

```
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
```

```
#define MAXPAROLA 30
#define MAXRIGA 80
```

```
int main(int argc, char *argv[])
{
    int freq[MAXPAROLA]; /* vettore di contatori
delle frequenze delle lunghezze delle parole */
    char riga[MAXRIGA];
    int i, inizio, lunghezza;
    FILE *f;
```

```
for(i=0; i<MAXPAROLA; i++)
    freq[i]=0;
```

```
if(argc != 2)
{
    fprintf(stderr, "ERRORE, serve un parametro con il nome del file\n");
    exit(1);
}
```

```
f = fopen(argv[1], "r");
if(f==NULL)
{
    fprintf(stderr, "ERRORE, impossibile aprire il file %s\n", argv[1]);
    exit(1);
}
```

```
while( fgets( riga, MAXRIGA, f ) != NULL )
```



## Caratteri e stringhe

# Operazioni elementari sulle stringhe

# Operazioni elementari sulle stringhe

- Lunghezza
- Copia di stringhe
- Concatenazione di stringhe
- Confronto di stringhe
- Ricerca di sotto-stringhe
- Ricerca di parole

```
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
```

```
#define MAXPAROLA 30
#define MAXRIGA 80
```

```
int main(int argc, char *argv[])
{
    int freq[MAXPAROLA]; /* vettore di contatori
delle frequenze delle lunghezze delle parole */
    char riga[MAXRIGA];
    int i, inizio, lunghezza;
    FILE *f;
```

```
    for(i=0; i<MAXPAROLA; i++)
        freq[i]=0;
```

```
    if(argc != 2)
    {
        fprintf(stderr, "ERRORE, serve un parametro con il nome del file\n");
        exit(1);
    }
```

```
    f = fopen(argv[1], "r");
    if(f==NULL)
    {
        fprintf(stderr, "ERRORE, impossibile aprire il file %s\n", argv[1]);
        exit(1);
    }
```

```
    while( fgets( riga, MAXRIGA, f ) != NULL )
```



# Operazioni elementari sulle stringhe

## Lunghezza

## Lunghezza di una stringa

- La lunghezza di una stringa si può determinare ricercando la posizione del terminatore nullo

```
char s[MAX+1] ;  
int lun ;
```

s	s	a	l	v	e	Ø	3	r	w	t
	0	1	2	3	4	5				

## Calcolo della lunghezza

```
const int MAX = 20 ;  
char s[MAX+1] ;  
int lun ;  
int i ;  
  
... /* lettura stringa */  
  
for( i=0 ; s[i] != 0 ; i++ )  
    /* Niente */ ;  
  
lun = i ;
```

## La funzione strlen

- Nella libreria standard C è disponibile la funzione `strlen`, che calcola la lunghezza della stringa passata come parametro
- Necessario includere `<string.h>`

```
const int MAX = 20 ;  
char s[MAX+1] ;  
int lun ;  
  
... /* lettura stringa */  
  
lun = strlen(s) ;
```



```
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
```

```
#define MAXPAROLA 30
#define MAXRIGA 80
```

```
int main(int argc, char *argv[])
{
    int freq[MAXPAROLA]; /* vettore di contatori
delle frequenze delle lunghezze delle parole */
    char riga[MAXRIGA];
    int i, inizio, lunghezza;
    FILE *f;
```

```
    for(i=0; i<MAXPAROLA; i++)
        freq[i]=0;
```

```
    if(argc != 2)
    {
        fprintf(stderr, "ERRORE, serve un parametro con il nome del file\n");
        exit(1);
    }
```

```
    f = fopen(argv[1], "r");
    if(f==NULL)
    {
        fprintf(stderr, "ERRORE, impossibile aprire il file %s\n", argv[1]);
        exit(1);
    }
```

```
    while( fgets( riga, MAXRIGA, f ) != NULL )
```



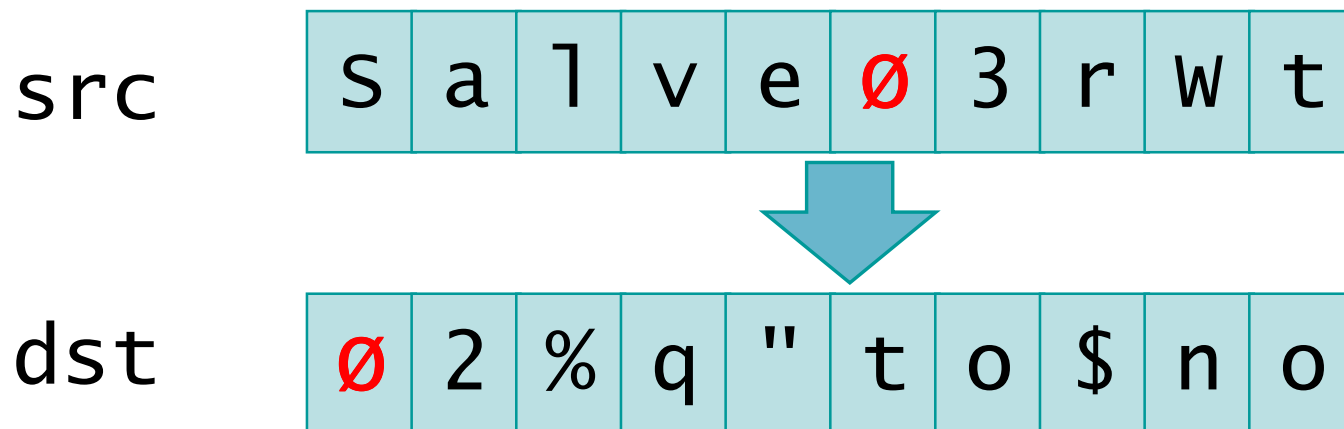
# Operazioni elementari sulle stringhe

## Copia di stringhe

## Copia di stringhe

- L'operazione di copia prevede di ricopiare il contenuto di una prima stringa "sorgente", in una seconda stringa "destinazione"

```
char src[MAXS+1] ;  
char dst[MAXD+1] ;
```





## Risultato della copia

src

S	a	l	v	e	Ø	3	r	w	t
---	---	---	---	---	---	---	---	---	---

dst

Ø	2	%	q	"	t	o	\$	n	o
---	---	---	---	---	---	---	----	---	---

Copia src in dst



dst

S	a	l	v	e	Ø	o	\$	n	o
---	---	---	---	---	---	---	----	---	---

```
const int MAXS = 20, MAXD = 30 ;  
char src[MAXS+1] ;  
char dst[MAXD+1] ;  
int i ;  
  
... /* lettura stringa src */  
  
for( i=0 ; src[i] != 0 ; i++ )  
    dst[i] = src[i] ; /* copia */  
  
dst[i] = 0 ; /* aggiunge terminatore */
```

## La funzione strcpy

- Nella libreria standard C, includendo `<string.h>`, è disponibile la funzione `strcpy`, che effettua la copia di stringhe
  - Primo parametro: stringa destinazione
  - Secondo parametro: stringa sorgente

```
const int MAXS = 20, MAXD = 30 ;  
char src[MAXS+1] ;  
char dst[MAXD+1] ;  
  
... /* lettura stringa src */  
  
strcpy(dst, src) ;
```

## Avvertenze

- Nella stringa destinazione vi deve essere un numero sufficiente di locazioni libere
  - `MAXD+1 >= strlen(src)+1`
- Il contenuto precedente della stringa destinazione viene perso
- La stringa sorgente non viene modificata
- Il terminatore nullo
  - Deve essere aggiunto in coda a `dst`
  - La `strcpy` pensa già autonomamente a farlo



## Errore frequente

- Per effettuare una copia di stringhe **non** si può assolutamente utilizzare l'operatore =
- Necessario usare strcpy

~~dst = src ;~~

~~dst[] = src[] ;~~

~~dst[MAXD] = src[MAXC] ;~~

strcpy(dst, src);

```
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
```

```
#define MAXPAROLA 30
#define MAXRIGA 80
```

```
int main(int argc, char *argv[])
{
    int freq[MAXPAROLA]; /* vettore di contatori
delle frequenze delle lunghezze delle parole */
    char riga[MAXRIGA];
    int i, inizio, lunghezza;
    FILE *f;
```

```
    for(i=0; i<MAXPAROLA; i++)
        freq[i]=0;
```

```
    if(argc != 2)
    {
        fprintf(stderr, "ERRORE, serve un parametro con il nome del file\n");
        exit(1);
    }
```

```
    f = fopen(argv[1], "r");
    if(f==NULL)
    {
        fprintf(stderr, "ERRORE, impossibile aprire il file %s\n", argv[1]);
        exit(1);
    }
```

```
    while( fgets( riga, MAXRIGA, f ) != NULL )
```



# Operazioni elementari sulle stringhe

## Concatenazione di stringhe



# Concatenazione di stringhe

- L'operazione di concatenazione corrisponde a creare una nuova stringa composta dai caratteri di una prima stringa, **seguiti** dai caratteri di una seconda stringa

sa

S	a	l	v	e	Ø	3	r	w	t
---	---	---	---	---	---	---	---	---	---

sb

m	o	n	d	o	Ø	o	\$	n	o
---	---	---	---	---	---	---	----	---	---

Concatenazione di sa con sb



S	a	l	v	e	m	o	n	d	o	Ø	w	1	Q	r
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

## Semplificazione

- Per maggior semplicità, in C l'operazione di concatenazione scrive il risultato **nello stesso vettore** della prima stringa
- Il valore precedente della prima stringa viene così perso
- Per memorizzare altrove il risultato, o per non perdere la prima stringa, è possibile ricorrere a stringhe temporanee ed alla funzione `strcpy`

## Esempio

sa S a l v e ~~Ø~~ w z 3 w 7 w 1 Q r

sb m o n d o ~~Ø~~ h ! L . 2 x y E P

Concatenazione di sa con sb




sa S a l v e m o n d o ~~Ø~~ w 1 Q r

sb m o n d o ~~Ø~~ h ! L . 2 x y E P

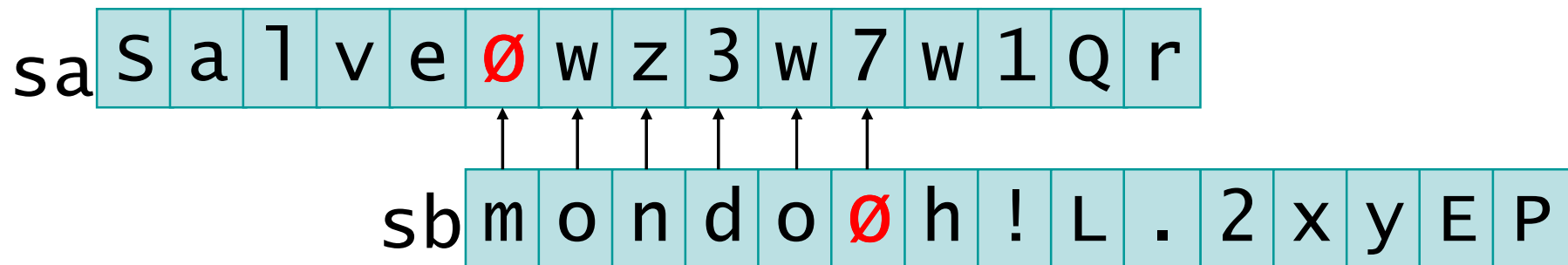
# Algoritmo di concatenazione

- Trova la fine della prima stringa

sa S a l v e  w z 3 w 7 w 1 Q r

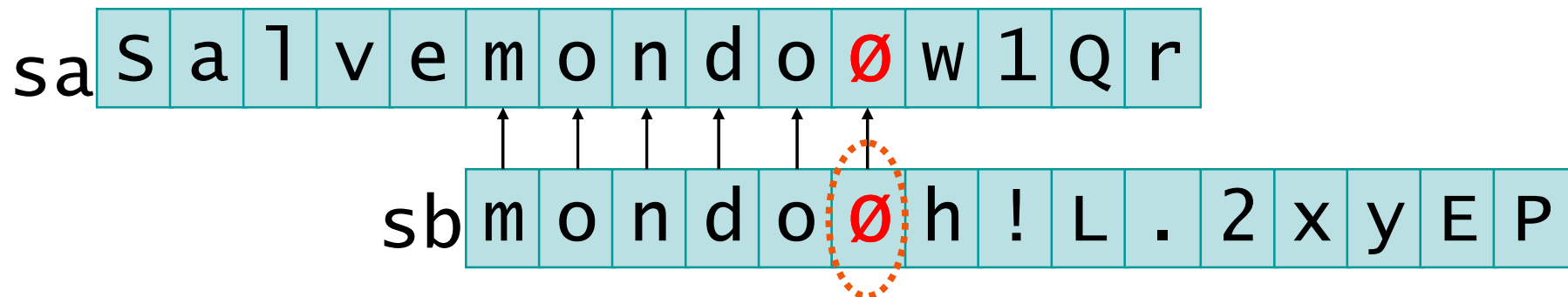
# Algoritmo di concatenazione

- Trova la fine della prima stringa
- Copia la seconda stringa nel vettore della prima, a partire dalla posizione del terminatore nullo (sovrascrivendolo)



# Algoritmo di concatenazione

- Trova la fine della prima stringa
- Copia la seconda stringa nel vettore della prima, a partire dalla posizione del terminatore nullo (sovrascrivendolo)
- Termina la copia non appena trovato il terminatore della seconda stringa





## Concatenazione

```
const int MAX = 20 ;  
char sa[MAX] ;  
char sb[MAX] ;  
int la ;  
int i ;  
  
... /* lettura stringhe */  
  
la = strlen(sa) ;  
  
for( i=0 ; sb[i] != 0 ; i++ )  
    sa[la+i] = sb[i] ; /* copia */  
  
sa[la+i] = 0 ; /* terminatore */
```

## La funzione strcat

- Nella libreria standard C, includendo `<string.h>`, è disponibile la funzione `strcat`, che effettua la concatenazione di stringhe
  - Primo parametro: prima stringa (destinazione)
  - Secondo parametro: seconda stringa

```
const int MAX = 20 ;  
char sa[MAX] ;  
char sb[MAX] ;  
  
... /* lettura stringhe */  
  
strcat(sa, sb) ;
```

## Avvertenze (1/2)

- Nella prima stringa vi deve essere un numero sufficiente di locazioni libere
  - $MAX+1 \geq \text{strlen}(sa) + \text{strlen}(sb) + 1$
- Il contenuto precedente della prima stringa viene perso
- La seconda stringa non viene modificata
- Il terminatore nullo
  - Deve essere aggiunto in coda alla prima stringa
  - La `strcat` pensa già autonomamente a farlo

## Avvertenze (2/2)

- Per concatenare 3 o più stringhe, occorre farlo due a due:
  - `strcat(sa, sb);`
  - `strcat(sa, sc);`
- È possibile concatenare anche stringhe costanti
  - `strcat(sa, "!");`