Stringa

- Strings are pieces of text which can be treated as values for variables. In C a string is represented as some characters enclosed by double quotes.

"This is a string"

- A string may contain any character, including special control characters, such as \n, \r, \7 etc...

"Beep! \7 Newline \n..."

Stringa

- È una sequenza di caratteri

- In C non esiste il dato di tipo stringa; una stringa viene memorizzata in un array di caratteri

`char nome[30]`

- Il C usa la tecnica di contrassegnare la fine effettiva di una stringa con il carattere avente codice 0 ('\0')

- Inizializzazione di una stringa:
  `char nome[30]="Luca"` (inserisce automaticamente il carattere '\0')

Array e stringhe

In C le *stringhe*, ovvero le sequenze ordinate di caratteri, sono rappresentate attraverso *array monodimensionali* di caratteri, dove la fine della stringa è segnalata dal carattere nullo ('\0').

I valori di tipo stringa vengono racchiusi tra *doppi apici* (come per i parametri della funzione printf), e possono essere utilizzati per inizializzare variabili di tipo 'stringa'.

Ad esempio, una variabile dichiarata da

```
int        char stringa[5] = "cane";
```

corrisponde alla dichiarazione `char stringa[5];` **seguita dalle assegnazioni**

```
int        stringa[0] = 'c';
int        stringa[1] = 'a';
int        stringa[2] = 'n';
int        stringa[3] = 'e';
int        stringa[4] = '\0';
```

Assegnamenti di valori Stringa

It is legal to *initialize* a string variable, like this.

```
char example[100] = "First value";
```

However, it is not legal to *assign* string variables, because you cannot assign to an entire array.

```
myString = "xyz"; !!!!! ILLEGAL: Cannot assign to an array
```

Furthermore, you cannot do comparisons like this.

```
if (myString == "xyz") printf(“bla bla”);
```

The comparison

```
myString == "xyz“; !!!!! Non usare
```

is actually legal (it will compile), but it will always evaluate to false.

Lunghezza di una stringa

- Per calcolare la lunghezza della stringa basta scandire l'array e cercare il carattere terminatore

```
int main() {
   int i;

   i=0;
   while (miastringa[i]|='\0') {
      i++;
   }
   printf("La lunghezza è %d", i);
}
```

- La dimensione della variabile `miastringa` è sempre di 20 caratteri ed è fissato a tempo di compilazione come tutti gli array
- È il contenuto della variabile `miastringa` che può variare anche a tempo di esecuzione ma non può essere mai superiore a 19

Copia di una stringa

- Come per tutti gli array, date due stringhe a e b il seguente assegnamento non produce una copia

```
a = b;
```

- È necessaria la copia elemento per elemento

```
for (i = 0; i<DIM; i++)
   a[i] = b[i];
```

#include <string.h>

In C esistono diverse funzioni per poter manipolare le stringhe, alcune di queste sono:

Per poter usare queste funzioni deve essere incluso il file d'intestazione standard *string.h*.

NOME	FUNZIONE
`strcpy (a1, a2)`	Copia a2 in a1
`strcat (a1, a2)`	Concatena a2 alla fine di a1
`strlen (a1)`	Ritorna la lunghezza di a1
`strchr (a1, ch)`	Restituisce una stringa che inizia dalla prima occorrenza di ch in a1
`strcmp (s1, s2)`	Confronta s1 con s2

Libreria standard sulle stringhe

Il C fornisce una libreria standard di funzioni per la gestione di stringhe.
Per poterla utilizzare e` necessario includere il file header <string.h>:

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

Tra tutte, le funzioni piu` comunemente utilizzate sono:

- **strlen**
- **strcmp**
- **strcat**
- **strcpy**

Libreria standard sulle stringhe

- **Lunghezza di una stringa:**

int strlen(char str[ ]);

restituisce la lunghezza (cioe`, il numero di caratteri significativi) della stringa str specificata come argomento.

Ad esempio:

```
char S[10]="bologna";
int k;
…
k=strlen(S); /* k vale 7*/
```

- **Confronto tra stringhe:**

int strcmp(char str1[ ], char str2[ ])

esegue il confronto tra le due stringhe date str1 e str2. Restituisce:

- **0** se le due stringhe sono identiche;
- un **valore negativo**(ad esempio, -1), se str1 precede str2 (in ordine lessicografico);
- un **valore positivo** (ad esempio, +1), se str2 precede str1 (in ordine lessicografico).

Ad esempio:

```
char S1[10]="bologna";
char S2[10]="napoli";
int k;
…
k=strcmp(S1,S2); /* k < 0 */
k=strcmp(S1,S1); /* k=0*/
k=strcmp(S2,S1); /* k>0 */
```

- **Concatenamento di stringhe:**

strcat(char str1[ ], char str2[ ])

concatena le 2 stringhe date str1 e str2. Il risultato del concatenamento e` in str1.

Ad esempio:

```
char S1[15]="reggio";
char S2[15]="emilia";
strcat(S1, S2); /*S1="reggioemilia"*/
```

- **Copia di stringhe:**

strcpy(char str1[ ], char str2[ ]);

copia la stringa str2 in str1.

Ad esempio:

```
char S1[10]="Giuseppe";
char S2[10];
…
strcpy(S1,S2); /* S1="Giuseppe"*/
```

strlen

- Restituisce la lunghezza della stringa

```
int main () {

    char s[10]="pippo";
    int l;

    l = strlen(p);
    printf("La lunghezza della stringa è: %d", l);

    return 0;
}
```

- A destra dell'operatore di assegnamento c'è l'invocazione ad una funzione
- Indica che il risultato dell' funzione viene inserito nella variabile a sinistra dell'operatore di assegnamento

strcpy

- Copia il contenuto della stringa passata come secondo parametro nella stringa passata come primo parametro

```
int main () {

    char sorgente[10]="pippo";
    char destinazione[10];
    int l;

    strcpy(destinazione, sorgente);
    printf("Il valore di destinazione è: %s",
            destinazione);

     return 0;
 }
```

- L'eventuale contenuto di `destinazione` precedente all'assegnamento viene perso

Comparazione strcmp

- The strcmp() function *lexically* compares the two input strings and returns:
  - **Less than zero** -- if string1 is lexically less than string2
  - **Zero** -- if string1 and string2 are lexically equal
  - **Greater than zero** -- if string1 is lexically greater than string2
- This can also confuse beginners and experience programmers forget this too.

strcmp 1/2

- Confronta il contenuto di due stringhe
- Restituisce:
  - 0 se le stringhe sono identiche
  - <0 se la stringa passata come primo parametro è minore della atringa passata come secondo paramentro
  - >0 se la stringa passata come primo parametro è maggiore della atringa passata come secondo paramentro
- La relazione d'ordine tra stringhe è definita dalla relazione d'ordine della codifica ASCII dei caratteri che la compongono
  - '0' < '9' < 'A' < 'Z' < 'a' < 'z'

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

int main ()
    {
char s1[10]="mamma";
char s2[10]="ciao";
int c;
    c = strcmp(s1, s2);
    if (c==0) {
    printf("\nle due stringhe sono uguali\n");
    } else if (c<0) {
    printf("\ns1 < s2\n");
    } else {
    printf("\ns1 > s2\n");
    }
return 0;
}
```

strcat

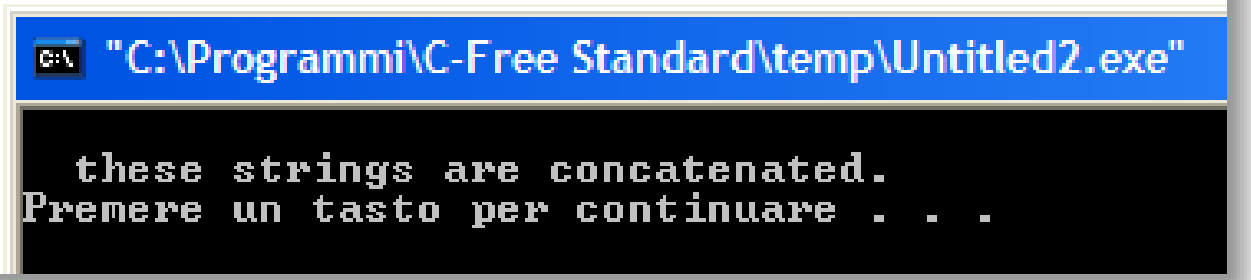
- Accoda il contenuto della stringa passata come secondo parametro nella stringa passata come primo parametro

```
int main () {

    char sorgente[10]="mamma";
    char destinazione[10]="ciao";
    int l;

    strcpy(destinazione, sorgente);
    printf("Il valore di destinazione è: %s",
            destinazione);

     return 0;
 }
```

- Devo essere certo che la dimensione massima della stringa `destinazione` sia sufficiente ad ospitare la nuova stringa

Concatenazione

```c
/* strcat example */
#include <stdio.h>
#include <string.h>

int main ()
{
  char str[80];
  strcpy (str,"these ");
  strcat (str,"strings ");
  strcat (str,"are ");
  strcat (str,"concatenated.");
  printf ("\n  %s\n",str);
  return 0;
}
```



Scrittura di una stringa su schermo

```c
#include <stdio.h>
#include <string.h>
#define MAX 20
int main(int argc, char *argv[])
{
    char stri1[20]="prima", stri2[MAX]="prova";
    printf ("%s %s\n", stri1, stri2);

    strcpy (stri1, stri2);
    printf ("%s %s\n", stri1, stri2);

    return 0;
}
```

```
"C:\Programmi\C-Free Standard\temp\Untitled1.exe"
prima prova
prova prova
Premere un tasto per continuare . . .
```

Uso delle funzioni di <string.h>

funz-stringhe.c

```c
#include <stdio.h>
#include <string.h>
#define MAX 20
int main(int argc, char *argv[])
{
    char stri1[MAX]="prima", stri2[MAX]="prova";

    printf ("A)\t%s %s\n", stri1, stri2);

    strcpy (stri1, stri2);
    printf ("B)\t%s %s\n", stri1, stri2);

    strcat (stri1, stri2);
    printf ("C)\t%s %s\n", stri1, stri2);

    printf("D)\t%d\n",strlen (stri1));

    printf("E)\t%s\n",strchr (stri1, 'o'));

    if(strcmp (stri1, stri2)!=0)
        printf ("F)\t%s  diversa da  %s\n",stri1, stri2);
        else
        printf ("F)\t%s  uguale a  %s\n", stri1, stri2);

    return 0;
}
```

```
"C:\Documents and Settings\DiDA\Documenti\9. PC
A)      prima prova
B)      prova prova
C)      provaprova prova
D)      10
E)      ovaprova
F)      provaprova  diversa da  prova
Premere un tasto per continuare . . .
```

Vediamo ora come sia possibile utilizzare il terminatore NULL ('\0') per troncare una stringa assegnata; in questo caso effettueremo l'assegnazione dei singoli elementi che costituiscono l'array, non utilizzando l'inizializzazione automatica vista in precedenza. Il programma riportato effettua l'assegnazione della stringa "Stringa test" e la tronca al sesto carattere:

```
/*************************************************************
 File: str2.c
 Desc: utilizzo del terminatore
 Comm:
*************************************************************/
#include <stdio.h>
#include <string.h>

int main (void)
{
  char str[13];                                 /* 001 */

  str[0]  = 'S';                                /* 002 */
  str[1]  = 't';
  str[2]  = 'r';
  str[3]  = 'i';
  str[4]  = 'n';
  str[5]  = 'g';
  str[6]  = 'a';
  str[7]  = ' ';
  str[8]  = 't';
  str[9]  = 'e';
  str[10] = 's';
  str[11] = 't';
  str[12] = '\0';

  printf("str: %d caratteri\n", strlen(str));   /* 003 */
  printf("str: %s\n\n", str);                   /* 004 */

  str[5] = '\0';                                /* 005 */

  printf("str: %d caratteri\n", strlen(str));   /* 003 */
  printf("str: %s\n\n", str);                   /* 0004 */

  return(0);
}
/******* End ********************************************/
```

Il programma produce il seguente output:

str: 12 caratteri
str: Stringa test

str: 5 caratteri
str: Strin

Analizziamo ora le istruzioni:

001 Dichiarazione di un array di caratteri di 13 elementi.

002 Assegnazione ad ogni elemento del rispettivo carattere costituente la stringa "Stringa test", e assegnazione del carattere '\0' (NULL) all'ultimo elemento.

003 Calcolo della lunghezza della stringa (strlen()) e output del valore restituito, questo è un esempio di nidificazione di operazioni di cui parleremo più diffusamente trattando degli operatori.

004 Output della stringa "str" mediante il carattere di conversione %s nello specificatore di formato della funzione printf().

005 Assegnazione del carattere '0\' (NULL) al sesto elemento dell'array (str[5]) e conseguente troncamento della stringa. Ripetendo i passi 003 e 004 otterremo ora i risultati relativi alla stringa troncata

Es.: Alfabeto Morse: numeri

- 1: · – – – – –
  2: · · – – –
  3: · · · – –
  4: · · · · –
  5: · · · · ·
  6: – · · · ·
  7: – – · · ·
  8: – – – · ·
  9: – – – – ·
  0: – – – – –

Morse code program.

Enter a number and find out what it is in Morse code

```
#include <stdio.h>
#include <string.h>
/************************************/
main ()
{ short digit;  char string[10];

printf ("Enter any digit in the range
    0..9\t");
scanf ("%d",&digit);
if ((digit < 0) || (digit > 9))
   {
   printf ("\nNumber was not in range
    0..9");
   return (0);
   }
printf ("\nThe Morse code of that digit
    is \t");
ToMorse (digit);
 }
```

```
/************************************/
ToMorse ( digit)        /* print out
    Morse code */
{
  static char code[][6]=
     {
     "-----",
     ".----",
     "..---",
     "...--",
     "....-",
     ".....",
     "-....",
     "--...",
     "---..",
     "----.",
     };
printf ("%s\n",code[digit]);
};
```

Morse code program.

Enter a Morse code and find out the number

```
#include <stdio.h>
#include <string.h>
/************************************/
main ()
{ short digit;  char string[10];

printf ("Adesso dammi una stringa
    Morse\t");

scanf("%s",string);
printf("\n\t%s",string);

FromMorse(string);
 }
```

```
FromMorse ( char string[])
{ int I;
static char letter[]=
  { '0‘, '1‘, '2‘, '3‘,'4‘,'5',
     '6‘,'7‘, '8‘, '9‘, };
static char code[][6]=
   { "-----“, ".----“, "..---",
     "...--“, "....-“, ".....",
     "-....“, "--...“, "---..",
     "----.“       };

for(I=0;I<10 ;I++ )
{
if (strcmp (string, code[I])==0 )  break;
};

printf ("il carattere morse %s
    corrisponde a %d\n",  code[I] ,I    );
}
```

Modificare il programa per riconoscere una frase

- Estendere le tabelle con le lettere dell'alfabeto
- Scrivere prima il ToMorse esteso
  - Input carattere
  - Output Morse
- Poi modificare il FromMorse
  - Leggere una sequenza di stringhe (ognuna è un codice Morse)
  - Ogni volta cercare e stampare il carattere relativo
  - Eseguire le operazioni in loop sino a che si inserisce un codice errato

Alfabeto Morse: lettere

- A: · –
  B: – · · ·
  C: – · – ·
  D: – · ·
  E: ·
  F: · · – ·
  G: – – ·
  H: · · · ·
  I: · ·
  J: · – – –
  K: – · –
  L: · – · ·
  M: – –
- N: – ·
  O: – – –
  P: · – – ·
  Q: – – · –
  R: · – ·
  S: · · ·
  T: –
  U: · · –
  V: · · · –
  W: · – –
  X: – · · –
  Y: – · – –
  Z: – – · ·

Codice Morse Completo

Lettere, numeri e punteggiatura

Lettere	Codice	Lettere	Codice	Numeri	Codice	Punteg.	Codice
A	•—	N	—•	0	—————	•	•—•—•—
B	—•••	O	———	1	•————	,	——••——
C	—•—•	P	•——•	2	••———	:	———•••
D	—••	Q	——•—	3	•••——	?	••——••
E	•	R	•—•	4	••••—	=	—•••—
F	••—•	S	•••	5	•••••	-	—••••—
G	——•	T	—	6	—••••	(	—•——•
H	••••	U	••—	7	——•••	)	—•——•—
I	••	V	•••—	8	———••	"	•—••—•
J	•———	W	•——	9	————•	'	•—————
K	—•—	X	—••—			/	—••—•
L	•—••	Y	—•——			Sottolineato	•——•—
M	——	Z	——••			@	•——•—•