Carlo Gaibisso
Bruno Martino

Programmazione e Laboratorio di Programmazione

Lezione XI

Le stringhe

Carlo Gaibisso
Bruno Martino

Premessa:

- **Il tipo char**

  Dimensione: **1 byte**

  Range: **da 0 a 255**

  Definizione: **char nome$_{variabile}$;**

- **Costanti di tipo char:**

  carattere racchiuso tra apici, equivale al al codice ASCII per il carattere

  Esempio:
  - **‘S’ (codice ASCII: 83)**
  - **‘c’ (codice ASCII: 67)**
  - **‘,’ (codice ASCII: 44)**
  - **‘+’ (codice ASCII: 43)**

Carlo Gaibisso
Bruno Martino

La tabella dei codici ASCII

Tabella dei codici ASCII

Char	Dec	Oct	Hex	Char	Dec	Oct	Hex	Char	Dec	Oct	Hex	Char	Dec	Oct	Hex
(nul)	0	0000	0x00	(sp)	32	0040	0x20	@	64	0100	0x40	`	96	0140	0x60
(soh)	1	0001	0x01	!	33	0041	0x21	A	65	0101	0x41	a	97	0141	0x61
(stx)	2	0002	0x02	"	34	0042	0x22	B	66	0102	0x42	b	98	0142	0x62
(etx)	3	0003	0x03	#	35	0043	0x23	C	67	0103	0x43	c	99	0143	0x63
(eot)	4	0004	0x04	$	36	0044	0x24	D	68	0104	0x44	d	100	0144	0x64
(enq)	5	0005	0x05	%	37	0045	0x25	E	69	0105	0x45	e	101	0145	0x65
(ack)	6	0006	0x06	&	38	0046	0x26	F	70	0106	0x46	f	102	0146	0x66
(bel)	7	0007	0x07	'	39	0047	0x27	G	71	0107	0x47	g	103	0147	0x67
(bs)	8	0010	0x08	(	40	0050	0x28	H	72	0110	0x48	h	104	0150	0x68
(ht)	9	0011	0x09	)	41	0051	0x29	I	73	0111	0x49	i	105	0151	0x69
(nl)	10	0012	0x0a	*	42	0052	0x2a	J	74	0112	0x4a	j	106	0152	0x6a
(vt)	11	0013	0x0b	+	43	0053	0x2b	K	75	0113	0x4b	k	107	0153	0x6b
(np)	12	0014	0x0c	,	44	0054	0x2c	L	76	0114	0x4c	l	108	0154	0x6c
(cr)	13	0015	0x0d	-	45	0055	0x2d	M	77	0115	0x4d	m	109	0155	0x6d
(so)	14	0016	0x0e	.	46	0056	0x2e	N	78	0116	0x4e	n	110	0156	0x6e
(si)	15	0017	0x0f	/	47	0057	0x2f	O	79	0117	0x4f	o	111	0157	0x6f
(dle)	16	0020	0x10	0	48	0060	0x30	P	80	0120	0x50	p	112	0160	0x70
(dc1)	17	0021	0x11	1	49	0061	0x31	Q	81	0121	0x51	q	113	0161	0x71
(dc2)	18	0022	0x12	2	50	0062	0x32	R	82	0122	0x52	r	114	0162	0x72
(dc3)	19	0023	0x13	3	51	0063	0x33	S	83	0123	0x53	s	115	0163	0x73
(dc4)	20	0024	0x14	4	52	0064	0x34	T	84	0124	0x54	t	116	0164	0x74
(nak)	21	0025	0x15	5	53	0065	0x35	U	85	0125	0x55	u	117	0165	0x75
(syn)	22	0026	0x16	6	54	0066	0x36	V	86	0126	0x56	v	118	0166	0x76
(etb)	23	0027	0x17	7	55	0067	0x37	W	87	0127	0x57	w	119	0167	0x77
(can)	24	0030	0x18	8	56	0070	0x38	X	88	0130	0x58	x	120	0170	0x78
(em)	25	0031	0x19	9	57	0071	0x39	Y	89	0131	0x59	y	121	0171	0x79
(sub)	26	0032	0x1a	:	58	0072	0x3a	Z	90	0132	0x5a	z	122	0172	0x7a
(esc)	27	0033	0x1b	;	59	0073	0x3b	[	91	0133	0x5b	{	123	0173	0x7b
(fs)	28	0034	0x1c	<	60	0074	0x3c	\	92	0134	0x5c	|	124	0174	0x7c
(gs)	29	0035	0x1d	=	61	0075	0x3d	]	93	0135	0x5d	}	125	0175	0x7d
(rs)	30	0036	0x1e	>	62	0076	0x3e	^	94	0136	0x5e	~	126	0176	0x7e
(us)	31	0037	0x1f	?	63	0077	0x3f	_	95	0137	0x5f	(del)	127	0177	0x7f

Carlo Gaibisso
Bruno Martino

Le stringhe in memoria

- **Stringa:**

  ogni sequenza di caratteri memorizzati in locazioni contigue di memoria e terminata dal carattere ‘\0’, detto carattere di fine stringa

‘s’	x
‘t’	x
‘r’	x
‘i’	x
‘n’	x
‘g’	x
‘a’	x
‘\0’	x
	x
	x

Definizione di una stringa

- **Definizione a run-time:**

  char *nome$_{stringa}$=(char *) malloc((espr+1) * sizeof(char));

  con espr espressione che identifica la lunghezza massima della stringa ‘\0’ escluso

- **Definizione statica:**

```
/* definizione della costante simbolica per la lunghezza
** massima della stringa,‘\0’ escluso */
#define STR_LENGTH 80

char nome_stringa[STR_LENGTH+1];
```

Carlo Gaibisso
Bruno Martino

I/O di stringhe

- **Acquisizione:**

  scanf("%s", dst);

  con dst espressione di tipo char * e %s specificatore di formato per le stringhe

- **Modifiche allo stato della memoria:**

  1. **elimina dallo stream di input qualunque sequenza iniziale di caratteri in un insieme di delimitatori ('\n', ' ', '\r', …);**
  2. **copia caratteri dallo stream di input, fino al primo di un insieme di delimitatori ('\n', ' ', '\r', …) escluso, in memoria a partire dall'indirizzo dst;**
  3. **memorizza '\0' nella locazione successiva all'ultima utilizzata per l'acquisizione.**

  Più informalmente:

  memorizza i caratteri acquisiti nella stringa dst.

Carlo Gaibisso
Bruno Martino

I/O di stringhe

- **Restituzione:**

  printf(“%s”, src);

  con src espressione di tipo char * e %s specificatore di formato per le stringhe

- **Effetto:**

  visualizza i caratteri memorizzati a partire dall’indirizzo src fino al primo ‘\0’ (escluso)

Più informalmente:

visualizza la stringa src

Carlo Gaibisso
Bruno Martino

Ingresso/uscita di stringhe

```
/* sorgente: StrIO.c */
/* illustra le modalità di acquisizione e restituzione delle stringhe */
/* inclusione del file di intestazione della libreria standard che
** contiene definizioni di macro, costanti e dichiarazioni di funzioni
** e tipi di interesse generale e funzionali alle varie operazioni di I/O */
#include <stdio.h>
#include <stdlib.h>
/* definizione della costante simbolica per la lunghezza massima di una
** stringa, '\0' escluso */
#define STR_LENGTH 80
/* chiamante */
int main ()
    {
    /* definizione delle due stringhe */
    char *Str1, *Str2;
    /* allocazione di memoria per le due stringhe. Se l'allocazione
    ** ha esito negativo, recupera la memoria eventualmente allocata e
    ** e termina */
    Str1 = (char *) malloc((STR_LENGTH+1)*sizeof(char));
    if (Str1 == NULL)
        return(0);
```

Continua ...

Carlo Gaibisso
Bruno Martino

Ingresso/uscita di stringhe

```
Str2 = (char *) malloc((STR_LENGTH+1)*sizeof(char));

if (Str2 == NULL)
    {
    free(Str1);
    return(0);
    };
/* acquisizione delle due stringhe */
printf("\nPrima Stringa? ");
scanf("%s", Str1);

printf("\nSeconda Stringa? ");
scanf("%s", Str2);

/* restituzione delle due stringhe */
printf("\nPrima stringa: %s", Str1);
printf("\nSeconda stringa: %s\n", Str2);

/* recupero della memoria allocata per le stringhe */
free(Str1);
free(Str2);

return(1);
}
```

Carlo Gaibisso
Bruno Martino

Stringhe costanti

- **Stringa costante:**

  sequenza di caratteri racchiusa tra doppi apici

- **Esempio:**
  - **“lkasdj jp656 #@!”**
  - **“Sono andato a casa”**
- **Esempio:**

```
/* sorgente: StrCost.c */
/* illustra un semplice esempio di uso delle stringhe costanti */
/* inclusione del file di intestazione della libreria standard che
** contiene definizioni di macro, costanti e dichiarazioni di funzioni
** e tipi funzionali alle varie operazioni di I/O */
#include <stdio.h>
/* chiamante */
int main ()
    {
    // visualizzazione di due diverse stringhe costanti
    printf("\n%s", "Vado a casa");
    printf("\n%s\n", "Vado a \0 casa");
    return(1);
    }
```

Carlo Gaibisso
Bruno Martino

Direttiva per il preprocessore

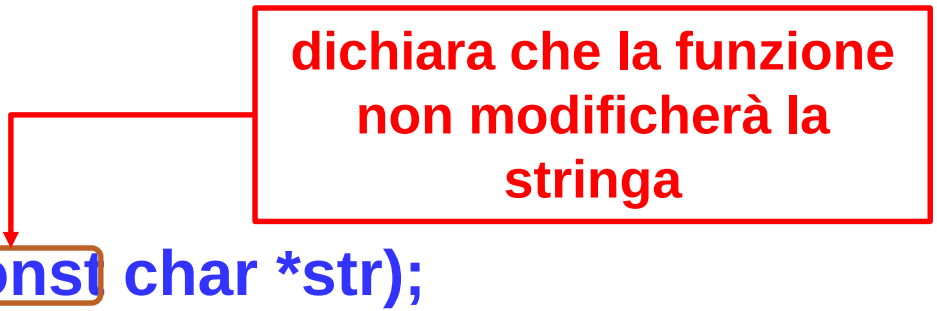
Attenzione!!!

Le librerie del C mettono a disposizione del programmatore un insieme di funzioni per il processamento delle stringhe. Per utilizzare tali funzioni all'interno di un file è necessario includere in testa allo stesso la direttiva per il preprocessore:

 # include <string.h>

Carlo Gaibisso
Bruno Martino

La funzione strlen()



- **Signature:**

  `size_t strlen (const char *str);`

- **Modifiche allo stato della memoria:**

  nessuno

- **Valore restituito:**

  il numero di caratteri memorizzati a partire dall'indirizzo `str` fino al primo `'\0'` escluso

Più informalmente:

la lunghezza della stringa `str`

Carlo Gaibisso
Bruno Martino

La funzione strlen()

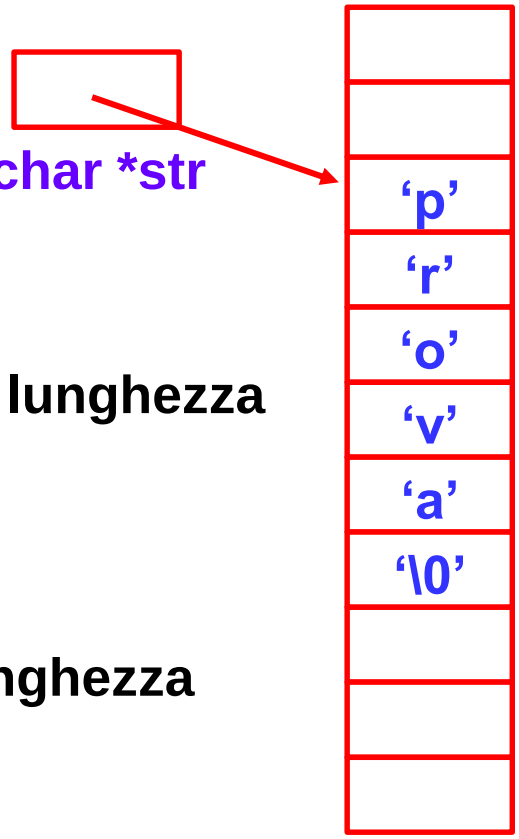
- **Esempio:**

```
int main ()
   {
   /* definisce la stringa */
   char str[...];
   /* definisce la variabile per la lunghezza
   ** della stringa */
   unsigned int lung;

   ….

   /* assegna alla variabile la lunghezza
   ** della stringa */
5  lung = strlen(stringa);

   ….
   }
```



Carlo Gaibisso
Bruno Martino

La funzione strlen()

```
/* sorgente: Strlen.c */
/* illustra le modalità di utilizzo della funzione strlen() */
/* inclusione del file di intestazione della libreria standard che
** contiene definizioni di macro, costanti e dichiarazioni di funzioni
**e tipi di interesse generale, funzionali alle varie operazioni di I/O
** e alla gestione delle stringhe, rispettivamente */
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
/* definizione della costante simbolica per la lunghezza massima di una
** stringa, '\0' escluso */
#define STR_LENGTH 80
/* chiamante */
int main ()
    {
    /* definizione della STRINGA */
    char Str[STR_LENGTH+1];
    /* acquisizione della stringa */
    printf("\nStringa? ");
    scanf("%s", Str);
    /* visualizzazione della lunghezza della stringa */
    printf("La lunghezza della stringa e': %u\n", strlen(Str));
    }
```

Carlo Gaibisso
Bruno Martino

La funzione strcpy()

- **Signature:**

  `char *strcpy(char *dst, const char *src);`

- **Modifiche allo stato della memoria:**

  copia la sequenza di caratteri di indirizzo iniziale **src**, fino al primo **'\0'** incluso, a partire dall'indirizzo **dst**.

  Più informalmente:

  copia la stringa **src** sulla stringa **dst**

- **Valore restituito:**

  l'indirizzo **dst**

Carlo Gaibisso
Bruno Martino

La funzione strcpy()

- **Esempio:**

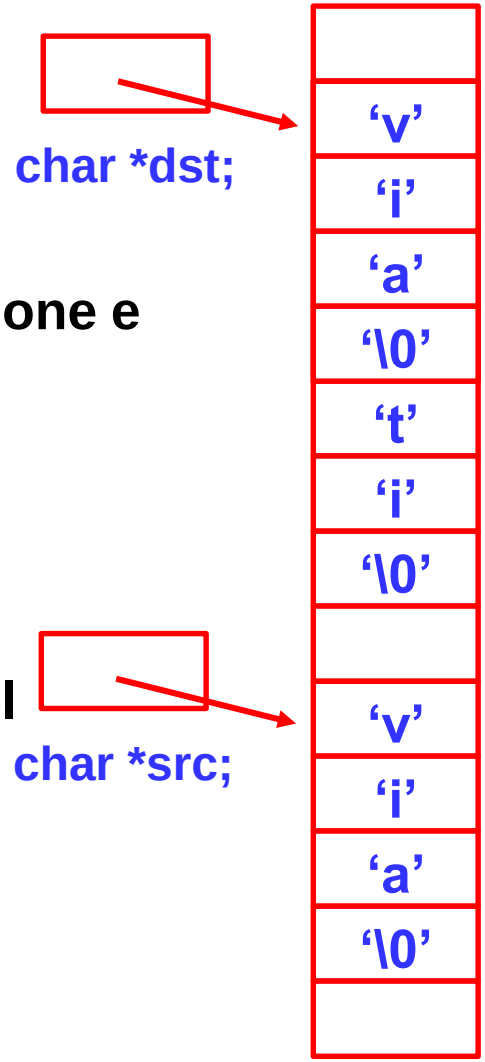
```
int main ()
   {
   /* definizione della stringa destinazione e
   ** di quella sorgente */
   char dst[...];
   char src[...];

   ….

   /* copia della stringa sorgente sulla
   ** destinazione e visualizzazione del
   ** risultato dell'operazione */
   printf("%s", strcpy(dst, src));

   …
   }
```

dst



Carlo Gaibisso
Bruno Martino

La funzione strcpy()

```
/* sorgente: Strcpy.c */
/* illustra le modalità di utilizzo della funzione strcpy() */
/* inclusione del file di intestazione della libreria standard che
** contiene definizioni di macro, costanti e dichiarazioni di funzioni
** e tipi di interesse generale, funzionali alle varie operazioni di I/O
** e alla gestione delle stringhe, rispettivamente */
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
/* definizione della costante simbolica per la lunghezza massima di una
** stringa,'\0' escluso */
#define STR_LENGTH 80
/* chiamante */
int main ()
    {
    /* definizione delle stringhe destinazione e sorgente */
    char Dst[STR_LENGTH+1];
    char Src[STR_LENGTH+1];
```

Continua ...

La funzione strcpy()

```
/* acquisizione delle due stringhe */
printf("\nDestinazione? ");
scanf("%s", Dst);
printf("\nSorgente? ");
scanf("%s", Src);
/* copia della stringa sorgente sulla destinazione e visualizzazione
** del risultato dell'operazione */
printf("\nDestinazione: %s\n", strcpy(Dst, Src));
return(1);
}
```

Carlo Gaibisso
Bruno Martino

La funzione strncpy()

- **Signature:**

```
char *strcnpy(char *dst,
        const char *src, size_t length);
```

- **Modifiche allo stato della memoria:**

  copia la sequenza di caratteri di indirizzo iniziale src a partire dall'indirizzo dst. Termina dopo length caratteri o al primo '\0' incontrato incluso.

  Più informalmente:

  copia length caratteri della stringa src sulla stringa dst

- **Valore restituito:**

  l'indirizzo dst

Carlo Gaibisso
Bruno Martino

La funzione strncpy()

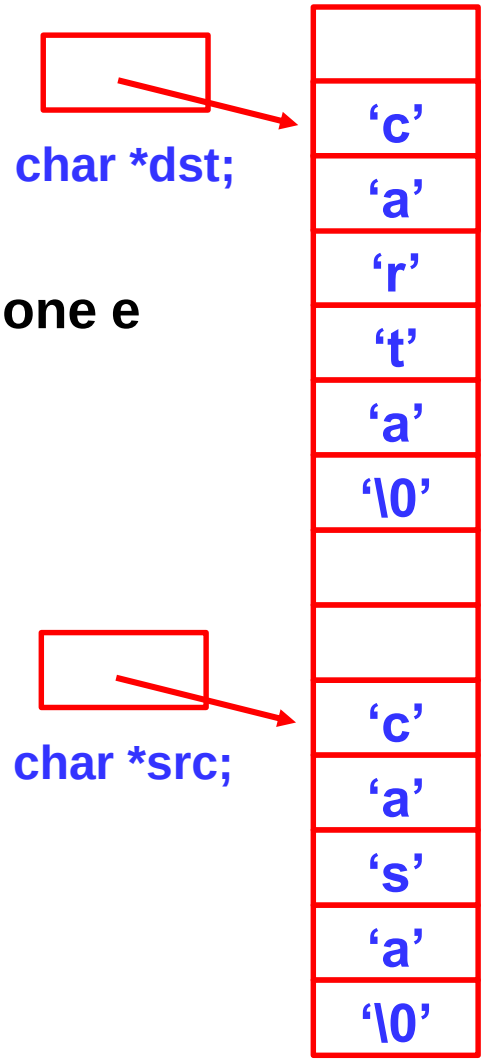
- **Esempio:**

```
int main ()
   {
   /* definizione della stringa destinazione e
   ** della stringa sorgente */
   char dst [...];
   char src [...];

   ....

   /* copia di 2 caratteri della stringa
   ** sorgente sulla destinazione e
   ** visualizzazione del risultato */
   printf("%s", strncpy(dst, src, 2));

   ...
   }
```



Carlo Gaibisso
Bruno Martino

La funzione strncpy()

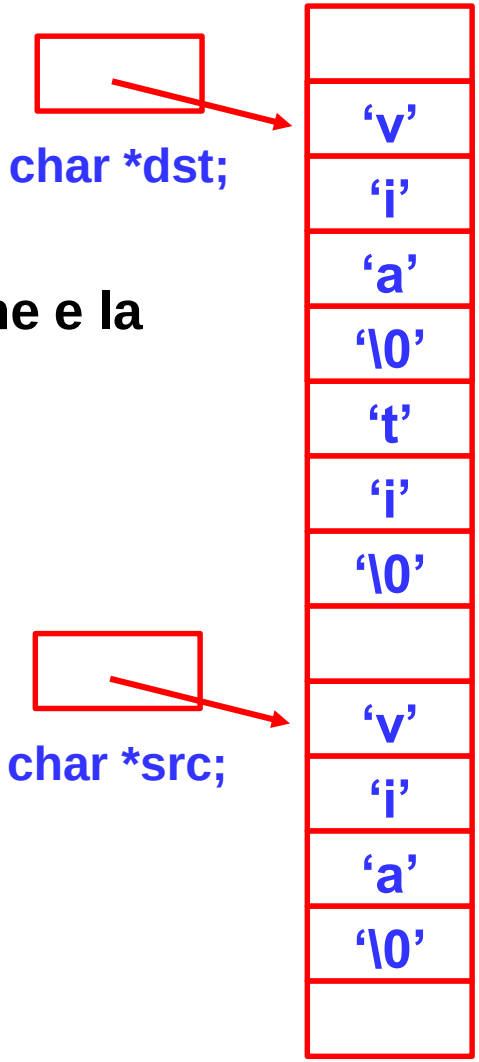
- **Esempio:**

```
int main ()
   {
   /* definizione della stringa destinazione e la
   ** stringa sorgente */
   char dst[...];
   char src = [...];

   ....

   /* copia di 4 caratteri della
   ** stringa sorgente sulla destinazione
   ** e visualizzazione del risultato */
   printf("%s", strncpy(dst, src, 8));

   …
   }
```



dst

Carlo Gaibisso
Bruno Martino

La funzione strncpy()

```
/* sorgente: Strncpy.c */
/* illustra le modalità di utilizzo della funzione strcnpy() */
/* inclusione del file di intestazione della libreria standard che
** contiene definizioni di macro, costanti e dichiarazioni di funzioni
** e tipi di interesse generale, funzionali alle varie operazioni di I/O
** e alla gestione delle stringhe, rispettivamente */
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
/* definizione della costante simbolica per la lunghezza massima di una stringa,
** '\0' escluso */
#define STR_LENGTH 80
/* chiamante */
int main ()
    {
    /* definizione della la stringa destinazione e della stringa sorgente */
    char Dst[STR_LENGTH+1];
    char Srg[STR_LENGTH+1];
    /* definizione di una variabile per il numero di caratteri da copiare */
    unsigned int nro;
```

Continua ...

Carlo Gaibisso
Bruno Martino

La funzione strncpy()

```
/* acquisizione delle due stringhe */
printf("\nDestinazione? ");
scanf("%s", Dst);
printf("\nSorgente? ");
scanf("%s", Srg);
/* acquisizione del numero di caratteri da copiare */
printf("\nNumero di caratteri? ");
scanf("%u", &nro);
/* copia di nro caratteri dalla stringa sorgente sulla destinazione e
** visualizzazione del risultato dell'operazione */
printf("\nDestinazione: %s\n", strncpy(Dst, Srg, nro));
return(1);
}
```

Carlo Gaibisso
Bruno Martino

La funzione strcat()

- **Signature:**

  `char *strcat(char *dst, const char *src);`

- **Modifiche allo stato della memoria:**

  copia la sequenza di caratteri di indirizzo iniziale **src**, fino al primo **'\0'** incluso, a partire dall'indirizzo del primo **'\0'** successivo a **dst**

  Più informalmente:

  concatena la stringa **src** alla stringa **dst**

- **Valore restituito:**

  l'indirizzo **dst**

Carlo Gaibisso
Bruno Martino

La funzione strcat()

- **Esempio:**

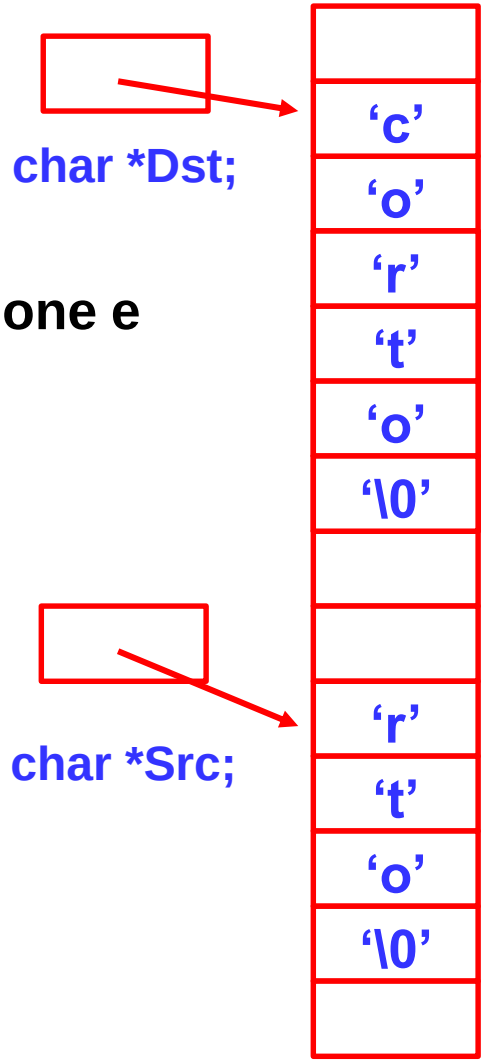
```
int main ()
   {
   /* definizione della stringa destinazione e
   ** della stringa sorgente */
   char Src[...];
   char Dst[...];

   ....

   /* concatenazione della stringa
   ** sorgente alla destinazione e
   ** visualizzazione del risultato */
   printf("%s", strcat(Src, Dst));

   ...
   }
```

Dst



Carlo Gaibisso
Bruno Martino

La funzione strcat()

```
/* sorgente: Strcat.c */
/* illustra le modalità di utilizzo della funzione strcat()
** concatenando due stringhe, separandole con uno spazio bianco */
/* inclusione del file di intestazione della libreria standard che
** contiene definizioni di macro, costanti e dichiarazioni di funzioni
** e tipi di interesse generale, funzionali alle varie operazioni di I/O
** e alla gestione delle stringhe, rispettivamente */
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
/* definizione della costante simbolica per la lunghezza massima di una
/**stringa, '\0' escluso */
#define STR_LENGTH 80
/* chiamante */
int main ()
    {
    /* definizione della stringa destinazione e della stringa sorgente */
    char Dst[STR_LENGTH+1];
    char Src[STR_LENGTH+1];
```

Continua ...

La funzione strcat()

```
/* concatenazione dello spazio bianco alla stringa destinazione */
strcat(Dst, " ");
/* concatenazione della stringa sorgente alla destinazione e visualizzazione
** del risultato dell'operazione */
printf("\nDestinazione: %s\n", strcat(Dst, Src));
return(1);
}
```

Carlo Gaibisso
Bruno Martino

La funzione strncat()

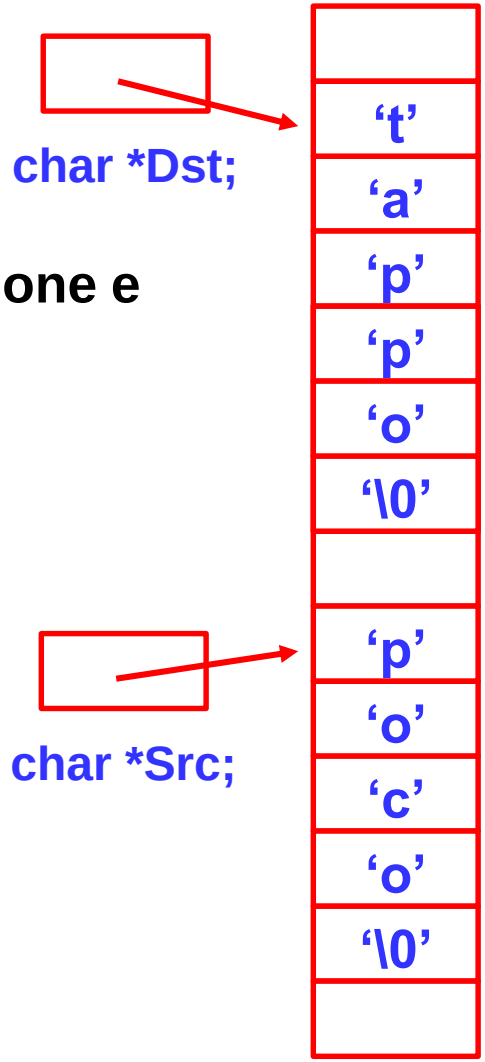
- **Esempio:**

```
int main ()
        {
        /* definizione della stringa destinazione e
        ** della stringa sorgente */
        char Dst[...];
        char Src[...];

        ….

        /* concatenazione dei primi 2
        ** caratteri della sorgente alla
        ** destinazione e visualizzazione
        ** del risultato */
        printf(“%s”, strcat(Src, Dst, 2));

        …
        }
```



Carlo Gaibisso
Bruno Martino

La funzione strncat()

```
/* sorgente: Strncat.c */
/* illustra le modalità di utilizzo della funzione strncat() */
/* inclusione del file di intestazione della libreria standard che
** contiene definizioni di macro, costanti e dichiarazioni di funzioni
** e tipi di interesse generale, funzionali alle varie operazioni di I/O
** e alla gestione delle stringhe, rispettivamente */
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
/* definizione della costante simbolica per la lunghezza massima di una
** stringa, '\0' escluso */
#define STR_LENGTH 80
/* chiamante */
int main ()
    {
    /* definizione della stringa destinazione e della stringa sorgente */
    char Dst[STR_LENGTH+1];
    char Src[STR_LENGTH+1];
    /* definizione della variabile per il numero di caratteri da concatenare */
    unsigned int nro;
    /* acquisizione delle due stringhe */
    printf("\nDestinazione? ");
    scanf("%s", Dst);
    printf("\nSorgente? ");
    scanf("%s", Src);
```

Continua ...

Carlo Gaibisso
Bruno Martino

La funzione strncat()

```
/* acquisizione del numero di caratteri da concatenare */
printf("\nNumero di caratteri? ");
scanf("%u", &nro);
/* concatenazione di nro caratteri della stringa sorgente alla destinazione e
** visualizzazione del risultato dell'operazione */
printf("\nDestinazione: %s\n", strncat(Dst, Src, nro));
return(1);
}
```

Carlo Gaibisso
Bruno Martino

La funzione strcmp()

- **Signature:**

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

- **Modifiche allo stato della memoria:**

  Alcuna

- **Valore restituito:**

  confronta carattere per carattere le due sequenze di caratteri di indirizzo iniziale str1 e str2.

  Termina al raggiungimento del primo '\0' o della prima coppia di caratteri differenti.

  Nel primo caso, siano c1 e c2 l'ultima coppia di caratteri raggiunti (almeno uno tra di loro è '\0') a partire da str1 e da str2, rispettivamente.

Carlo Gaibisso
Bruno Martino

La funzione strcmp()

Restituisce:

- **0 se c1 = c2 = '\0';**
- **un valore negativo se c1 = '\0' e c2 ≠ '\0'**
- **un valore positivo se c1 ≠ '\0' e c2 = '\0'**

Nel secondo caso siano c1 e c2 l'ultima coppia di caratteri confrontati (c1, c2 ≠ '\0'; c1≠ c2), in str1 e str2, rispettivamente.

Restituisce:

- **un valore negativo se c1 < c2 (codici ASCII)**
- **un valore positivo se c1 > c2**

Carlo Gaibisso
Bruno Martino

La funzione strcmp()

Più informalmente:

restituisce un intero il cui segno identifica il risultato del confronto lessicografico tra str1 e str2:

- **0 se str1 = str2**
- **negativo se str1 precede lessicograficamente str2**
- **positivo se str2 precede lessicograficamente str1**

Carlo Gaibisso
Bruno Martino

La funzione strcmp()

```
/* sorgente: Strcmp.c */
/* illustra le modalità di utilizzo della funzione strcmp() */
/* inclusione del file di intestazione della libreria standard che
** contiene definizioni di macro, costanti e dichiarazioni di funzioni
** e tipi di interesse generale, funzionali alle varie operazioni di I/O
** e alla gestione delle stringhe, rispettivamente */
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
/* definizione della costante simbolica per la lunghezza massima di una
** stringa, '\0' escluso */
#define STR_LENGTH 80
/* chiamante */
int main ()
    {
    /* definizione delle due stringhe oggetto del confronto */
    char Str1[STR_LENGTH+1];
    char Str2[STR_LENGTH+1];
    /* acquisizione delle due stringhe */
    printf("\nI Stringa? ");
    scanf("%s", Str1);
    printf("\nII Stringa? ");
    scanf("%s", Str2);
```

Continua ...

Carlo Gaibisso
Bruno Martino

La funzione strcmp()

```
/* confronto lessicografico delle due stringhe e visualizzazione
** del risultato del confronto */
if (strcmp(Str1, Str2) > 0)
    printf("\n%s > %s\n", Str1, Str2);
else
    if (strcmp(Str1, Str2) < 0)
        printf("\n%s < %s\n", Str1, Str2);
    else
        printf("\n%s = %s\n", Str1, Str2);
return(1);
}
```

Raccomandazione

Attenzione!!!

Consultare la manualistica per avere un quadro esaustivo delle funzioni per la manipolazione delle stringhe messe a disposizione dalle librerie del C