



UNIVERSITÀ DI PARMA

# Le Stringhe

*If a string has one end, it has another.*

Gautama Buddha

- Definizione stringhe
- Memorizzazione
- I/O
- La libreria `string.h`
- Principali operazioni
- Array di stringhe

SUMMARY



- In informatica, si definisce *stringa* una sequenza *ordinata* di simboli
- Per il C
  - Solo codici ASCII → sequenza di dati di tipo char
  - Terminata da carattere nullo ‘\0’
- Abbiamo già visto stringhe costanti
  - Tutto ciò che è racchiuso tra “ e ”
  - “” → stringa “vuota”

- Come le memorizzo?
- Sono sequenze di char → array di char
- Non vale il contrario
  - Un array di char non è automaticamente una stringa
  - Ci vuole il carattere nullo di terminazione
- L'array è solo un “contenitore” la stringa è il contenuto
  - `char mystring[100] = “Hello World!”;`
- **Attenzione!** Ci vuole spazio per il `'\0'`

- `printf()` e `scanf()`
  - Specificatore di formato `%s`
    - Per `scanf()` si ferma al primo spazio != **`printf()`**
    - Legge solo “parole”
    - Soluzione → `%[...]` e nello specifico `%[^\\n]`
  - destinazione/sorgente ↔ array di char
- Nel caso della lettura lo `'\\0'` viene aggiunto automaticamente

- Il C fornisce numerose funzioni per la manipolazione delle stringhe:
- `#include<string.h>`
- Prendono in ingresso/uscita array di char **MA**
  - Non funzionano se manca il `'\0'`
  - Ovvero gli array devono contenere stringhe

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

```
int      strcasecmp(const char *s1, const char *s2);
char     *strcat(char *dest, const char *src);
char     *strchr(const char *s, int c);
int      strcmp(const char *s1, const char *s2);
int      strcoll(const char *s1, const char *s2);
char     *strcpy(char *dest, const char *src);
size_t   strcspn(const char *s, const char *reject);
char     *strdup(const char *s);
char     *strfry(char *string);
size_t   strlen(const char *s);
char     *strncat(char *dest, const char *src, size_t n);
int      strncmp(const char *s1, const char *s2, size_t n);
char     *strncpy(char *dest, const char *src, size_t n);
int      strncasecmp(const char *s1, const char *s2, size_t n);
char     *strpbrk(const char *s, const char *accept);
char     *strrchr(const char *s, int c);
char     *strsep(char **stringp, const char *delim);
size_t   strspn(const char *s, const char *accept);
char     *strstr(const char *haystack, const char *needle);
char     *strtok(char *s, const char *delim);
size_t   strxfrm(char *dest, const char *src, size_t n);
char     *index(const char *s, int c);
char     *rindex(const char *s, int c);
```

- Calcolo lunghezza
- Confronto
- Copia/assegnamento
- Accodamento
- Ricerca caratteri e sottostringhe



```
size_t strlen(const char *s);
```

- `strlen(s)`
- Restituisce numero caratteri stringa `s`
- Non conta carattere nullo di terminazione

```
int strcmp(const char *s1, const char *s2);
```

- `strcmp(s1, s2)`
- Confronta le stringhe s1 e s2
- Restituisce
  - 0 → stringhe uguali carattere per carattere
  - <0 → s1 precede s2 in “ordine ASCII”
  - >0 → s1 segue s2 in “ordine ASCII”

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

- `strcpy(dest, src)`
- Copia la stringa `src` nell'array `dest`
- Non viene fatto controllo sullo spazio della destinazione
- `src` può essere stringa costante

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

- `strcat(dest, src)`
- Accoda la stringa `src` alla stringa `dest`
  - Nessun controllo sullo spazio disponibile
  - `dest` può essere anche la stringa vuota ma deve essere una stringa!

```
char *strchr(const char *s, int c);  
char *strrchr(const char *s, int c);
```

- `strchr(s, c)` o `strrchr(s, c)`
- Ricerca la prima occorrenza del carattere `c` nella stringa `s`
  - Da sinistra o destra
  - NULL se non lo trova
- `strstr(h, n)`
- Ricerca la prima occorrenza della stringa `n` nella stringa `h`
  - Da sinistra
  - NULL se non esiste

# char \*strtok(char \*str, const char \*delim)

- strtok() permette di dividere una stringa in “token”
  - Sottostringhe opportunamente suddivise da delimitatori
- Prende in ingresso la prima volta:
  - Stringa da “spezzettare”
  - Stringa contenente caratteri di delimitazione
- Le volte successive
  - NULL
  - Stringa contenente caratteri di delimitazione
- Restituisce
  - Indirizzo prossimo token individuato oppure NULL se non ce ne sono più

- La stringhe già sono memorizzate in array
  - Posso usare array di array
- Array bidimensionali
  - Devo dimensionare ogni riga con la lunghezza massima prevista
  - Spreco spazio
- Array di puntatori + allocazione dinamica
  - Più complesso ma ottimizzo uso spazio



UNIVERSITÀ DI PARMA

# Le Stringhe



KEEP  
CALM  
IT'S  
QUESTION  
TIME

*If a string has one end, it has another.*

Gautama Buddha