitamente dal **carattere di fine stringa**: `'\0'`

`'\0'` : carattere nullo, **NULL**, codice ASCII **00000000**

stringhe di caratteri in C

in C il tipo stringa non è direttamente definito

- ✓ il tipo **stringa** in C è realizzato come un **array 1D** di elementi di tipo **char**
- ✓ la **fine** della sequenza di caratteri della stringa è specificata esplicitamente dal **carattere di fine stringa**: `'\0'`



considerare una stringa come se avesse **lunghezza variabile**, delimitata da `\0`, entro una **lunghezza massima** determinata dal **size** dell'array

stringhe di caratteri in C

in C il tipo stringa non è direttamente definito

- ✓ il tipo **stringa** in C è realizzato come un **array 1D** di elementi di tipo **char**
- ✓ la **fine** della sequenza di caratteri della stringa è specificata esplicitamente dal **carattere di fine stringa**: `'\0'`



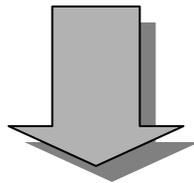
il **size** dell'array deve tener conto anche dello spazio necessario per memorizzare il **carattere di fine stringa**

stringhe e array in C

```
char nome_Corso_Laurea[12];  
char testo[1000];  
char pagina_libro[1600];  
char gene[250];  
char genoma[100000000];
```

Informatica

11 caratteri



nome_Corso_Laurea

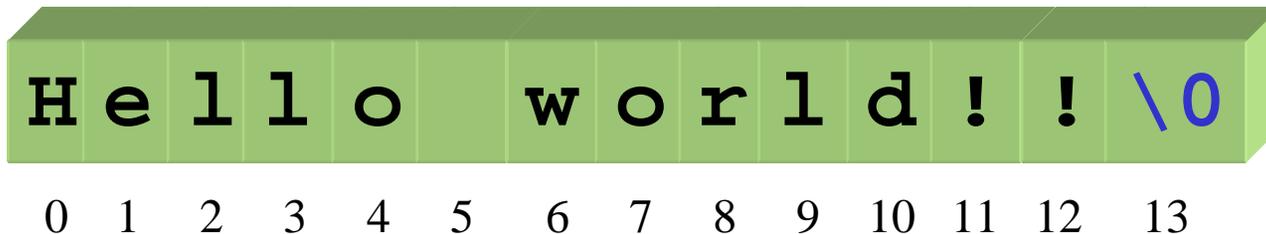
size 12

costanti stringa in C

una **costante stringa** è specificata dalla sequenza di caratteri **racchiusa tra doppi apici**

`"Informatica"`

`"Hello world!!"`



una **costante stringa**, come il nome di un array 1D, è trattata dal compilatore C come un **puntatore costante**, che punta alla cella di memoria del **primo carattere** della stringa (**indirizzo base della stringa**)

dichiarazione-inizializzazione di stringhe in C

```
char nome_Corso_Laurea[12]="Informatica";
```

```
char nome_Corso_Laurea[]="Informatica";
```

```
char nome_Corso_Laurea[]={ 'I', 'n', 'f',  
'o', 'r', 'm', 'a', 't', 'i', 'c', 'a', '\0' };
```

```
char *nome_Corso_Laurea="Informatica";
```



un **puntatore** viene inizializzato a un **indirizzo**

costanti *stringa* e costanti *carattere* in C

differenza tra costanti *stringa* e costanti *carattere*

```
char d = 'r';  
char s[] = "r";  
char *p = "r";
```

variabile scalare e costante carattere

array di due elementi e costante stringa

array di due elementi e costante stringa

```
d = 'a';
```

assegnazione **corretta**

```
s[0] = 'a';
```

assegnazione **corretta**

```
s = "a";
```

assegnazione **non** corretta

```
*p = "a";
```

assegnazione **non** corretta

stringhe e puntatori a **char** in C

```
char *pnome = "Giulio Rossi";
```

il puntatore **pnome** punta al primo carattere della costante stringa, ovvero al primo elemento dell'array (**indirizzo base**) di caratteri in cui è memorizzata la costante stringa

```
char *pnome;
```

```
*pnome = "Giulio Rossi";
```

ERRORE

in C **non esistono** operatori che trattano una stringa come una unità

```
char *pnome;
```

```
pnome = "Giulio Rossi";
```

OK

questa assegnazione **non copia** la stringa costante, ma **coinvolge solo** i puntatori (**gli indirizzi**)

- ✓ viene dichiarato un puntatore
- ✓ si alloca memoria per la costante stringa
- ✓ l'indirizzo base della stringa viene assegnato al puntatore

```
char *pnome;  
pnome = "Giulio Rossi";
```

sono equivalenti

```
char *pnome = "Giulio Rossi";
```

- ✓ si alloca memoria per la costante stringa
- ✓ viene dichiarato un puntatore
- ✓ l'indirizzo base della stringa viene assegnato al puntatore

```
char nome[] = "Giulio Rossi";
```

- ✓ si alloca memoria per un array adeguata almeno a contenere la costante stringa
- ✓ l'array viene inizializzato con la costante stringa
- ✓ **è possibile copiare un'altra stringa nell'array**

```
pnome = &nome[0];  
pnome++;
```

↓
punta a `nome[1]`,
secondo carattere della
stringa

```
nome[1];  
*pnome;
```

notazione standard

notazione a puntatore

visualizzazione di stringhe

```
char nome_Corso_Laurea[12]="Informatica";  
printf("%s",nome_Corso_Laurea);
```

```
char *nome_Corso_Laurea="Informatica";  
printf("%s",nome_Corso_Laurea);
```

ATTENZIONE

il codice di formato `%s` (codice di formato stringa) richiede che il corrispondente argomento nella chiamata della `printf` sia un **puntatore**

```
printf("%s",nome_Corso_Laurea+2);
```

formatica

—

function C per l'I/O di caratteri singoli

`main` che conta il numero di caratteri digitati sulla tastiera, fino al carattere di fine immissione (**EOF**)

```
#include <stdio.h>
void main()
{
    int c, i=0;
    while ((c = getchar()) != EOF)
        i++;
    printf("numero di caratteri: %d", i);
    putchar('\n');
}
```

EOF è **Ctrl+z** in Microsoft VisualC++ (in `stdio.h`)

function C per l'I/O di stringhe

libreria **stdio**

```
char stringa[100];  
stringa = gets();
```

fino a nuova linea o EOF

```
char stringa[100];  
gets(stringa);
```

```
puts(stringa);
```

la function **gets()**
ha un parametro di tipo puntatore a **char**
e restituisce un puntatore a **char**

la function **puts()**
ha un parametro di tipo puntatore a **char**

```
#include <stdio.h>

void main()
{
    int i;
    char *p="informatica";
    char a[20]="calcolo";
    puts(p);
    puts(a);
    p = "applicata";
    for(i=0;i<12;i++)
        a[i] = "numerico"[i];
    puts(p);
    puts(a);
}
```

```
informatica
calcolo
applicata
numerico
_
```

```
#include <stdio.h>

void main()
{
    int i;
    char *p="informatica";
    char a[]="calcolo";
    puts(p);
    puts(a);
    *(p+1) = 'N';
    for(i=0;i<9;i++)
        a[i] = "numerico"[i];
    puts(p);
    puts(a);
}
```

function in C per operazioni su stringhe:
lunghezza di una stringa

```
int lunghezza_stringa(char *s)
{
    int i;
    for (i=0; *s != '\0'; i++)
        s++;
    return i;
}
```

libreria `string` :

```
unsigned int strlen(char *str);
```

function in C per operazioni su stringhe:
concatenazione di due stringhe

```
void cat_stringa(char s[], char t[])
{
    int i, j;

    j = lunghezza_stringa(s);

    for (i=0; t[i] != '\0'; i++)
        s[j++] = t[i];

    s[j] = '\0';
}
```

libreria **string** :

```
void strcat(char *str1, char *str2);
```

function in C per operazioni su stringhe:
concatenazione di sottostringhe

```
void catN_stringa(char s[], char t[], int n)
{
    int i, j;

    j = lunghezza_stringa(s);

    for (i=0; t[i] != '\\0' && i < n; i++)
        s[j++] = t[i];

    s[j] = '\\0';
}
```

libreria **string** :

```
void strncat(char *str1, char *str2, int n);
```

function in C per operazioni su stringhe:
copia di una stringa

```
void copia_stringa(char s[], char t[])
{
    int i=0;
    for (i=0; t[i] != '\0'; i++)
        s[i] = t[i];
    s[i] = '\0';
}
```

libreria `string` :

```
char *strcpy(char *str1, char *str2);
```

function in C per operazioni su stringhe:
copia di una sottostringa

```
void copiaN_stringa(char s[], char t[], int n)
{
    int i=0;
    for (i=0; t[i] != '\0' && i < n; i++)
        s[i] = t[i];
}
```

libreria **string** :

```
char *strncpy(char *str1, char *str2, int n);
```

function in C per operazioni su stringhe:
confronto tra due stringhe

```
int confronto_stringhe(char s[], char t[])
{
    int i;
    if(lunghezza_stringa(s) != lunghezza_stringa(t))
        return 0;
    for (i=0; s[i] != '\0'; i++)
        if (s[i] != t[i])
            return 0;
    return 1;
}
```

libreria `string` :

```
int strcmp(char *str1, char *str2);
```

function in C per operazioni su stringhe:
confronto tra due sottostringhe

```
int confrontoN_stringhe(char s[], char t[], int n)
{
    int i;
    for (i=0; s[i] != '\0' && i < n ; i++)
        if (s[i] != t[i]) return 0;
    return 1;
}
```

libreria **string** :

```
int strncmp(char *str1, char *str2, int n);
```

function in C per operazioni su stringhe:
ricerca di un carattere in una stringa

```
int trova_carattere(char s[], char t)
{
    int i;
    for (i=0; s[i] != '\0'; i++)
        if (s[i]== t)
            return 1;
    return 0;
}
```

libreria `string` :

```
char *strrchr(char *str1, char ch);
```

array di stringhe

in molte applicazioni si usano collezioni di stringhe:

- elenco di nominativi
- sequenze di domande
- raccolte di delibere
-



- ✓ **array 2D di char** (array di array di char)
- ✓ **array di puntatori a char**

- elenco di **10 nominativi**
- ciascun nominativo è lungo al più **11 caratteri**

```
char elenco_a2D[10][12]={ "Alberto",  
                           "Maria",  
                           "Luisa",  
                           "Remigio",  
                           "Renzo",  
                           "Ugo",  
                           "Beppe",  
                           "Dino",  
                           "Laura",  
                           "Michele",  
                           };
```

array 2D di char
(array di array di
char)

array di stringhe

esempio

memoria allocata e inizializzata

A	l	b	e	r	t	o	\0	0	0	0	0
M	a	r	i	a	\0	0	0	0	0	0	0
L	u	i	s	a	\0	0	0	0	0	0	0
R	e	m	i	g	i	o	\0	0	0	0	0
R	e	n	z	o	\0	0	0	0	0	0	0
U	g	o	\0	0	0	0	0	0	0	0	0
B	e	p	p	e	\0	0	0	0	0	0	0
D	i	n	o	\0	0	0	0	0	0	0	0
L	a	u	r	a	\0	0	0	0	0	0	0
M	i	c	h	e	l	e	\0	0	0	0	0

elenco_a2D[10][12]

array di stringhe

`elenco_a2D[i][j]` è il j -simo carattere dell' i -sima riga

`elenco_a2D[i]` è l' i -simo array di `char`

`elenco_a2D[i]` è il puntatore all' i -sima riga

```
if (confronto_stringhe
    (elenco_a2D[2],elenco_a2D[3]))
.....
if(strcmp(elenco_a2D[2],elenco_a2D[3]))
.....
```

- elenco di **10 nominativi**
- ciascun nominativo è lungo al più **11 caratteri**

```
char *elenco_p1D[10]={"Alberto",  
                      "Maria",  
                      "Luisa",  
                      "Remigio",  
                      "Renzo",  
                      "Ugo",  
                      "Beppe",  
                      "Dino",  
                      "Laura",  
                      "Michele",  
                      };
```

array di
puntatori a char

esempio

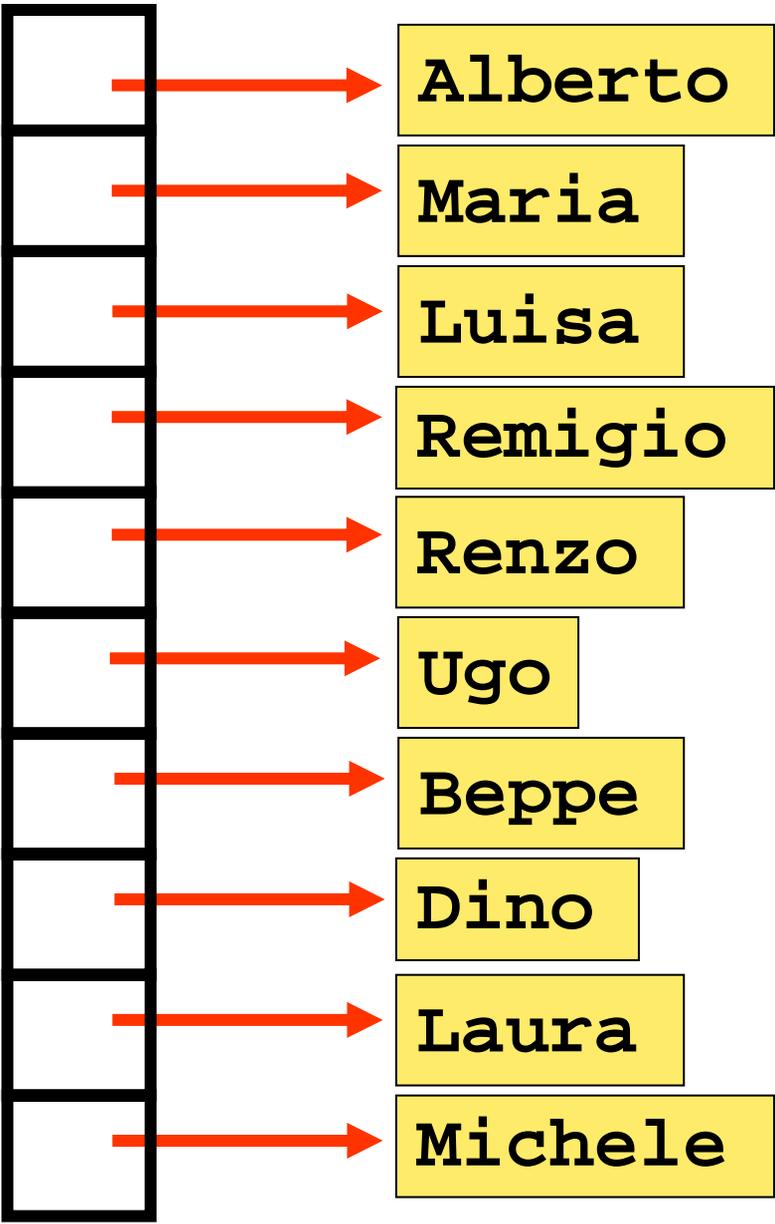
memoria allocata e inizializzata

A	l	b	e	r	t	o	\0
M	a	r	i	a	\0		
L	u	i	s	a	\0		
R	e	m	i	g	i	o	\0
R	e	n	z	o	\0		
U	g	o	\0				
B	e	p	p	e	\0		
D	i	n	o	\0			
L	a	u	r	a	\0		
M	i	c	h	e	l	e	\0

array frastagliato
di stringhe

*elenco_p1D[10]

array di puntatori a char



stringhe

elenco_p1D

array di stringhe

`elenco_p1D[i]` è il puntatore all'*i*-sima costante stringa

`elenco_p1D[i]` è l'*i*-sima costante stringa

```
if(confronto_stringhe
    (elenco_p1D[2],elenco_p1D[3]))
.....
if(strcmp(elenco_p1D[2],elenco_p1D[3]))
.....
```

array di stringhe

- ✓ array 2D di `char` (array di array di `char`)
- ✓ array di puntatori a `char`

differenze

- le stringhe cui si accede con `elenco_a2D[i]` sono modificabili
- le stringhe cui si accede con `elenco_p1D[i]` sono costanti stringhe e **non** sono modificabili
- lo spazio di memoria allocato è diverso

array di puntatori a **char**

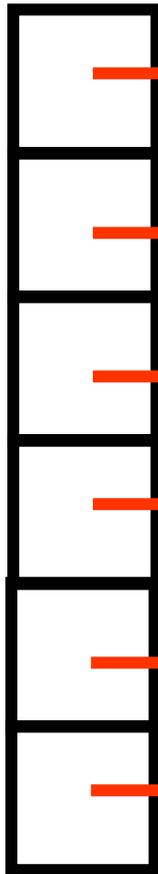
```
#include <stdio.h>
#include <string.h>
void main(){
    int i;
    char *elenco_nomi[]={ "Giulio Giunta",
        "Raffaele Montella", "Umberto Scafuri",
        "Angelo Ciaramella", "Mariarosaria Rizzardi",
        "Annalisa Amadori"};
    char elenco_Corsi[6][20], array_di_char[10];
    printf("\n uso di scanf per inserire le
        componenti di array di char\n");
    for(i=0; i<10; i++)
    {
        printf(" inserire %d-mo carattere\n", i);
        fflush(stdin);
        scanf("%c", &array_di_char[i]);
        fflush(stdin);
    }
}
```

```
printf("\n l'array di char inserito con
scanf\n");
for(i=0;i<10;i++)
    printf("%c",array_di_char[i]);

printf("\n inserire i corsi\n");
for(i=0;i<6;i++)
    gets(elenco_Corsi[i]);

printf("\n elenco dei corsi inseriti\n");
for(i=0;i<6;i++)
    puts(elenco_Corsi[i]);

printf("\n elenco dei nomi\n");
for(i=0;i<6;i++)
    printf("%s\n",elenco_nomi[i]);
}
```



Giulio Giunta

Raffaele Montella

Umberto Scafuri

Angelo Ciaramella

Mariarosaria Rizzardi

Annalisa Amadori

`elenco_Nomi`

stringhe