Stringhe

DD Cap. 8 pp. 305-341

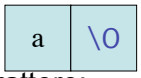
KP Cap. 6 pp. 241-247

Cos'è una stringa (1)

- Una stringa è una serie di caratteri trattati come una singola unità. Essa potrà includere lettere, cifre, simboli e caratteri speciali.
- In C una stringa si memorizza in un vettore di tipo char.
- Per convenzione, in C, una stringa termina con il carattere terminatore '\0', per questo le stringhe hanno delle peculiarità rispetto ai vettori, e verranno trattate a parte.
- La dimensione della stringa DEVE includere anche il carattere '\0'.
- E' utile pensare alle stringhe come aventi lunghezza variabile, delimitata da '\0', entro una lunghezza massima, data dalla lunghezza del vettore. E' compito del programmatore assicurarsi che il limite massimo non venga superato.

Cos'è una stringa (2)

- Una stringa viene scritta tra doppi apici: "ciao".
- "a" e 'a' sono due cose completamente diverse.

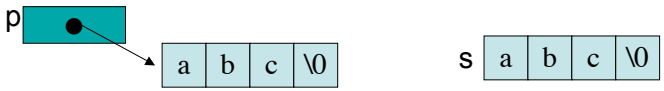
"a": 

- Una stringa viene trattata come un puntatore a carattere:
  Esempio: `char*p="abc";`
  `printf("%s %s", p, p+1);`

Cosa stampa?

abc bc

- Tuttavia ci sono delle differenze:
  Esempio: `char*p="abc"; char s[]="abc";`



Leggere e scrivere stringhe (1)

- Una stringa può essere assegnata ad una zona di memoria in molti modi:
  `char s1[]="ciao";`
  `char s2[]={'c', 'i', 'a', 'o', '\0'};`
  `char s3[5]; scanf("%s3", s3);`
  N.B. In quest'ultima scrittura, non è necessario `&` perché s3 è già un puntatore.
- Una stringa può essere stampata con:
  `printf("%s", s);`
  Questa istruzione funziona correttamente solo se s contiene il carattere di fine stringa '\0'.

Leggere e scrivere stringhe (2)

- Ci sono poi funzioni di libreria che permettono di memorizzare stringhe. Il loro prototipo è nella libreria <stdio.h>

`char *gets(char *s);`
legge dei caratteri dallo standard input e li memorizza nel vettore s finchè non incontra un carattere terminatore. Alla fine del vettore viene accodato il carattere '\0'.

`int puts(char *s);`
visualizza la stringa memorizzata nel vettore s sullo standard output

`int sprintf(char s*, stringa di formato, elenco delle variabili);`
equivalente a printf, eccetto che l'output viene riversato in s anziché nello standard output.

Un esempio

```
#include <stdio.h>

  int main()
  {
    char s[80]; int x; double d;
    printf("inserisci un intero ed un reale");
    scanf("%d%f", &x, &d);
    sprintf(s, "Intero:%d\n Reale:%f",x, d);
    printf("l'output è\n %s");
    return 0;
  }
```

Output: l'output è
Intero: 23
Reale: 43,340000

Funzioni di libreria per la conversione delle stringhe

- Il C fornisce delle funzioni che convertono le stringhe di cifre in formati numerici. I prototipi di quest efunzioni si trovano in stdlib.h

`double atof(char*s);  int atoi(char*s);`
ricevono come parametro una stringa s, la convertono in double oppure in intero e restituiscono il valore numerico.
Se la stringa non può essere convertita (ad es. perché contiene dei caratteri non numerici) il comportamento di queste funzioni è indefinito.

Un esempio

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

  int main()
  {
    double d;
    d=atof("99.0");
    printf("la stringa è convertita in %f", d);
    return 0;
  }
```

Output: la stringa è convertita in 99.000000

Funzioni di libreria per la gestione delle stringhe (1)

- Il C fornisce molte funzioni per la gestione delle stringhe, i cui prototipi si trovano in string.h

`char *strcat(char*s1, char*s2);`
riceve come parametri due stringhe s1 ed s2, le concatena in un'unica stringa e pone il risultato in s1, che restituisce. E' compito del programmatore assicurarsi che s1 sia grande abbastanza.

Funzioni di libreria per la gestione delle stringhe (2)

`int strcmp(char*s1, char*s2);`
riceve come parametri due stringhe s1 ed s2 e restituisce un intero minore, uguale o maggiore di 0, a seconda che s1 sia lessicograficamente minore, uguale o maggiore di s2.

`char *strcpy(char*s1, char*s2);`
riceve come parametri due stringhe s1 ed s2 e restituisce s1, sulla quale è stata copiata s2, quindi il contenuto di s1 va perso. E' compito del programmatore assicurarsi che s1 sia grande abbastanza.

`unsigned int *strlen(char*s);`
riceve come parametro una stringa e ne restituisce la lunghezza (escluso '\0').

Esempi (1)

```
int strcmp(char *s1, char*s2)
{
  while (*s1==*s2)
  {
    if (*s1=='\0') /*quindi anche s2*/
      return 0;
    s1++; s2++;
  }
  if (*s1=='\0')
    return -1;
  if (*s2=='\0')
    return 1;
  return (((*s1-*s2)>0)?1: -1);
}
```

Proviamo con:
case
case
    case
    caserma
        case
        casse

Esempi (2)

```
unsigned int strlen(char *s)
{
  unsigned int n;
  for (n=0; *s !='\0'; s++)
    n++;
  return n;
}
```

```
char* strcpy(char *s1, char *s2)
{
  char *p=s1;
  while (*(p++)=*(s2++));
  return s1;
}
```

```
unsigned int strlenRic(char *s)
{
  if (*s=='\0')
    return 0;
  return (1+strlenRic(s++));
}
```

Funzioni di libreria per la ricerca nelle stringhe

```
char *strchr(char*s, char c);
```
Individua la prima occorrenza del carattere c nella stringa s e restituisce un puntatore alla locazione di c in s trovata. Se c non è in s, restituisce NULL.

```
char *strrchr(char*s, char c);
```
Individua l'ultima occorrenza del carattere c nella stringa s e restituisce un puntatore alla locazione di c in s trovata. Se c non è in s, restituisce NULL.

```
char *strstr(char*s1, char*s2);
```
Individua la prima occorrenza della stringa s2 nella stringa s1 e restituisce un puntatore alla locazione di s2 in s1 trovata. Se s2 non è in s1 restituisce NULL.

Un esempio

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

int main()
{
  char *s="Questa è una prova";
  char c1='a'; char c2='z';
  if (strchr(s,c1)!=NULL) printf("trovato %c",c1);
  else printf("non trovato %c",c1);
  if (strchr(s,c2)!=NULL) printf("trovato %c",c2);
  else printf("non trovato %c",c2);
  return 0;
}
```

Output: trovato a non trovato z

Un altro esempio

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

int main()
{
  char *s="Una zebra è nello zoo";
  char c='z';
  printf("resto della stringa dopo la prima occorrenza:
                %s\n", strchr(s, c));
  printf("resto della stringa dopo l'ultima occorrenza:
                %s\n", strrchr(s, c));
return 0;
}
```

Output: resto della stringa dopo la prima occorrenza: zebra è nello zoo
resto della stringa dopo l'ultima occorrenza: zoo

Un esempio

Problema: scrivere una funzione che prenda come parametro una stringa e restituisca 1 o 0 a seconda che sia palindroma o no.

```
int palindroma(char s[])
  {
    int i, j;
    for (j=0; s[j]!='\0'; j++); /*posizionato j  in fondo*/
    j--;
    for (i=0; ((s[i]==s[j])&&(i<=j)); i++, j--);
    return ((i>j)?1:0);
  }
```

Esercizi

- Riscrivere in C le seguenti funzioni di libreria, in modo che abbiano lo stesso prototipo delle omonime funzioni di libreria:

  Strcat Strchr Strrchr
- Progettare e scrivere un programma che prenda in input 4 stringhe che rappresentino degli interi, le converta in double, sommi i valori ottenuti e visualizzi il totale.
- Progettare e scrivere un programma che prenda in input una riga di testo ed un carattere da ricercare, e -utilizzando la funzione strchr- determini il numero delle occorrenze del carattere nel testo.
- Progettare e scrivere un programma che prenda in input una riga di testo e -utilizzando la funzione strchr- determini il numero delle occorrenze di ciascun carattere dell'alfabeto; le maiuscole e le minuscole devono essere conteggiate insieme. Memorizzare i risultati in un vettore e visualizzarli in forma tabulare.