Fondamenti di Informatica

Ingegneria Clinica

Lezione 17/01/2011

Prof. Raffaele Nicolussi

FUB - Fondazione Ugo Bordoni

Via del Policlinico, 147 - 00161 Roma

Docente	**Raffaele Nicolussi**	*rnicolussi@fub.it* *0654803323*
Lezioni Aula 54 (ex aula 4) Via del Castro Laurenziano, 7	**Lunedì, Giovedì, Venerdì**	**12:00 – 13:30**
Esercitazioni Aula 15 Via Tiburtina, 205	**Lunedì**	**14:00 – 17:30**
Ricevimento:	**Per appuntamento**	**in FUB, per email, per telefono**
Sito web:	**http://w3.uniroma1.it/IngClinFondinf**	

Le stringhe

- **Uso comune degli array: memorizzazione di stringhe di caratteri**

- **Stringa costante: serie di caratteri racchiusa tra doppi apici**
  - **“HELLO WORLD”**
- **Stringa: array di caratteri che termina SEMPRE con il carattere nullo rappresentato dalla sequenza di escape ‘\0’**

H	E	L	L	O		W	O	R	L	D	\0

- **Per memorizzare una stringa si usa un array di caratteri (memorizzazione su un byte).**

char str [ ] = “some text”;

Le stringhe

- **Le stringhe sono diverse dalle costanti carattere**
  - **"a": array di due elementi**
    ('a' e '\0', char array[2];)
  - **'a': costante carattere**
    - **(char ch = 'a';)**
- **In memoria sarà assegnato**
  - **"a": due byte (uno per 'a' e uno per '\0') consecutivi**
  - **'a' : un byte**

- **"abc" : array di caratteri di dimensione 4 (tre per i caratteri 'a', 'b', 'c' e uno per '\0')**
- **'a'**
- **'b'** } **sono tre costanti carattere**
- **'c'**

- **L'array che rappresenta la stringa ha dimensione maggiore di 1 rispetto la dimensione della stringa**
- **Viene automaticamente aggiunto il carattere di terminazione \0**

s	o	m	e		t	e	x	t	\0

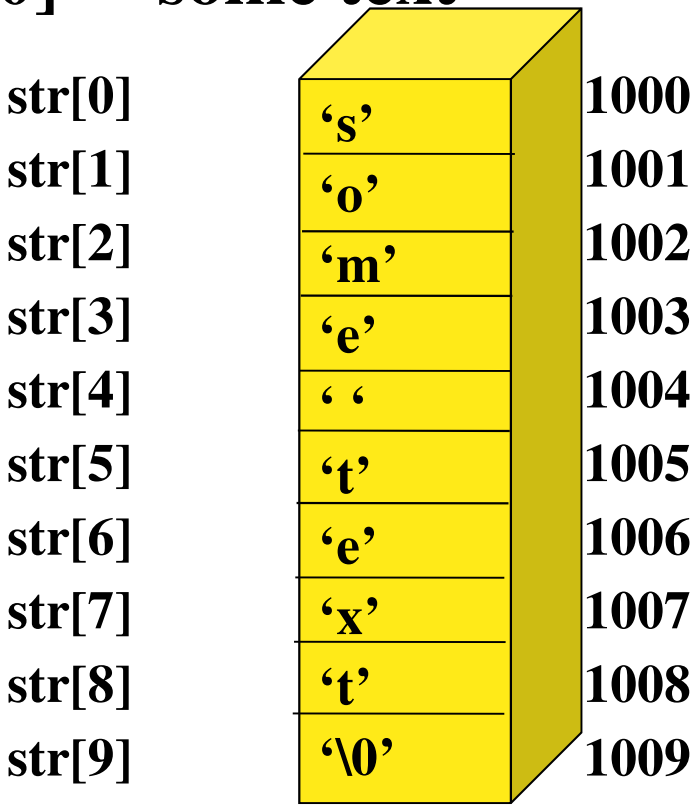
- **Se specifichiamo la lunghezza della stringa, allora il numero di caratteri deve potere essere contenuto nell'array**

  char str [3] = "guarda"; /* Errato */

- **Se la stringa è più corta della lunghezza dell'array, gli elementi mancanti sono inizializzati con 0.**

s	o	m	e		t	e	x	t	\0

char str [10]= “some text”



- **Inizializzazione con singole costanti di carattere nella lista di inizializzazione**

  char str[ ] = {'g', 'u', 'a', 'r', 'd', 'a'};

  - **Sarà aggiunto il carattere di terminazione**
  - **L'array è quindi di 7 elementi**

- **La stringa di caratteri è un array a tutti gli effetti**
  - **Accesso ai singoli elementi con nome e indice**

    str[3] → 'r'

- **Una stringa è un array:**
  - **Viene trattata dal compilatore come un puntatore (nome di array)**
  - **Il suo nome è l'indirizzo del primo elemento della stringa**

Lettura e scrittura

- Si usano le funzioni scanf e printf con lo specificatore di formato %s

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

NB: manca &
(nome_array =indirizzo)

```
#include <stdio.h>
#define N 20

int main (void)        {
   char ar[N]= "Oggi piove.";
   char str[N];
   printf("Introduci una stringa
              \n");
   scanf("%s", str);
   printf("prima stringa: %s\n
              seconda stringa: %s\n",
              ar, str);
   return 0;
}
```

stringa.c

- *scanf*
  - **il suo argomento deve essere un puntatore ad un array di dimensioni sufficienti per contenere la riga di input**
  - **Termine immissione: carattere di invio**
  - **Dopo avere letto la stringa viene automaticamente aggiunto un carattere '\0'**
- *printf*
  - **il suo argomento deve essere un puntatore a un array di caratteri terminato dal carattere nullo**
  - **i caratteri sono stampati fino al terminatore**

```
int main( void )  {
   char *p = "abc";
   printf("%s %s %s \n",
              p, p+1,p+2);
   return 0 ;  }
   /* stampa abc bc c*/
```

stringaP.c

- **Alla variabile p viene assegnato l'indirizzo dell'array di caratteri "abc"**
- **La stampa di un puntatore a char provoca la stampa di ogni carattere successivo dell'array fino a '\0'.**
- **printf(.. p): stampa di abc (p = puntatore di inizio stringa)**
- **printf(.. p+1): stampa di bc (p+1 = puntatore al secondo elemento della stringa)**
- **printf(.. p+2): stampa di c (p+2 = puntatore al terzo elemento della stringa)**

- **Esempio:**
  - **Funzione che conta il numero di parole in una stringa**
  - **Ipotesi: le parole sono separate da caratteri di spaziatura**

- **viene utilizzata la macro isspace ( )**
  - **verifica se il carattere è uno spazio, una tabulazione o un newline**
  - **definita in ctype.h**

- **Algoritmo**
- **Inizializza il contatore di parole cnt=0**
  - **Fino a che non arrivi alla fine della stringa**
    - **Leggi i caratteri della stringa fino a che non trovi un carattere utile (ossia diverso da spazio, una tabulazione o un a capo)**
    - **Se il carattere non è uno spazio, incrementa il contatore di parole**
    - **Salta tutti i caratteri della parola**

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

int word_cnt (char *);

int main( void )     {
   char ar[ ] = "guarda che il
      tempo non promette
      bene";
   int parole;

  parole = word_cnt(ar);
  printf("%d \n", parole);

  return 0 ;
}
```

contaPar.c

```
int word_cnt (char *s) {
   int cnt =0;
   while (*s != '\0')     /*fino a che la stringa NON è
                           finita*/
    {
        while (isspace (*s))        /*salta spazi*/
            s++;
         if (*s != '\0')
           {                        /*trovato inizio parola*/
            ++cnt;
            while(!isspace(*s) &&
                    *s != '\0')
                ++s;                /*salta la parola fino allo
                                    spazio successivo o alla
                                    fine della stringa */
             }
     }
   return cnt;
}
```

contaPar.c