

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

Titolo modulo : La libreria `string` del C [04-C]

Proprietà e utilizzo delle principali function C in `string`

Argomenti trattati:

- ✓ libreria di function C per il trattamento delle stringhe: la libreria `string`
- ✓ specifiche e funzionalità di alcune function in `string`

Prerequisiti richiesti: AP-09-03-C

strcat()

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

- ✓ concatena una copia della stringa (puntata da) **str2** alla stringa (puntata da) **str1** e chiude la nuova sequenza di caratteri di **str1** con il carattere nullo (terminatore di stringa)
- ✓ il primo carattere di **str2** si sovrappone al terminatore originale di **str1**
- ✓ la stringa **str2** rimane inalterata
- ✓ la function restituisce **str1** anche via **return**
- ✓ non si effettua alcun controllo sui size degli array (è compito del programmatore assicurarsi che la stringa **str1** abbia size sufficiente per contenere i caratteri originari di **str1** e di **str2** e il terminatore)

strcat(): esempio

```
#include <string.h>
#include <stdio.h>
void main()
{
    char s1[80], s2[80];
    gets(s1);
    gets(s2);
    strcat(s2, s1);
    printf("%s", s2);
}
```

- concatena la prima stringa letta da tastiera alla seconda stringa letta da tastiera e visualizza la concatenazione (**s2**)

strchr()

```
char *strchr(char *str, char chiave)
```

- ✓ restituisce un puntatore all'indirizzo del primo carattere della stringa (puntata da) `str` che risulta uguale al valore della variabile `chiave`
- ✓ se non viene trovata alcuna corrispondenza, la function restituisce un puntatore nullo

strchr() : esempio

```
#include <string.h>
#include <stdio.h>
void main()
{
    char *p;
    p = strchr("Nel mezzo", 'm');
    printf("%c\n", *p);
}
```

- `p` punta al carattere `m` della costante stringa primo argomento di chiamata
- la `printf` produce la visualizzazione di `m`

strcmp ()

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

- ✓ confronta, secondo le regole lessicografiche, le due stringhe (puntate da) `str1`, `str2` e restituisce un intero il cui valore è
 - **minore di zero** se `str1` è minore di `str2`
 - **zero** se `str1` è uguale a `str2`
 - **maggiore di zero** se `str1` è maggiore di `str2`

strcmp() : esempio

```
void main() /* controllo password */
{
    char s[80];
    while(1) ← ciclo potenzialmente infinito
    {
        printf("Inserire la password: ");
        gets(s);
        if(strcmp(s, "pass") != 0)
            printf("password errata\n");
        else
            break;
    }
    printf("password ok, si puo' continuare\n");
    .....
}
```

strcpy()

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

- ✓ copia la stringa (puntata da) `str2` in `str1`
- ✓ la sequenza di caratteri in `str2` deve essere chiusa dal carattere nullo (terminatore)
- ✓ la function restituisce `str1` anche via `return`
- ✓ se gli argomenti di chiamata per `str1` e `str2` puntano alla stessa stringa, il comportamento della function non è definito

strcpy() : esempio

```
...  
char str[80];  
strcpy(str, "Salve");  
...
```

- copia la costante stringa "Salve" in `str`

strlen()

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

- ✓ restituisce la lunghezza della stringa (puntata da **str**)
- ✓ la stringa deve essere chiusa dal carattere nullo (terminatore), che **non viene conteggiato** ai fini della determinazione della lunghezza

strlen(): esempio

...

```
strcpy(s, "Salve");
```

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

...

- produce la visualizzazione del valore **5**, la lunghezza della stringa (puntata da) **s**

strlwr() e strupr()

```
char *strlwr(char *str)
```

✓ converte in minuscolo la stringa (puntata da) `str`

```
char *strupr(char *str)
```

✓ converte in maiuscolo la stringa (puntata da) `str`

strlwr() estrupr(): esempio

```
#include <string.h>
#include <stdio.h>
void main()
{
    char *s1 = "PROVA STRMINUSCOLO";
    char *s2 = "prova strmaiuscolo";
    strlwr(s1);
    printf("%s  %s\n", s1,strupr(s2));
}
```

prova strminuscolo PROVA STRMAIUSCOLO

strncat()

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

- ✓ concatena al più **n** caratteri della stringa (puntata da) **str2** alla stringa **str1** e chiude la sequenza di caratteri di **str1** con il carattere nullo (terminatore)
- ✓ il primo carattere di **str2** si sovrappone al terminatore originale di **str1**
- ✓ la stringa **str2** rimane inalterata
- ✓ function restituisce **str1** anche via **return**
- ✓ non si effettua alcun controllo sui size degli array (è compito del programmatore assicurarsi che la stringa **str1** abbia size sufficiente per contenere i caratteri originari di **str1** e di **str2** e il terminatore)

strncat () : esempio

```
#include <string.h>
#include <stdio.h>
void main()
{
    char s1[80], s2[80];
    int n;
    gets(s1);
    gets(s2);
    n = 79 - strlen(s2);
    strncat(s2, s1, n);
    printf("%s", s2);
}
```

- concatena i primi **n** caratteri della prima stringa letta da tastiera (**s1**) alla seconda stringa letta da tastiera e visualizza la concatenazione (**s2**)

strncmp ()

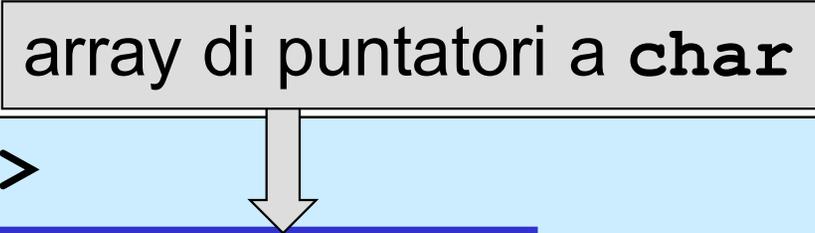
```
int strncmp(char *str1, char *str2, int n)
```

✓ confronta, secondo le regole lessicografiche, al più **n** caratteri delle due stringhe (puntate da) **str1**, **str2** e restituisce un intero il cui valore è

- **minore di zero** se la sottostringa di **str1** è minore della sottostringa di **str2**
- **zero** se la sottostringa di **str1** è uguale alla sottostringa di **str2**
- **maggiore di zero** se la sottostringa di **str1** è maggiore della sottostringa di **str2**

strncmp () : esempio

array di puntatori a char



```
#include <string.h>
void main(int argc, char *argv)
{
    if (strncmp(argv[1], argv[2], 8) == 0)
        printf("I nomi sono identici\n");
}
```

- confronta gli 8 caratteri iniziali della **prima stringa** e della **seconda stringa** dopo il comando al sistema operativo (modalità linea di comando) di esecuzione del programma C

strncpy()

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

- ✓ copia al più **n** caratteri della stringa (puntata da **str2**) al posto dei primi **n** caratteri di **str1**
- ✓ la function restituisce **str1** anche via **return**
- ✓ non viene inserito il carattere nullo (terminatore)

strncpy(): esempio

```
...  
char str1[180], str2[45];  
gets(str1);  
strncpy(str2, str1, 44);  
...
```

- copia i primi 44 caratteri della stringa `str1` nella stringa `str2`

strstr()

```
char *strstr(char *str1, char *str2)
```

- ✓ restituisce un puntatore all'indirizzo del primo carattere della sottostringa della stringa (puntata da) **str1** che risulta uguale alla stringa **str2**
- ✓ se non viene trovata alcuna corrispondenza, la function restituisce un puntatore nullo

strstr(): esempio

```
#include <string.h>
#include <stdio.h>
void main()
{
    char *p;
    p = strstr("Sono una prova", "una");
    printf("%c\n", *p);
}
```

- `p` punta al carattere `u` della stringa primo argomento di chiamata
- la `printf` visualizza il carattere `u` (di `"Sono una prova"`)