Ripasso: cosa è una stringa in C?

Esempio: la stringa **abcd**

Indirizzo di memoria	1000	1001	1002	1003	1004
Contenuto Carattere	a	b	c	d	\0
Valore(decimale)	97	98	99	100	0

```
char *p;
p = "abcd";
strlen( p )= 4;
char s[5] = "abcd";
strlen( s )= 4;
char t[5] = { 'a', 'b', 'c', 'd', '\0' };
strlen( t )= 4;
```

Ripasso: come si manipolano le stringhe in C?

```
char s[10];
s = "0123456789";
```

NON VA!! Sono due puntatori ad aree di memoria diversa

```
char *s = "0123456789";
char *t = s;
```

Sono due puntatori alla stessa area di memoria

```
char *s,*t;
s = "0123456789";
t = malloc(11*sizeof(char));
t = strcpy(t,s);
```

Copia il contenuto di sorg in dest e restituisce dest

```
char *strcpy(char *dest, const char *sorg);
```

Possibili implementazioni della copia

```
void stringcopy(char *s1, char *s2)
{* Attenzione: lo spazio allocato per s1 deve bastare a contenere s2.
Prec: *s1 != NULL && s2 != NULL
Postc: s1 conterrà un copia di s2*/
for (;*s2 !='\0';s1++,s2++)
     *s1=*s2;
     *s1='\0';
}
```


```
void stringCopy(char *s1, char *s2)
{* Attenzione: lo spazio allocato per s1 deve bastare a contenere s2.
Prec: *s1 != NULL && s2 != NULL
Postc: s1 conterrà un copia di s2*/
while (*s1++ = *s2++)
;}
```

Possibili implementazioni (una ricorsiva!) del calcolo della lunghezza

```
int NumCar (char *s)
{/*prec: s != NULL;
Postc: restituisce il numerodi caratteri ( != \0) in s*/
 if (*s == '\0')
                return 0;
      else
                return (1+NumCar(s+1));
}
```

```
int strLen (char *s)
{/*prec: s != NULL;
Postc: restituisce il numerodi caratteri ( != \0) in s*/
 char * p = s;
 while (*p!= '\0') p++;
 return p-s;
}
```

Ripasso: come si manipolano le stringhe in C?

```
char s[N], t[N];
...
If ( s == t ) { .. }
else { ... };
```

NON VA!! Sono due puntatori ad aree di memoria diversa

```
char s[N], t[N];
...
If ( strcmp(s,t )==0) { .. }
else { ... };
```

Se s e t sono uguali strcmp(s,t) dà 0, un valore <0 se s precede nell'ordine lessicografico t, un valore >0 altrimenti

```
int strcmp(const char *string1,const char *string2)
```

Possibile implementazione del confronto

```
int stringcmp(char *s1, char *s2)
/*confronta s1 e s2.
Prec: s1 != NULL && s2 !=  NULL
Postc: restituisce -1 se s1 è minore in ordine lessicografico di s2, 0 se
    sono uguali, 1 altrimenti  */
{while (*s1 == *s2)
    {if (*s1=='\0') return 0;
    s1++;
    s2++;
    }
if (*s1 == '\0')return -1;
else if (*s2 == '\0')return 1;
       else
       return (((*s1 - *s2)>0)?1:0);
}
```

implementazione alternativa del confronto

```
int stringConfr(const char *s1, const char *s2)
{/*prec: s1 != NULL && s2 != NULL
postc: restituisce 1 se sono uguali, 0 altrimenti*/
for (; *s1 && *s2; s1++,s2++)
        if (*s1 != *s2) return 0;
return (*s1 == *s2);
}
```

Altre funzioni di libreria sulle stringhe

Per fare una copia di una stringa o confrontare due stringhe è più SICURO utilizzare:

```
int strncmp(const char *s1, char *s2, size_t n);
```

che confronta i primi n caratteri delle due stringhe, restituisce 0 se sono uguali, un valore negativo se s1<s2, uno positivo se s1>s2

e

```
char *strncpy(const char *dest,const char *sorg, size_t n);
```

che copia i primi n caratteri di sorg in dest. Restituisce dest, riempie dest con il terminatore se sorg ha meno di n caratteri.

Ma bisogna fare attenzione al terminatore di stringa!