I Vettori di caratteri
Le stringhe

Prof. Francesco Accarino
IIS Altiero Spinelli Sesto San Giovanni

Vettori di caratteri: le stringhe

- Le variabili di tipo char possono contenere un solo carattere: per trattare sequenze di caratteri come nomi o, più in generale, testi, il C prevede le *stringhe*.

- Differentemente dagli altri tipi di dato (intero, reale, ecc.) per le *stringhe* non è sempre possibile fissare a priori le dimensioni: la loro caratteristica peculiare è proprio la lunghezza variabile.

- Per gestire dati di questo tipo occorrerebbe l'allocazione dinamica della memoria, in modo da riservare tutta e solo la memoria che serve.

Vettori di caratteri: le stringhe

- Il C prevede per le stringhe un vettore di caratteri, il quale deve quindi avere una lunghezza massima prefissata.

- All'interno di questo vettore, la lunghezza reale della stringa è determinata dalla presenza di un carattere delimitatore particolare, '\0', detto anche NULL.

- Come per i tipi base, anche per le stringhe è prevista una notazione particolare per indicarne i valori: la sequenza di caratteri deve essere delimitata da una coppia di doppi apici (").

Vettori di caratteri: le stringhe

- Per la definizione di una stringa si può anche utilizzare la direttiva define:

```
#define MESSAGGIO  "Ciao!"
```

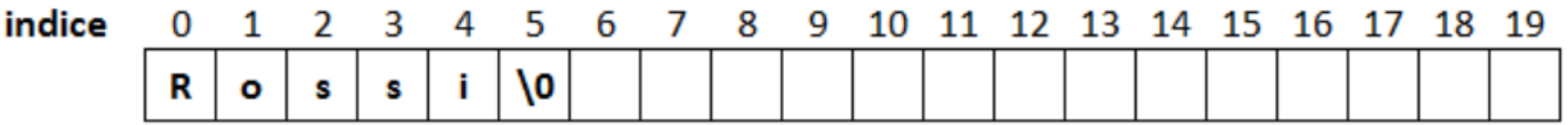
- Attenzione infine a non confondere variabili di tipo carattere con stringhe: per esempio,

  'C' rappresenta un unico carattere che è memorizzato in un'unica cella.

  "C" rappresenta invece una stringa che è memorizzata in due celle consecutive che contengono i caratteri 'C' e '\0'.

Dichiarazione di una stringa

```
char cognome[20] = "Rossi"; //dichiarazione con inizializzazione
```



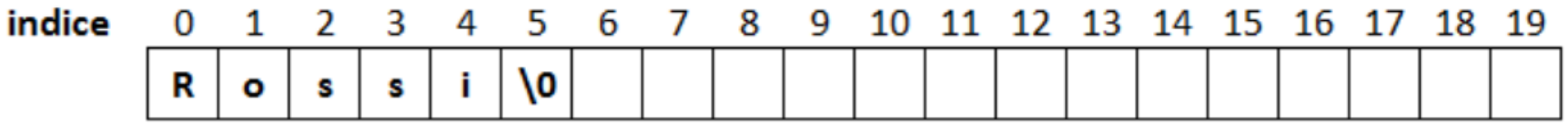
- Con l'istruzione viene dichiarata la variabile 'cognome' come un vettore di 20 char che può essere utilizzato per memorizzare stringhe lunghe al massimo 19 caratteri. Perché deve essere presente il carattere terminatore.
- Con questo tipo di inizializzazione il carattere terminatore viene aggiunto automaticamente.
- Si noti che dopo l'inizializzazione, le celle di indice superiore a 5 non sono "occupate" da caratteri, ma sono vuote solo teoricamente in quanto in realtà esse possono contenere dei valori casuali.
- Di qui l'importanza della presenza del carattere terminatore per capire dove finisce la stringa

Dichiarazione di una stringa

```
char cognome[20] = {'R', 'o', 's', 's', 'i', '\0'};
```

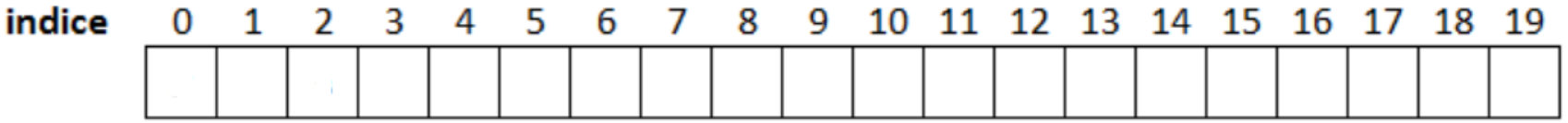
- In questo caso si utilizza la stessa sintassi di un vettore normale e l'inizializzazione avviene passano i singoli caratteri tra le parentesi graffe

- Ovviamente il risultato sarà lo stesso del precedente



Ovviamente si può dichiarare una stringa senza inizializzarla

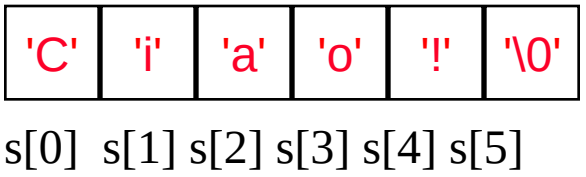
```
char cognome[20] ;
```



In questo caso il vettore è tutto vuoto e il terminatore sarà aggiunto quando introdurremo qualcosa nel vettore

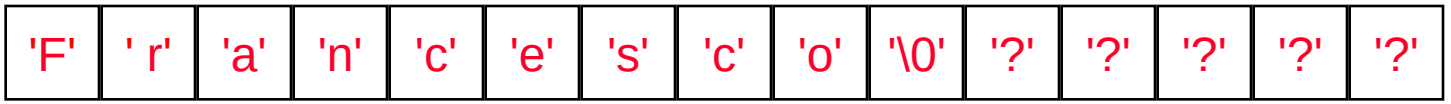
Vettori di caratteri: le stringhe

- Esempio:
  char s[6] = "Ciao!";



- Il messaggio è lungo solo 5 caratteri, ma deve essere riservato un carattere in più per il terminatore di stringa, che deve essere sempre presente e viene forzato automaticamente dal linguaggio C.

- Il vettore può essere definito più lungo, con gli ultimi elementi indefiniti, ma non più corto: char nome[20]= "Francesco"



- NOTA: la *stringa vuota* non è un vettore “vuoto”!

char s[ ] = " ";

I/O di stringhe

- Le stringhe possono comparire come argomento di printf e scanf: per esse si utilizza lo specificatore di formato %s.

- In particolare la printf, quando trova nel format lo specificatore %s, interpreta i valori contenuti nella variabile corrispondente (che deve essere un vettore di caratteri!) come caratteri e li visualizza finché non trova un carattere '\0'.

- Se non è presente il carattere terminatore la printf continua l'output oltre i confini del vettore fino a che non incontra un '\0' (ovvero una cella di memoria che contiene 0!).

I/O di stringhe

Bisogna ricordarsi , che una stringa essendo definita come un vettore di char, il nome della variabile di una stringa è una variabile di tipo puntatore ad un char e, quindi, essa contiene già l'indirizzo della stringa, pertanto:

```
char s[40] ;
scanf("%s", &s); //ERRORE
scanf("%s", s); //CORRETTO
```

Esempio di in put e output con scanf e printf:

```
#include <stdio.h>
int main() {
    char s1[10];
    printf("Inserisci una stringa \n");
    scanf("%s", s1);
    printf("La stringa da te inserita e' %s", s1);
    return 0;
}
```

Manipolazione di stringhe

Attenzione, l'assegnazione di una stringa ad un vettore di char è ammessa solo per l'inizializzazione del vettore nella sua dichiarazione , char s[6] = "Ciao!"; ma non è ammessa in nessun altra istruzione.
Dopo la sua dichiarazione il vettore di char va trattato come un vettore a tutti gli effetti. Per cui non è possibile copiare una stringa in un'altra stringa con un'assegnazione (Per questa e altre operazioni vedremo più avanti le funzioni specifiche che si trovano in string.h)

```
char s1[10]  = "testo 1";
char s2[10]  = "testo 2";
s1 = s2 //ERRORE
```

L'operazione dovrebbe essere fatta con un ciclo:

```
int i=0;
while(s2[i]!='\0') i++) {
   s1[i] = s2[i];
   i++;
}
```

Manipolazione di stringhe

Per la stessa ragione non è possibile confrontare due stringhe trattandole come due comuni variabili:

```
char s1[10]  = "testo 1";
char s1[10]  = "testo 2";
if (s1 == s2) //ERRORE
```

per cui anche i confronti possono essere fatti solo fra gli elementi dei vettori. Solo a scopo didattico, per verificare l'uguaglianza tra s1 e s2 una possibile soluzione (che è da sconsigliare nei casi reali in cui vedremo è possibile utilizzare una opportuna funzione di libreria) potrebbe essere questa di fianco

```
#include <stdio.h>
int main() {
    char s1[10];
    char s2[10];
     int  i ,uguale=1; //ipotizza che s1 e s2 siano uguali

    printf("Inserisci la stringa n. 1\n");
    scanf("%s", s1);
    printf("Inserisci la stringa n. 2\n");
    scanf("%s", s2);

       //verifica se s1 e s2 sono uguali
       for(i=0; s1[i]!='\0'; i++) {
        if (s1[i]!=s2[i]) {
            uguale = 0; //s1 e s2 sono diverse
            break;
        }
    }

 //se è arrivato sul carattere terminatore di s1, verifica che sia così anche per s2
    if (s1[i]='\0' && s2[i]!='\0')
        uguale = 0; //s1 e s2 sono diverse
    if (uguale)
        printf("Le due stringhe sono uguali");
    else
        printf("Le due stringhe sono diverse");
    return 0;
}
```

Funzioni sulle stringhe: strcmp

La libreria standard del C dispone di funzioni molto utili per la gestione delle stringhe. Per poter utilizzare tali funzioni bisogna includere nel proprio file sorgente il file di libreria string.h:

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

strcmp(s1, s2);
La funzione strcmp (string compare) confronta due stringhe passate come parametro e restituisce al chiamante:

- 0, se s1 == s2;
- un numero <0, se s1<s2;
- un numero >0, se s1>s2

- Esempio:

```
char s1[30], s2[30];

if(strcmp(s1, s2) == 0)
      printf("Le due stringhe sono uguali");
else
     if (strcmp(s1, s2) < 0)
         printf(" la prima stringa e' minore della seconda");
else
    printf(" la prima stringa e' maggiore della seconda ");
```

Funzioni sulle stringhe: strcpy

strcpy(destinazione, sorgente);

La funzione strcpy copia la variabile stringa ‘sorgente’ nella variabile stringa ‘destinazione’ . La stringa ‘destinazione’ deve essere sufficientemente grande per accogliere tutti gli elementi della stringa ‘sorgente’.

```
char destinazione[30]= "testo 1";
char sorgente[30 ];
printf("inserisci la stringa da copiare: " );
scanf("%s", sorgente);
strcpy(destinazione, sorgente);
printf("%s", destinazione);
```

Stamperebbe a video la stringa inserita dall’utente

Funzioni sulle stringhe: strlen

strlen(stringa);

La funzione strlen restituisce al chiamante la lunghezza in caratteri della stringa passata come parametro.

Esempio:

```
char s[30];
int n;
printf("Scrivi una parola: ");
scanf("%s", s);
n = strlen(s);
printf("%s e' formata da %d caratteri.", s, n);
```

Funzioni sulle stringhe: strcat

```
strcat(s1, s2);
```

La funzione strcat concatena le due stringhe passate come parametro. Ovviamente la prima stringa deve assere abbastanza capiente da contenerle entrambe.

Esempio:

```
char nome[15], cognome[15], nome_cognome[31];

printf("Inserisci il tuo nome: ");
scanf("%s", nome);
printf("Inserisci il tuo cognome: ");
scanf("%s", cognome);
strcpy(nome_cognome, nome);
strcat(nome_cognome, " ");
strcat(nome_cognome, cognome);
printf("%s", nome_cognome);
```

Per queste e altre funzioni di string.h consultare gli appunti a questo indirizzo

Input e Output con puts e gets

La funzione **gets()** acquisisce una stringa da tastiera, fino alla fine, compresi eventuali spazi e il ritorno a capo che trasforma nel carattere terminatore (\0).

La funzione **puts()** visualizza l'intera riga di testo, ad esempio una stringa inserita da tastiera, compreso il ritorno a capo.

Esempio:

```
#include <stdio.h>
int main(){
  char nome[30];

  printf("inserisci il nome ");
  gets(nome);
  puts(nome);

  return 0;
}
```