

Informatica 1

Corso di Laurea Triennale in Matematica

Gianluca Rossi

`gianluca.rossi@uniroma2.it`

Dipartimento di Matematica
Università di Roma "Tor Vergata"

11: Stringhe



- `char c = 97`: Alla variabile `c` viene assegnato il codice valore 97 (codice ASCII del carattere `a`).
- `char c = 'a'`: Alla variabile viene assegnato il codice ASCII del carattere `a` (l'intero 97).
- Una stringa (sequenza di caratteri) è un vettore di `char`.



```
char s[100];
```

- Definisce una stringa di lunghezza al più 100.
- s può contenere stringhe più piccole: L'ultimo carattere deve essere seguito dal carattere speciale di fine stringa `'\0'`.
- È buona norma utilizzarlo sempre.
- La stampa della stringa con la **printf** avviene per mezzo del descrittore `%s`

```
printf(" %s ", s);
```



Stringhe: Esempio

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

```
main(){
```

```
    char s[100];
```

```
    s[0] = 'c';
```

```
    s[1] = 'o';
```

```
    s[2] = 'm';
```

```
    s[3] = 'p';
```

```
    s[4] = 'i';
```

```
    s[5] = 'l';
```

```
    s[6] = 'a';
```

```
    s[7] = 't';
```

```
    s[8] = 'o';
```

```
    s[9] = 'r';
```

```
    s[10] = 'e';
```

```
    s[11] = ' ';
```

```
    s[12] = 'c';
```

```
    s[13] = '\\0'; /*eol*/
```

```
    s[14] = 'x'; /* ? */
```

```
    printf("%s\\n", s);
```

```
}
```

- Il carattere di fine-stringa (*EOL*) determina la fine effettiva della stringa;
- Tutti i caratteri che seguono vengono ignorati dalla funzione **printf**;
- Se non fosse indicato il termine della stringa verrebbero stampati tutti i caratteri del vettore *s* che, in questo caso, sarebbero indefiniti.



Definizione implicita

```
char s[] = "ecco una stringa";
```

Crea il vettore di caratteri della dimensione “giusta” per memorizzare la stringa di inizializzazione.

e	c	c	o		u	n	a		s	t	r	i	n	g	a	\0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16




Esempio: Lunghezza di una stringa

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

```
int stringlength(char*);
```

```
main(){  
    char s[] = "ecco una stringa";  
    printf("%d\n", stringlength(s));  
    s[4] = '\\0';  
    printf("%d\n", stringlength(s));  
    printf("%d\n", sizeof(s));  
}
```

```
int stringlength(char *s){  
    int l = 0;  
    while (*s != '\\0'){  
        s++;  
        l++;  
    }  
    return l;  
}
```



A terminal window titled "Terminale — bash — 43x9" showing the execution of the program. The user is at the prompt "gianluca@iggy-mac.local [04/28 23:22:06 - 11]". They run "gcc 02-esempio1.c", then "./a.out". The output shows three lines: "16", "4", and "17". The user then runs "gianluca@iggy-mac.local [04/28 23:22:20 - 11]" and the prompt "\$> |" is shown.



```
char s[100];  
scanf("%s", s);
```

- Non c'è controllo sulla lunghezza della stringa inserita da tastiera;
- L'immissione termina con lo spazio, i caratteri che seguono sono ignotati;
- L'alternativa consiste nell'inserire un carattere alla volta.



```
int getchar();
```

- Funzione di libreria `stdio`;
- Prende l'input da tastiera, un carattere alla volta. Se l'input è un carattere ASCII lo restituisce in output;
- Altrimenti (errore o fine-riga) restituisce `EOF` (costante definita in `stdio.h`).



Input di stringhe: Esempio

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

```
int getstr(char[], int);
```

```
main(){  
    char s[100];  
    getstr(s, sizeof(s) - 1);  
    printf("%s\n", s);  
}
```

```
int getstr(char s[], int dim){  
    int i = 0, c;  
    while( (c = getchar()) != '\n' && i < dim )  
        s[i++] = c;  
    s[i] = '\0';  
}
```



malloc

```
void *malloc(int num_bytes);
```

- Nella libreria `stdlib` (includere `stdlib.h`);
- Alloca `num_bytes` byte consecutivi e restituisce l'indirizzo del primo byte allocato;
- Il tipo `void` è da considerare come tipo jolly. Il puntatore di ritorno deve essere convertito (*casting*) al tipo desiderato.



Esempio: Concatenazione di due stringhe

```
#include<stdio.h>
#include<stdlib.h>
int stringlenght(char*);

main(){
    char s1[] = "primastringa";
    char s2[] = "secondastringa";
    char *s;
    int i, j;
    int len1 = stringlenght(s1);
    int len2 = stringlenght(s2);
    s = (char*)malloc((len1+len2+2)*sizeof(char));
    i=0;
```

```
    /*copia di s1 nella prima parte di s*/
    for(j = 0; j < stringlenght(s1); j++)
        s[i++] = s1[j];
    /*uno spazio vuoto*/
    s[i++] = ' ';
    /*copia di s2 nella seconda parte di s*/
    for(j = 0; j < stringlenght(s2); j++)
        s[i++] = s2[j];
    s[i] = '\0';
    printf("%s\n", s);
}
```

```
int stringlenght(char *s){
    int l = 0;
    while (*s != '\0'){
        s++;
        l++;
    }
    return l;
}
```



Libreria **string**

Includere `string.h`

`int strlen(char *str);` Restituisce la lunghezza della stringa in input.

`char *strcpy(char *str1, char *str2);` La stringa `str2` viene copiata in `str1`, viene restituito il riferimento a `str1`.

`char *strcat(char *str1, char *str2);` La stringa `str2` viene accodata a `str1`, viene restituito il riferimento a `str1` (nessun controllo sulle dimensioni).

`int strcmp(char *str1, char *str2);` $\left\{ \begin{array}{ll} < 0 & \text{se } str1 < str2 \\ = 0 & \text{se } str1 = str2 \\ > 0 & \text{se } str1 > str2 \end{array} \right.$

The C Library Reference Guide

http://www.acm.uiuc.edu/webmonkeys/book/c_guide/



Esempio: una implementazione della **strcmp**

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

```
int stringcompare(char*, char*);
```

```
main(){  
    printf("%d\n", stringcompare("progettazione", "programmazione"));  
}
```

```
int stringcompare(char *str1, char *str2){  
    int i = 0;  
    while( str1[i] != '\0' && str2[i] != '\0' && str1[i] == str2[i] )  
        i++;  
    if(str1[i] == str2[i] ) /* str1[i] = str2[i] = '\0' */  
        return 0;  
    /* altrimenti str1[i] != str2[i] oppure solo una tra str1[i] e str2[i] e' '\0' */  
    if(str1[i] == '\0' || str1[i] < str2[i] )  
        return -1;  
    return 1;  
}
```

