

Stringhe di caratteri

- In C non è predefinito un tipo per rappresentare le stringhe, cioè le sequenze di caratteri

Una stringa di caratteri è rappresentata tramite un vettore di caratteri, terminato dal carattere speciale '`\0`', con codice ASCII pari a zero.

- Per memorizzare stringhe di caratteri di lunghezza massima N:

```
char S[N+1]; /* variabile di tipo stringa */
```

una posizione del vettore riporta il carattere terminatore '`\0`'.

- la stringa memorizzata in S può avere una lunghezza L minore di N: in tal caso il carattere terminatore '`\0`' sarà l'elemento L+1, cioè l'elemento in posizione L; le posizioni del vettore successive ad L conterranno un valore indefinito.

Stringa S vuota (lunghezza 0) : `S[0]=' \0'`.

Una *costante stringa* è una sequenza di caratteri racchiusa tra "...":

e	'		u	n	a		p	r	o	v	a	\0
---	---	--	---	---	---	--	---	---	---	---	---	----

- Inizializzazione di una variabile stringa

```
char S[6] = {'p', 'r', 'o', 'v', 'a', '\0'};
```

oppure utilizzare una costante stringa:

```
char S[6] = "prova";
```

```
char S[] = "prova"; /* 6 carat. l'ultimo e` \0 */
```

- Tutti gli algoritmi sulle stringhe presentati sono basati sull'ipotesi che la stringa in input sia terminata correttamente con '`\0`'.

Calcolo della lunghezza di una stringa:

```
char S[] = "prova";  
int lunghezza=0;  
while(S[lunghezza]!='\0')  
    lunghezza++;  
printf("La stringa %s e' lunga %d", S,lunghezza);
```

Copia di una stringa s1 in un'altra stringa s2:

```
char S1[] = "prova"; /* stringa origine */  
char S2[10]; /* stringa destinazione */  
int i;  
  
i=0;  
while(S1[i]!='\0') {  
    S2[i]=S1[i];  
    i++;  
}  
S2[i]='\0';
```

Lettura di una stringa da input

```
char S1[5],S2[5]; /* due stringhe di 4 caratteri */
```

Lettura tramite la funzione scanf

La variabile stringa deve essere **indicata senza l'operatore indirizzo &**, in quanto, il nome di una stringa (ed in generale di un array) denota già un indirizzo (l'indirizzo del primo elemento)

```
scanf("%s",S1);  
scanf("%s",S2);
```

oppure

```
scanf("%s%s",S1,S2);
```

Con %s la stringa immessa da input deve terminare con uno spazio bianco o con ↵.

uno due↵ → S1="uno", S2="due"

uno ↵ due↵ → S1="uno", S2="due"

Funziona correttamente solo se le stringhe immesse da input hanno una lunghezza minore o uguale a quella dichiarata; in tal caso le stringhe sono terminate con '\0'.

Si hanno dei **problemi** se da input vengono introdotte **stringhe più lunghe**: le stringhe non vengono acquisite correttamente, non vengono terminate con '\0', ...

Indichiamo allora la dimensione massima della stringa nel carattere di formattazione

```
scanf("%4s%4s",S1,S2);
```

Se introduco una stringa più lunga di 4, alla stringa S1 vengono assegnati i primi 4 caratteri (e in più viene messo il carattere '\0') ma i rimanenti caratteri vengono messi in S2!

undici ↵ dodici ↵ → S1="undi", S2="ci"

Quindi possiamo leggere le stringhe con scanf "sperando" che chi usa il programma rispetti il vincolo sui numeri dei caratteri ...

Altrimenti devo realizzare un algoritmo per leggere la stringa **carattere per carattere**.

Esempio: leggere una stringa di lunghezza MAXSTRINGA (#define MAXSTRINGA 4); la lettura deve considerare anche gli spazi bianchi e deve terminare quando si immette ↵ (carattere '\n'); inoltre, eventuale caratteri in più devono essere scartati!

```
scanf("%c",&Ch);  
I=0;  
while(Ch!='\n'){  
    if (I<MAXSTRINGA){  
        S[I]=Ch;  
        I++;  
    }  
    scanf("%c",&Ch);  
}  
S[I]='\0'; /* si deve assegnare esplicitamente il '\0' */
```

Il valore di I al termine della lettura rappresenta la **lunghezza della stringa**.

Esempio : campi di record definiti come string

```
#include <stdio.h>
#define MAXSTRINGA 20 /* massima lunghezza della stringa */

main() {
    typedef char Parola[MAXSTRINGA+1];
    typedef struct { Parola Nome, Cognome;
                    int Giorno, Mese, Anno;
                    } Persona;

    Persona P1,P2;

    printf("Inserire Cognome e Nome della Persona P1 \n");
    scanf("%s%s",&P1.Cognome,&P1.Nome);

    printf("Inserire Data Nascita della Persona P1 \n");
    scanf("%d%d%d",&P1.Giorno,&P1.Mese,&P1.Anno);

    printf("Inserire Cognome , Nome e Data di Nascita di P2 \n");
    scanf("%s%s",&P2.Cognome,&P2.Nome);
    scanf("%d%d%d",&P2.Giorno,&P2.Mese,&P2.Anno);

    printf("%s   %s   è   nato   il   %d/%d/%d\n",P1.Nome,P1.Cognome,
           P1.Giorno,P1.Mese,P1.Anno);

    printf("%s   %s   è   nato   il   %d/%d/%d\n",P2.Nome,P2.Cognome,
           P2.Giorno,P2.Mese,P2.Anno);
}
```

Definiamo anche Data come record :

```
main() {
    typedef char Parola[MAXSTRINGA+1];
    typedef struct {
        int Giorno, Mese, Anno;
    } TipoData;

    typedef struct { Parola Nome, Cognome;
                    TipoData Data;
    } Persona;

    Persona P1,P2;

    printf("Inserire Cognome e Nome della Persona P1 \n");
    scanf("%s%s",&P1.Cognome,&P1.Nome);

    printf("Inserire Data Nascita della Persona P1 \n");
    scanf("%d%d%d",&P1.Data.Giorno,&P1.Data.Mese,&P1.Data.Anno);

    printf("%s   %s   è   nato   il   %d/%d/%d\n",P1.Nome,P1.Cognome,
           P1.Data.Giorno,P1.Data.Mese,P1.Data.Anno);
}
```