

Stringhe

Antonella Santone

1

Definizione

Una variabile di tipo `char` consente di memorizzare un singolo carattere

Trattare come una sola unità un insieme di caratteri alfanumerici: **STRINGA**

→ Array di `char`

Esempio

```
char a[10];
```

2

Definizione (cont.)

Una stringa di caratteri in *C* è un array di caratteri terminato dal carattere nullo `'\0'`

Un vettore di *N* caratteri può dunque ospitare stringhe lunghe al più *N*-1 caratteri, perché una cella è destinata al terminatore `'\0'`

a	p	e	'\0'
0	1	2	3

`char a[11];` può contenere una stringa di 10 caratteri

3

Inizializzazione

Una stringa si può inizializzare, come ogni altro array, elencando le singole componenti:

```
char s[4] = {'a', 'p', 'e', '\0'};
```

oppure anche, più brevemente, con la forma compatta seguente:

```
char s[4] = "ape" ;
```

Il carattere di terminazione `'\0'` è automaticamente incluso in fondo. Attenzione alla lunghezza!

4

Inizializzazione (cont.)

```
char frase[] = "Analisi, requisiti ";
```

Dichiara un vettore di caratteri il cui numero di elementi è determinato dalla quantità di caratteri presenti tra doppi apici + 1 (carattere `'\0'`)

A	n	a	l	i	s	i	,		r	e	q	u	i	s	i	t	i		\0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

```
char s[21] = "ciao";
```

Array parzialmente riempito. In questo caso, le celle oltre la *k*-esima (*k* essendo la lunghezza della stringa) sono concettualmente vuote: praticamente sono inutilizzate e contengono un valore casuale

5

Attenzione!

- 1) `char d = 'r';`
diverso da
- 2) `char b[] = "r";`

La prima assegna alla variabile *d* di tipo `char` il valore *r*

La seconda assegna all'array *b*[] la sequenza *r* e `\0`

```
b[0] = 't';
```

E' lecito e vuol dire assegnare al primo elemento dell'array *b* il carattere *t*



6

Attenzione!

```
char x = 'a';           // OK
char y = "b";          // NO !!!
char z[] = "c";        // OK
char w[] = 'd';        // NO !!!
```

7

Stringa: lettura

Una stringa si può leggere da tastiera e stampare, come ogni altro array, elencando le singole componenti:

```
...
char str[4]; int i;
for (i=0; i < 3; i++)
    scanf("%c", &str[i]);
str[3] = '\0';
...
```

8

Stringa: lettura (cont.)

oppure, più brevemente, con la forma compatta:

```
...
char str[4];
scanf("%s", str);
```

*str è un vettore (puntatore)
→ l'operatore & non va messo*

`scanf` leggerà caratteri finché non avrà incontrato uno spazio o un **newline**

NOTA

E' un'eccezione !!!

Gli array non si possono leggere e scrivere interamente, ma elemento per elemento

Attenzione: Se si immette un numero maggiore di caratteri si rischia di sovrascrivere delle zone di memoria con altri dati

9

Stringa: lettura (cont.)

```
#include <stdio.h>
```

```
main() {
    char s[40];
    scanf("%s", s);
    printf("%s", s);
}
```

`gets(s)`:

legge i caratteri dallo standard input e li immagazina nel vettore `s`, finché non incontra un carattere `newline`. Alla fine del vettore sarà accodato un carattere nullo di terminazione

Funzione inclusa nella libreria per l'input/output standard

Digito:
elementi di informatica
Stampa:
elementi

10

Stringa: lettura (cont.)

Letture con `gets`

```
#include <stdio.h>
main() {
    char str[40];
    gets(str);
    printf(str);
}
```

Digito:
elementi di informatica
Stampa:
elementi di informatica

11

Stringa: scrittura

```
#include <stdio.h>
```

```
main () {
    char frase[] = "Questa e' una stringa";
    int i = 0;
    while (frase[i] != '\0') {
        printf("%c", frase[i]);
        i++;
    }
}
```

12

Stringa: scrittura (cont.)

Oppure equivalentemente:

```
#include <stdio.h>

main () {
    char frase[] = "Questa e' una stringa";
    printf("%s", frase);
}
```

In questo caso è l'istruzione `printf` stessa che provvede a stampare carattere per carattere la stringa e a bloccarsi nel momento in cui si identifica il carattere `\0`

```
puts(s)
visualizza la stringa seguita da un carattere di newline
Funzione inclusa nella libreria per l'input/output standard
```

13

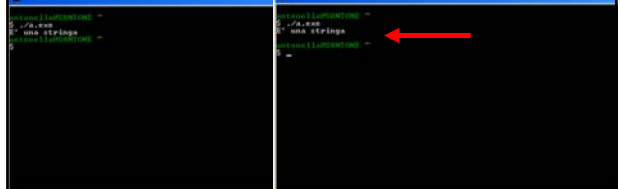
Differenza

```
#include <stdio.h>
```

```
main () {
    char frase[] = "E' una stringa";
    printf("%s", frase);
}
```

```
#include <stdio.h>
```

```
main () {
    char frase[] = "E' una stringa";
    puts(frase);
}
```



Carattere \0

Il carattere terminatore `\0` ci consente di trattare le stringhe senza conoscere a priori la dimensione

Esempio

```
#include <stdio.h>
main() {
    char frase[] = "Analisi, requisiti ";
    int i=0;
    while (frase[i]!='\0') {
        printf("%c\n", frase[i]);
        i++;
    }
}
```

15

Funzioni di libreria

Libreria `string.h`

```
strcpy(stringa1, stringa2);
consente di copiare stringa2 su stringa1
```

```
strncpy(stringa1, stringa2, n);
consente di copiare i primi n caratteri di stringa2 in stringa1
```

```
strcmp(stringa1, stringa2);
confronta le due stringhe: se sono uguali restituisce 0, se stringa1 è maggiore (lessicograficamente) di stringa2 restituisce un valore positivo, altrimenti un valore negativo
```

16

Cosa stampa?

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

main() {
    char frase1[] = "Ape";
    char frase2[] = "Api";
    printf("%d", strcmp(frase1, frase2));
}
```

-4

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

main() {
    char frase1[] = "Api";
    char frase2[] = "Ape";
    printf("%d", strcmp(frase1, frase2));
}
```

4

17

Esercizi

18

Esercizio

Trovate l'errore in ognuno dei seguenti segmenti di programma e spiegate come correggerlo

- a)

```
char s[10];
strcpy(s, "hello", 5);
```
- b)

```
printf("%s", 'a');
```
- c)

```
if (strcmp(string1, string2))
printf("The strings are equal\n");
```

19

Soluzione

- a)

```
char s[10];
strcpy(s, "hello", 5);
```

Errore: la funzione `strcpy` non inserirà un carattere `NULL` di terminazione nel vettore `s`, perché il suo terzo argomento è uguale alla lunghezza della stringa "hello"
Correzione: cambiare il terzo argomento di `strcpy` in 6
- ```
printf("%s", 'a');
```

**Errore:** tentativo di visualizzare una costante di tipo carattere come se fosse una stringa  
**Correzione:** usare `%c` oppure sostituire `'a'` con `"a"`
- c) 

```
if (strcmp(string1, string2))
printf("The strings are equal\n");
```

**Errore:** la funzione `strcmp` restituisce 0 qualora le stringhe sono uguali, di conseguenza la condizione di `if` risulterà falsa  
**Correzione:** nella condizione confrontare il risultato di `strcmp` con 0

20

## Esercizio

Stampare i singoli caratteri e la codifica ASCII di una stringa

21

## Soluzione

```
#include<stdio.h>
main(){
char stringa[]="questa e' una stringa";
int i = 0;
while (stringa[i] != '\0'){
printf ("%c=%d\n", stringa[i], stringa[i]);
++i;
}
}
```

22

## Output

```
q=113
u=117
e=101
s=115
t=116
a=97
=32
e=101
'=39
=32
u=117
n=110
a=97
=32
s=115
t=116
r=114
i=105
n=110
g=103
a=97
```

## Esercizio

Data una stringa di caratteri, copiarla in un altro array di caratteri (di lunghezza non inferiore).

**Ipotesi**

La stringa è "ben formata", ossia correttamente terminata dal carattere `'\0'`

24

## Soluzione

```
main() {
 char s[] = "Nel mezzo del cammin di";
 char s2[40]; ← La dimensione deve essere tale da
 garantire che la stringa non ecceda
 int i=0;
 for (i=0; s[i]!='\0'; i++)
 s2[i] = s[i];
 s2[i] = '\0';
}
```

Al termine, occorre garantire che anche la nuova stringa sia "ben formata", inserendo esplicitamente il terminatore

25

## Perché non così?

```
main() {
 char s[] = "Nel mezzo del cammin di";
 char s2[40];
 s2 = s;
}
```

ERRORE DI COMPILAZIONE:  
incompatible types in assignment !!

**GLI ARRAY NON POSSONO ESSERE  
MANIPOLATI NELLA LORO INTEREZZA**

26

## Esercizio

Data una stringa di caratteri, scrivere un programma che ne calcoli la lunghezza

### Esempio

"ape"  
stampa 3

27

## Soluzione

```
...
int lung=0;
for (lung=0; s[lung]!='\0'; lung++);
}
```

28

## Esercizio

Date due stringhe di caratteri, decidere quale precede l'altra in ordine alfabetico

29

## Soluzione

Scandire uno a uno gli elementi di ugual posizione delle due stringhe:

1. fino a che se ne trovano due diversi; oppure
2. fino alla fine di una delle stringhe.

Nel primo caso, sono diverse: confrontare i due caratteri così trovati, e determinare qual è il minore;  
la stringa a cui appartiene tale carattere precede l'altra

Nel secondo caso, le stringhe sono uguali se hanno la stessa lunghezza oppure la stringa più corta precede la più lunga:  
esempio: **Maria Marianna**

30

## Soluzione

```
#include<stdio.h>
main(){
 char s1[] = "Maria";
 char s2[] = "Marianna";
 int i=0, stato;

 while (s1[i]!='\0' && s2[i]!='\0' && s1[i]==s2[i]) i++;
 stato = s1[i]-s2[i];
 if (stato > 0) printf("%s precede %s", s2,s1);
 else if(stato<0) printf("%s precede %s", s1,s2);
 else printf("le stringhe sono uguali");
}
}
```

31

## Esercizio

Conta le occorrenze di ciascuna lettera in una stringa data (contenente solo lettere minuscole)

32

## Soluzione

```
#include <stdio.h>
main() {
 char ch, frase[]="questa frase contiene molti caratteri";
 int count[26], j,i ;
 for (j=0; j<26;j++) count[j]=0; // azzera elementi di count[]
 i=0;
 while (frase[i]!='\0'){
 if ((frase[i]>='a') && (frase[i]<='z')){
 ch=frase[i];
 count[ch - 'a']++;
 }
 /* incrementa di 1 il valore dell'elemento di count
 corrispondente al carattere ch in frase */
 i++;
 }
 for (j=0; j<26;j++) printf("%c %d \n", 'a'+j, count[j]);
}
}
```

33

## Esercizio

Scrivere un programma che, richiasta all'utente una stringa, controlli se vi compaiono almeno tre caratteri uguali consecutivi

34