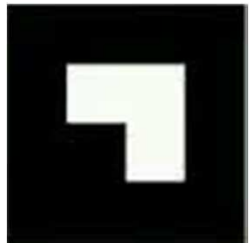
Stringhe

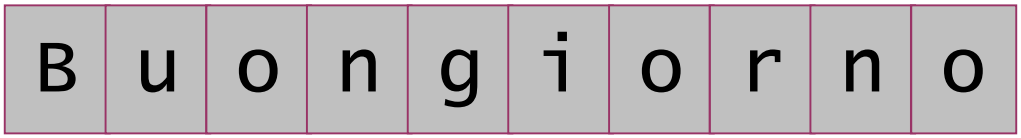


STRINGHE

Stringhe

- Nel linguaggio C non è supportato esplicitamente alcun tipo di dato "stringa"
- Le informazioni di tipo stringa vengono memorizzate ed elaborate ricorrendo a semplici **vettori di caratteri**

```
char saluto[10] ;
```



Stringhe (Cont.)

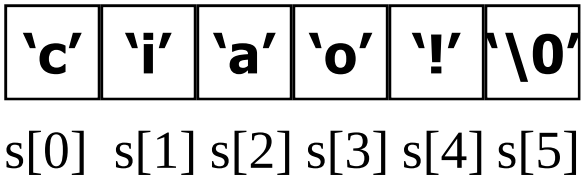
- Definizione:
  Sequenze di caratteri terminate dal carattere '\0' (NULL)

- Tecnicamente:
  Vettori di caratteri terminati da un carattere aggiuntivo '\0' (NULL)

- Memorizzate come i vettori

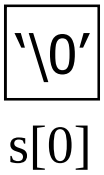
- La lunghezza della stringa può essere definita implicitamente mediante l'assegnazione di una costante stringa, rappresentata da una sequenza di caratteri racchiusa tra doppi apici

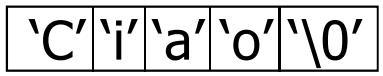
- Esempio:
  `char s[] = "ciao!";`

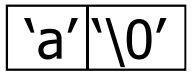


Stringhe (Cont.)

- NOTA: La stringa vuota non è un vettore "vuoto"!
  - Esempio: `char s[] = "";`



- Attenzione: 'a' è diverso da "a"
- Infatti 'a' indica il carattere a, mentre "a" rappresenta la stringa a (quindi con '\0' finale).
- Graficamente:
  - "Ciao" ----> 

  - "a" ----> 

  - 'a' ----> 

Formattazione di stringhe

- Le operazioni di I/O formattato possono essere effettuate anche da/su stringhe

- Funzioni

  `int sscanf(char* <`***stringa***`>,char* <`***formato***`>,<`***espressioni***`>);`

  - Restituisce `EOF` in caso di errore, altrimenti il numero di campi letti con successo

  `int sprintf(char* <`***stringa***`>,char* <`***formato***`>,<`***variabili***`>));`

  - Restituisce il numero di caratteri scritti

- Utili in casi molto particolari per costruire/analizzare stringhe con un formato fisso

I/O di stringhe

- Diamo per scontato di utilizzare la convenzione del terminatore nullo
- Si possono utilizzare
  - Funzioni di lettura e scrittura carattere per carattere
    - Come nell'esercizio precedente
  - Funzioni di lettura e scrittura di stringhe intere
    - `scanf` e `printf`
    - `gets` e `puts`

Esempio

```
const int MAX = 20 ;
char nome[MAX+1] ;

printf("Come ti chiami? ") ;

scanf("%s", nome) ;
```

Esempio

```
const int MAX = 20 ;
char nome[MAX+1] ;

printf("Come ti chiami? ") ;

gets(nome) ;
```

Esempio

```
printf("Buongiorno, ") ;
printf("%s", nome) ;
printf("!\n") ;


printf("Buongiorno, %s!\n", nome) ;
```

Esempio

```
printf("Buongiorno, ") ;
puts(nome) ;

/* No!! printf("!\n") ; */
```

Manipolazione di stringhe

- Data la loro natura di tipo “aggregato”, le stringhe non possono essere usate come variabili qualunque

- Esempi di operazioni non lecite:

```
char s1[20], s2[10], s3[50];
...
s1 = “abcdefg”;
s2 = “hijklmno”;
s3 = s1 + s2;
```

NO!

- A questo scopo esistono apposite funzioni per la manipolazione delle stringhe

Funzioni di libreria per stringhe

- Utilizzabili includendo in testa al programma

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

funzione	*definizione*
`char* strcat (char* s1, char* s2);`	concatenazione s1+s2
`char* strchr (char* s, int c);`	ricerca di c in s
`int   strcmp (char* s1, char* s2);`	confronto
`char* strcpy (char* s1, char* s2);`	s1 <= s2
`int   strlen (char* s);`	lunghezza di s
`char* strncat (char* s1,char* s2,int n);`	concat. n car. max
`char* strncpy (char* s1,char* s2,int n);`	copia n car. max
`char* strncmp(char* dest,char* src,int n);`	cfr. n car. max

Funzioni di libreria per stringhe (Cont.)

- NOTE:
  - Non è possibile usare vettori come valori di ritorno delle funzioni di libreria
    - Esempio:

```
char s[20]
...
s = strcat(stringa1, stringa2);  /* NO! */
```

  - Alcune funzioni possono essere usate "senza risultato"
    - Esempio:

      `strcpy(<`***stringa destinazione***`>, <`***stringa origine***`>)`

      `strcat(<`***stringa destinazione***`>, <`***stringa origine***`>)`

      - Il valore di ritorno coincide con la stringa destinazione

Esercizio 1

- Realizzare un meccanismo di "cifratura" di un testo che consiste nell'invertire le righe del testo stesso

- Esempio:

C'era una volta un re che ...		atlov anu are'C ehc er nu

Esercizio 1: Soluzione

```
#include <stdio.h>
main()
{
  int i,j, len;
  char s[80], dest[80];   /* una riga */

  while (gets(s) != NULL) {
       len = strlen(s); j = 0;
       for (i=len-1;i>=0;i--) {
             dest[j] = s[i];
             j++;
       }
       dest[j]='\0';
       puts(dest);
  }
}
```

Esercizio 2

- Si scriva un programma che legga da tastiera due stringhe e cancelli dalla prima stringa i caratteri contenuti nella seconda stringa

- Esempio:
  - str1: “Olimpico”
  - str2: “Oio”
  - risultato: “lmpc”

Esercizio 2: Soluzione

```
#include <stdio.h>
#define MAXCAR 128

char *elimina(char str1[], char str2[]);

main()
{
   char str1[MAXCAR], str2[MAXCAR];
   printf("Dammi la stringa str1: ");
   scanf("%s", str1);
   printf("Dammi la stringa str2: ");
   scanf("%s", str2);
   printf("str1-str2= %s\n", elimina(str1,str2));
}
```

Esercizio 2: Soluzione (Cont.)

```
char *elimina(char str1[], char str2[])
{
   int i, j, k;

   for(i=j=0;str1[i]!= '\0';i++)
   {
      for(k=0;(str2[k]!= '\0') && (str1[i]!=str2[k]);k++);
      if(str2[k]== '\0')
         str1[j++]=str1[i];
   }
   str1[j]='\0';
   return str1;
}
```

I/O a righe

- Acquisizione/stampa di una riga alla volta
  - Riga = Serie di caratteri terminata da '\n'

- Istruzioni:
  - `char *gets(<`***stringa***`>)`
    - Legge una riga da tastiera (fino al '\n')
    - La riga viene fornita come stringa (<***stringa***>), senza il carattere '\n'
    - In caso di errore, il risultato è la costante `NULL` (definita in `stdio.h`)
  - `int puts(`***<stringa>***`)`
    - Stampa <***stringa***> su schermo
    - Aggiunge sempre '\n' alla stringa

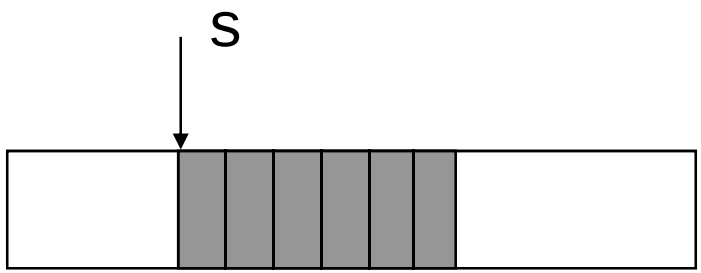
I/O a righe (Cont.)

- L'argomento di gets/puts è di tipo "puntatore" (discussione più avanti), definito come segue:

`char*`

- Significato: Il puntatore ad una stringa contiene l'indirizzo di memoria in cui il primo carattere della stringa è memorizzato

- Esempio:
  - `char*  s;`



I/O a righe (Cont.)

- NOTE:
  - `puts/gets` sono "costruite" a partire da `getchar/putchar`
  - Uso di `gets` richiede l'allocazione dello spazio di memoria per la riga letta in input
    - Gestione dei puntatori che vedremo più avanti
  - `puts(s)` è identica a `printf("%s\n",s);`

- Usate meno di frequente delle altre istruzioni di I/O

I/O a righe: Esempio

- Programma che replica su video una riga di testo scritta dall'utente

```
#include <stdio.h>

main()
{
  char *s, *res;
  printf("Scrivi qualcosa\n");
  //res = gets(s);
  if (res != NULL)   // if (gets(s)!= NULL) /* errore ? */
  {
       puts("Hai inserito"); //printf("Hai inserito\n");
       puts(s); //printf("%s\n",s);
  }
}
```

Fine Capitolo