Stringhe: Array di Caratteri

Informatica A AA 2023/24

Giacomo Boracchi

giacomo.boracchi@polimi.it

4 Ottobre 2023

Array di Caratteri: le stringhe

Nel C le stringhe (sequenze ininterrotte di caratteri) sono realizzate mediante array di caratteri

Esempio

```
char luogo[100];
```

è un array atto a contenere 100 elementi di tipo **char**

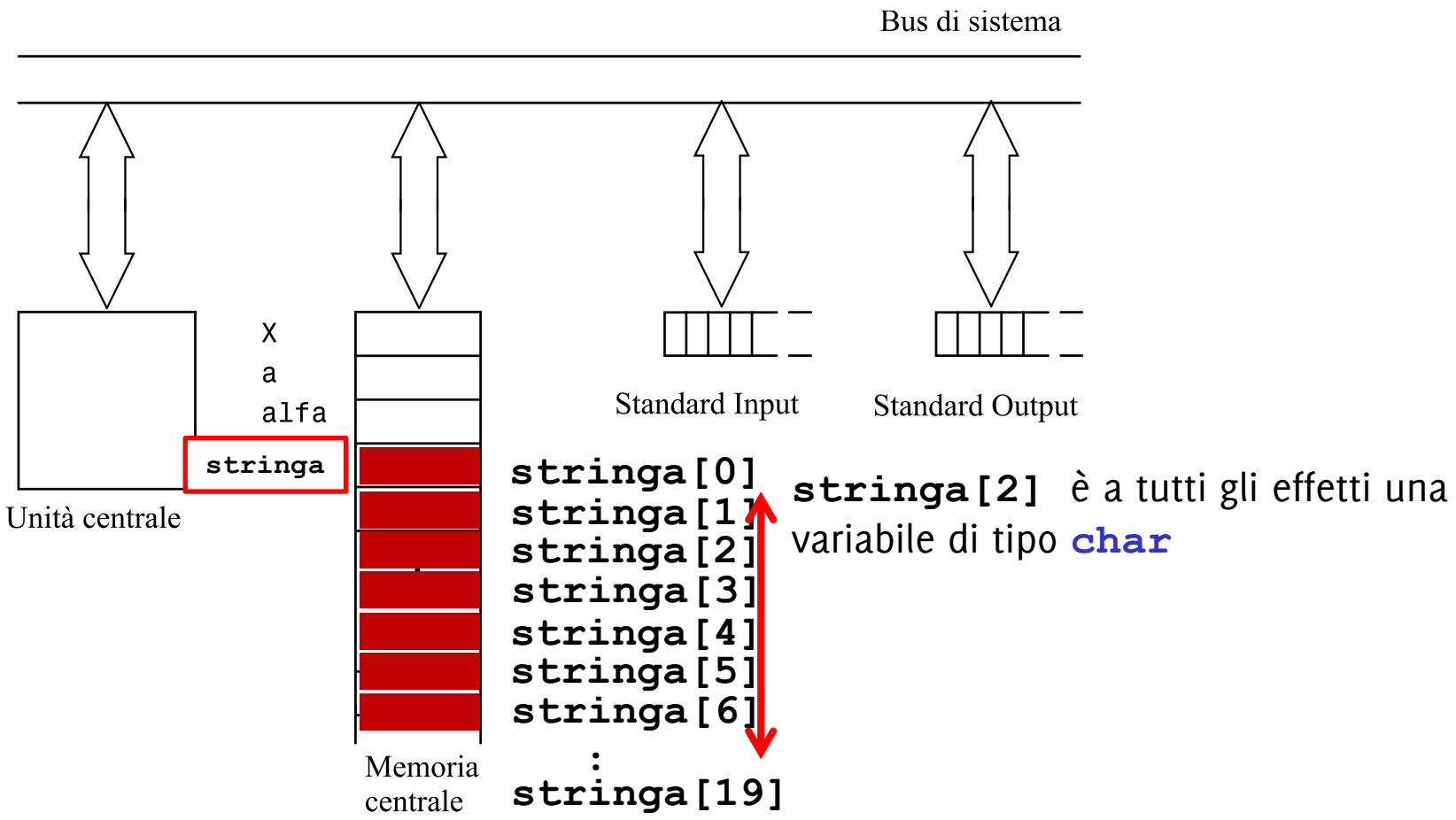
Dato il frequente utilizzo ci sono standard e comandi particolari per facilitare l'uso delle stringhe,

- I/O
- Calcolo lunghezza
- Confronto e Copia

NB NON esiste il tipo predefinito "string" né altri simili

Lo spazio allocato per gli array

```
char stringa[20];
```



stringa[2] è a tutti gli effetti una variabile di tipo **char**

Acquisizione e stampa di stringhe

Esempio

Acquisizione e stampa di una stringa elemento per elemento

Esempio

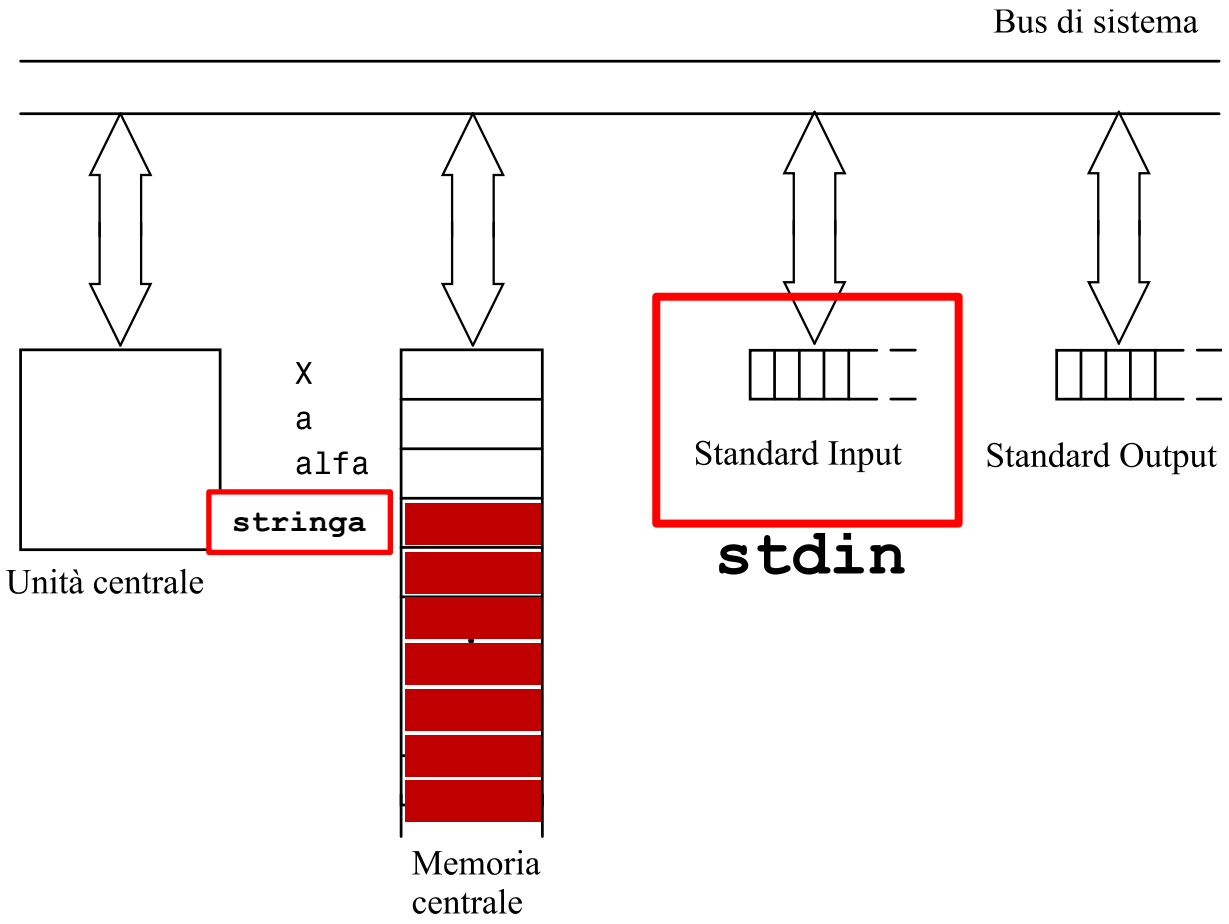
Acquisizione e stampa di una stringa elemento per elemento

```
int main()
{
int i;
char s[10];
printf("\ninserire carattere 1");
scanf("%c", &s[0]); scanf("%*c");
for (i = 1; i < 10; i++)
     {printf("\ninserire carattere %d",i+1);
     scanf("%c", &s[i]); scanf("%*c");
}
for(i = 0; i < 10; i++)
     printf("%c\n", s[i]);
return 0;
}
```

Oppure
fflush(stdin)

Lo spazio allocato per gli array

```
char stringa[20];
```



È necessario inserire **`scanf("%*c");`** dopo le acquisizioni di caratteri perché altrimenti, le **`scanf("%c",...);`** successive, finirebbero per acquisire quello che rimane in **`stdin`**, in questo caso il carattere invio per chiudere la scanf precedente..

Il problema non si pone con i cicli di interi, perché «invio» non è un intero

Esempio

Acquisizione e stampa di una stringa elemento per elemento

```
int main()
{
int i;
char s[10];
printf("\ninserire carattere 1");
scanf("%c", &s[0]); scanf("%*c");
for (i = 1; i < 10; i++)
     {printf("\ninserire carattere %d",i+1);
     scanf("%c", &s[i]); scanf("%*c");
}
for(i = 0; i < 10; i++)
     printf("%c\n", s[i]);
return 0;
}
```

E' necessario inserirlo anche qua, altrimenti si acquisisce solo l'invio (pulisco il carattere inserito)

Esempio

Acquisizione e stampa di una stringa elemento per elemento

Andava bene
anche così
ovviamente

```
int main()
{
int i;
char s[10];
for (i = 0; i < 10; i++)
     {printf("\ninserire carattere %d",i+1);
     scanf("%c", &s[i]); scanf("%*c");
}
for(i = 0; i < 10; i++)
     printf("%c\n", s[i]);
return 0;
}
```

Esempio

Acquisizione e stampa di una stringa elemento per elemento

```
int main()
{
int i;
char s[10];
for (i = 0; i < 10; i++)
     {printf("\ninserire carattere %d",i+1);
     scanf(" %c", &s[i]);
}
for(i = 0; i < 10; i++)
     printf("%c\n", s[i]);
return 0;
}
```

Oppure così

Acquisizione stringa: `scanf` e `gets`

È possibile evitare di inserire una stringa carattere per carattere grazie alle funzioni **`scanf`** e **`gets`**

```
char str[10]

scanf("%s", str);

gets(str);
```

Nota bene:

`str` è già un indirizzo… è l'indiriso del primo elemento **`&str[0]`**,
quindi **`scanf`** non ha bisogno della **`&`**.

`gets` richiede la libreria **`string`**, quindi occorre inserire **`#include <string.h>`**

Attenzione che se in input si inserisce una stringa troppo lunga, essa è memorizzata oltre lo spazio riservato all'array !!!

Errore grave che può causare crash a runtime!!!

Acquisizione stringa: `scanf` e `gets`

Sia **`scanf("%s",…)`** che **`gets(…)`** delimitano la **parte significativa** (i caratteri inseriti dall'utente) con il **carattere speciale** **`'\0'`** (con codifica ASCII = 0).

Se a **`gets(luogo);`** digito: `Milano` in memoria verrà scritto **`"Milano\0"`.**

Differenze

- **`scanf("%s",luogo);`** **termina** l'inserimento **al primo spazio o invio.**
- **`gets(luogo);`** **termina** l'inserimento al primo **invio**

Ecco cosa acquisiscono se digito: `Piazza san Babila`

- **`scanf("%s",luogo);`** **`"Piazza\0"`**
- **`gets(luogo);`** **`"Piazza san Babila\0"`**

Le stringhe

Array di caratteri: spesso chiamati stringhe

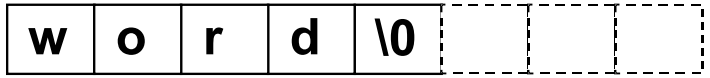
- Quando rappresentano “caratteri da leggersi in fila”

Dichiarazione+inizializzazione di una stringa:

```
char stringa[] = "word";
```

Il carattere nullo '\0' termina le stringhe

Perciò l’array stringa ha 5 elementi (non 4):



Dichiarazione equivalente:

```
char stringa[] = {'w', 'o', 'r', 'd', '\0'};
```

Stampa di Stringhe

È possibile stampare i caratteri in una stringa **fino al terminatore** utilizzando **`printf("%s",…);`**

Esempio

```
gets(luogo);
printf("Io abito a %s", luogo);
```

Quando si popola la stringa carattere per carattere, è necessario inserire il terminatore di stringa `'\0'`

```
char provincia[3];
provincia[0] = 'M';
provincia[1] = 'I';
provincia[2] = '\0';
printf("Io abito a %s", provincia);
```

Occorre tener conto del terminatore anche nella dimensione

Una domanda

Qual è la differenza tra `'x'` e `"x"`?

- `'x'` costante carattere (rappresentata da un intero)
- `"x"` costante stringa, rappresentata da un array che contiene due caratteri: 'x' e '\0'

ATTENZIONE

Le stringhe **non** sono propriamente un **tipo** di dato.
Non sono un tipo base, **ma array di caratteri**

Calcolo della lunghezza di una stringa

Calcolo della Lunghezza

È possibile calcolare la lunghezza di una stringa andando a contare gli elementi che precedono '\0' .

```
int len = 0;
char luogo[100];
scanf("%s", luogo);
while(luogo[len] != '\0')
    len++;
printf("%s e' lunga %d", luogo, len);
```

La lunghezza di una stringa corrisponde alla posizione del carattere '\0'. Il valore viene assegnato a **len**

Oppure è possibile usare la funzione **strlen**, contenuta nella libreria string

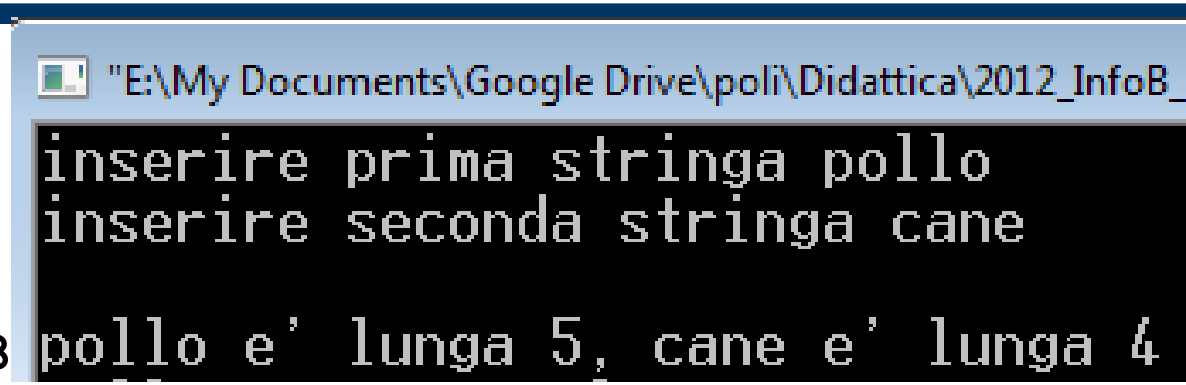
```
len = strlen(luogo);
```

Esempio (Calcolo Lunghezza di Una Stringa)

```
#include<string.h>
int main()
{
	int coincidono,len1,len2,flag;
	char str1[30], str2[30], str3[30];
	printf("inserire prima stringa ");
	gets(str1);
	printf("inserire seconda stringa ");
	gets(str2);

	// calcolo le lunghezze
	len1 = strlen(str1);
	// calcolo le lunghezze
	len2 = strlen(str2);

	printf("\n%s e' lunga %d, %s e' lunga %d", str1, len1, str2, len2);
	return 0;
}
```

Esempio (Calcolo Lunghezza di Una Stringa)

```
#include<string.h>
int main()
{
        int coincidono,len1,len2,flag;
        char str1[30], str2[30], str3[3
        printf("inserire prima stringa ");
        gets(str1);
        printf("inserire seconda stringa ");
        gets(str2);

        // calcolo le lunghezze
        len1 = strlen(str1);
        // calcolo le lunghezze
        len2 = strlen(str2);

        printf("\n%s e' lunga %d, %s e' lunga %d", str1, len1, str2, len2);
        return 0;
}
```



Confronto tra stringhe

Confronto tra Stringhe

È possibile verificare se due stringhe coincidono:

1. Verificando che la loro lunghezza coincide **&&**
2. Verificando se esse coincidono in ogni elemento

Confronto tra Stringhe

È possibile verificare se due stringhe coincidono:

1. Verificando che la loro lunghezza coincide **&&**
2. Verificando se esse coincidono in ogni elemento

```
int flag = 1; int len,i;
char str1[30], str2[30];
gets(str1); gets(str2);
len = strlen(str1);
if(len == strlen(str2))
    for(i = 0; i < len && flag; i++)
          { if(str1[i] != str2[i])
                flag = 0; }
else // non hanno la stessa lunghezza
    flag = 0;
printf("%s == %s : %d", str1, str2, flag);
```

È indispensabile mettere le parentesi attorno a questo if altrimenti l'else seguente viene associato a questo e non a quello più esterno

Confronto tra Stringhe

Oppure è possibile usare la funzione **strcmp**, contenuta nella libreria **string**. Sintassi

```
strcmp(s1 , s2);
```

Diventa un intero

- == 0 se coincidono
- < 0 se **s1** precede **s2** in ordine alfabetico
- > 0 se **s1** segue **s2** in ordine alfabetico

```
int cmpr;
cmpr = strcmp(str1 , str2);
if (cmpr == 0)
    printf("%s e %s coincidono", str1,str2);
```

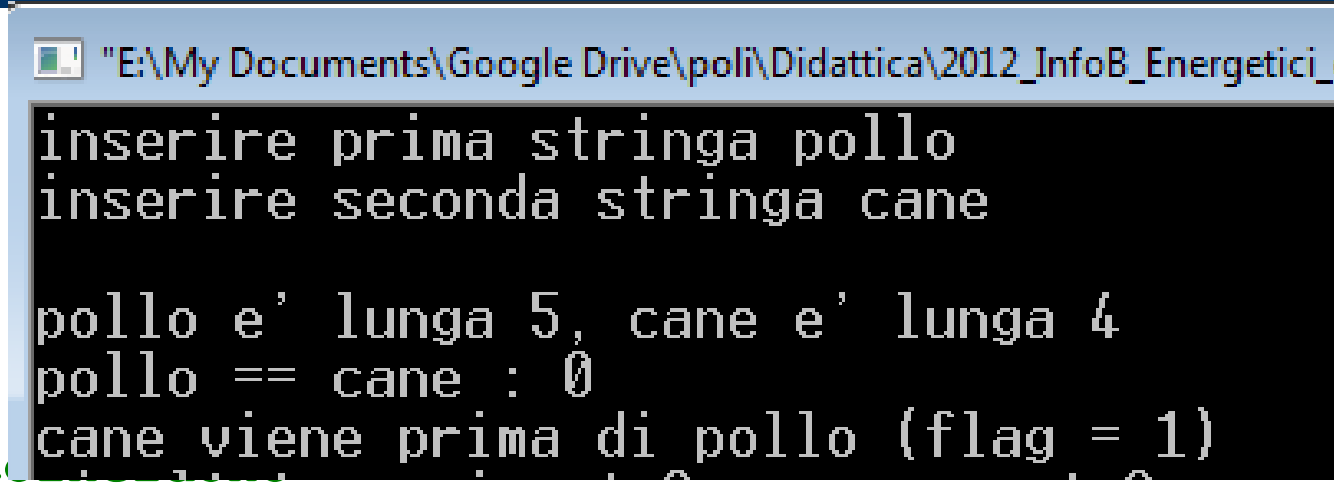
NB. Le stringhe **str1** e **str2** devono terminare con **'\0'**

Esempio di Confronto Tra Stringhe

```
#include<string.h>
void main()
{
int coincidono, len1, len2, flag;
char str1[30], str2[30], str3[30];
…
// strcmp che restituisce 0 se coincidono
flag = strcmp(str1 , str2);
// metto coincidono a 1 quando flag è 0
coincidono = (flag == 0);
printf("\n%s == %s : %d", str1, str2, coincidono);
if (flag > 0)
 printf("\n%s precede%s(flag = %d)",str2, str1,flag);
if(flag < 0)
 printf("\n%s precede%s(flag = %d)",str1, str1,flag);
}
```

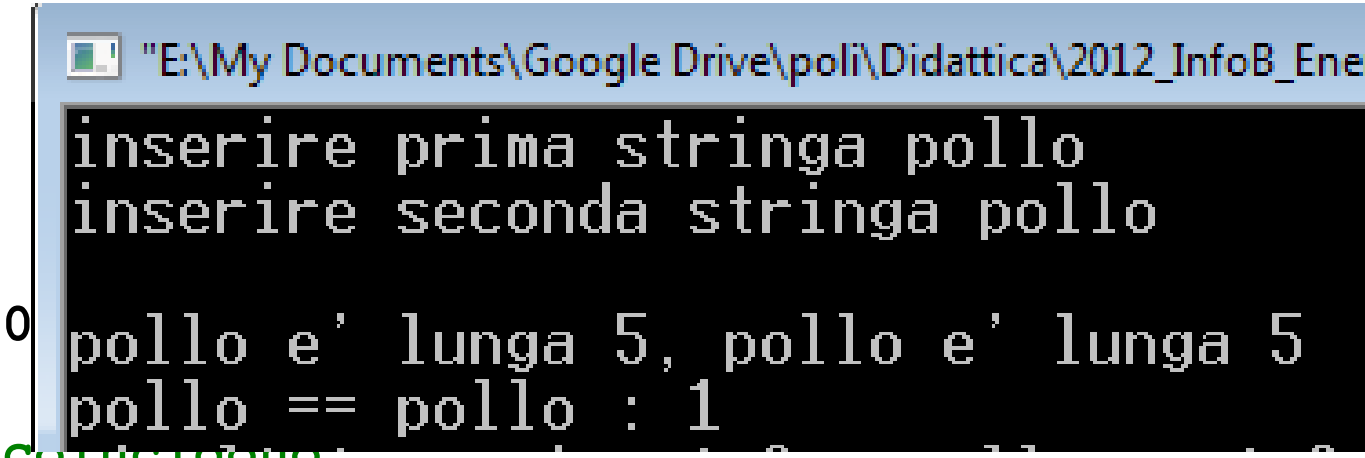
Esempio

```
#include<string.h>
void main()
{
int coincidono,len1,len2,flag;
char str1[30], str2[30], str3[30
…
// strcmp che restituisce 0 se c
flag = strcmp(str1 , str2);
// metto coincidono a 1 quando flag è 0
coincidono = (flag == 0);
printf("\n%s == %s : %d", str1, str2, coincidono);
if (flag > 0)
 printf("\n%s precede%s(flag = %d)",str2, str1,flag);
if(flag < 0)
 printf("\n%s precede%s(flag = %d)",str1, str1,flag);
}
```



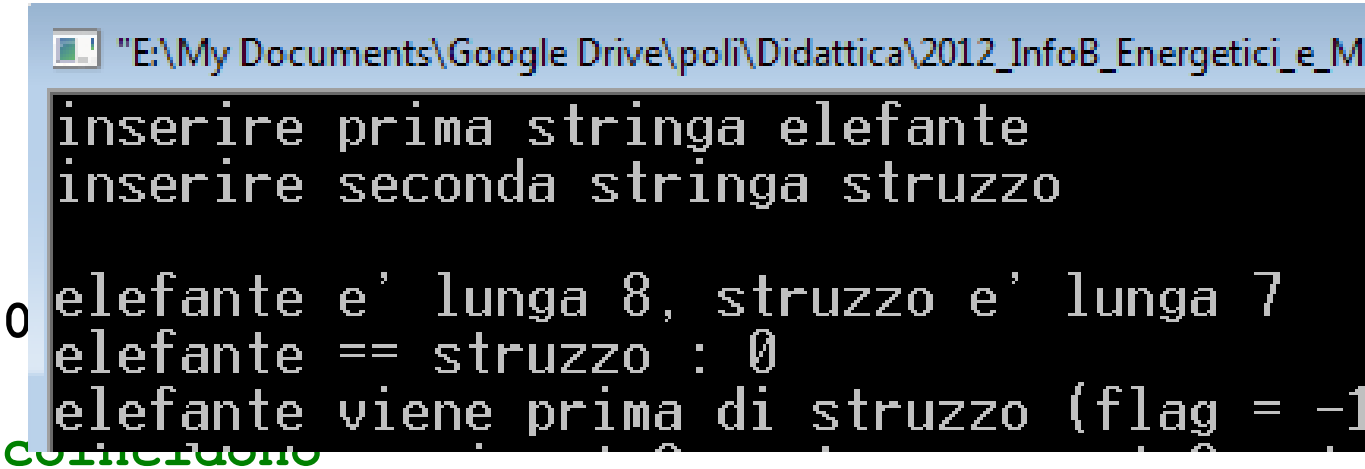
Esempio

```
#include<string.h>
void main()
{
int coincidono,len1,len2,flag;
char str1[30], str2[30], str3[30
…
// strcmp che restituisce 0 se coincidono
flag = strcmp(str1 , str2);
// metto coincidono a 1 quando flag è 0
coincidono = (flag == 0);
printf("\n%s == %s : %d", str1, str2, coincidono);
if (flag > 0)
 printf("\n%s precede%s(flag = %d)",str2, str1,flag);
if(flag < 0)
 printf("\n%s precede%s(flag = %d)",str1, str1,flag);
}
```



Esempio

```
#include<string.h>
void main()
{
int coincidono,len1,len2,flag;
char str1[30], str2[30], str3[30
…
// strcmp che restituisce 0 se coincidono
flag = strcmp(str1 , str2);
// metto coincidono a 1 quando flag è 0
coincidono = (flag == 0);
printf("\n%s == %s : %d", str1, str2, coincidono);
if (flag > 0)
 printf("\n%s precede%s(flag = %d)",str2, str1,flag);
if(flag < 0)
 printf("\n%s precede%s(flag = %d)",str1, str1,flag);
}
```



Copia tra stringhe

Copia tra stringhe

Assegnamo sempre un carattere alla volta

```
char s1[N], s2[M];
/* assegnamento di s2, omesso */
int i = 0;
while( i <= strlen(s2) && i < N ) {
      s1[i] = s2[i];
      ++i;
}
```

N.B. funziona correttamente se s2 è una stringa ben formata (terminata da '\0') e s1 è sufficientemente grande (N >= strlen(s2))

strcpy(s1, s2)

Assegnamo sempre un carattere alla volta

```
char s1[N], s2[M];
/* assegnamento di s2, omesso */
int i = 0,dim;
dim=strlen(s2);
while( i <= dim && i < N ) {
      s1[i] = s2[i];
      ++i;
}
```

Evita di calcolare la lunghezza di s2
in ogni iterazione

strcpy(s1, s2)

Assegnamo sempre un carattere alla volta

```
char s1[N], s2[M];
/* assegnamento di s2, omesso */
int i = 0,dim;
dim=strlen(s2);
while( i <= dim && i < N ) {
      s1[i] = s2[i];
      ++i;
}
s1[i]='\0';
```

N.B. occorre formattare correttamente s1, aggiungendo il terminatore di stringa

QUIZ

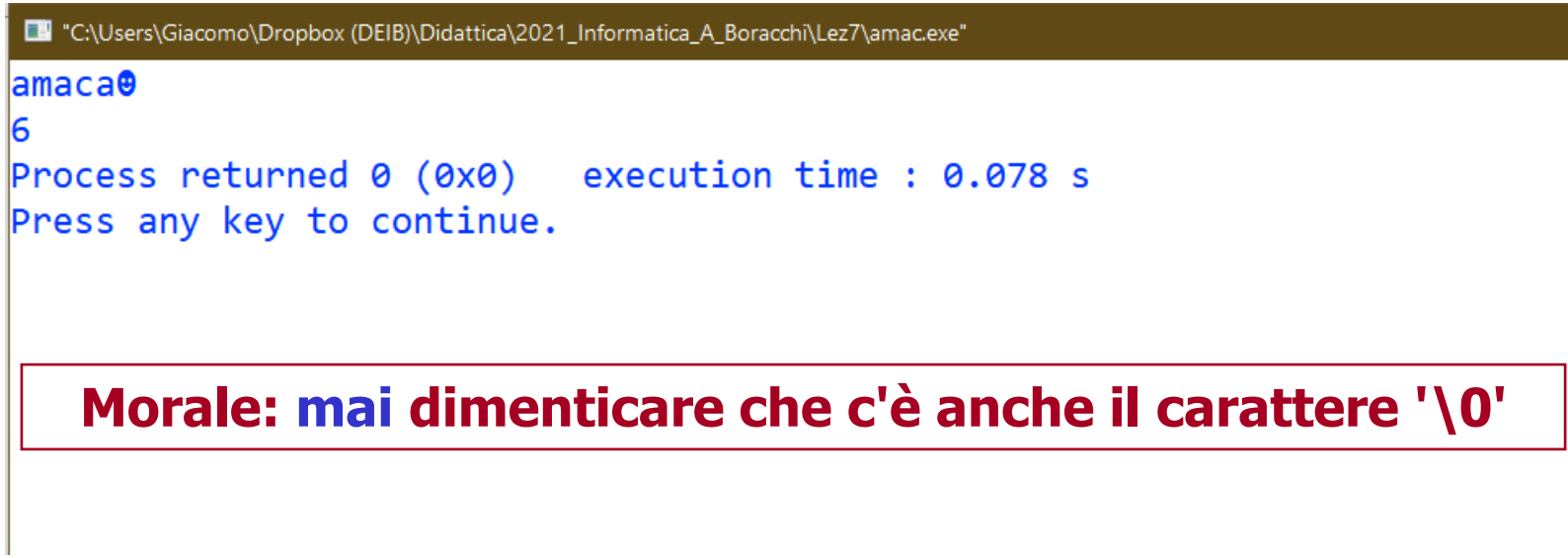
Che cosa stampano le seguenti printf()?

```c
#include<stdio.h>
#define N 10
int main()
{
int a=2;
char amac[] = "amac";

amac[strlen(amac)] = amac[2];

printf("%s", amac);
printf("%d", strlen(amac));
return 0;
}
```

QUIZ

Che cosa stampano le seguenti printf()?

```c
#include<stdio.h>
#define N 10
int main()
{
int a=2;
char amac[] = "amac";

amac[strlen(amac)] = amac[2];

printf("%s", amac);
printf("%d", strlen(amac));
return 0;
}
```



Morale: mai dimenticare che c'è anche il carattere '\0'

QUIZ

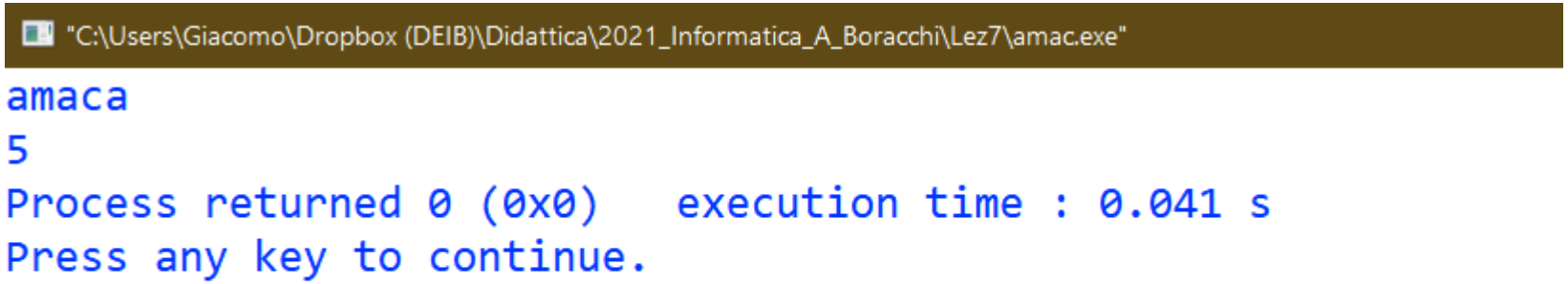
Che cosa stampano le seguenti printf()?

```
#include<stdio.h>
#define N 10
int main()
{
int a=2, len;
char amac[6] = "amac";
len = strlen(amac);
amac[len] = amac[2];
amac[len + 1] = '\0';

printf("%s\n", amac);
printf("%d", strlen(amac));
return 0;}
```

QUIZ

Che cosa stampano le seguenti printf()?

```c
#include<stdio.h>
#define N 10
int main()
{
int a=2, len;
char amac[6] = "amac";
len = strlen(amac);
amac[len] = amac[2];
amac[len + 1] = '\0';

printf("%s\n", amac);
printf("%d", strlen(amac));
return 0;}
```



Copia tra Stringhe

È possibile eseguire la copia elemento per elemento da un array ad un altro, come nell'esercizio precedente

Oppure è possibile usare la funzione **`strcpy`**, contenuta nella libreria **string**. Sintassi:

```
strcpy(s1 , s2);
```

Copia il contenuto di **`s2`** in **`s1`** incluso il **`'\0'`**

Per accodate le stringhe si usa la funzione **`strcat`**, contenuta nella libreria **string**.

Sintassi:

```
strcat(s1 , s2);
```

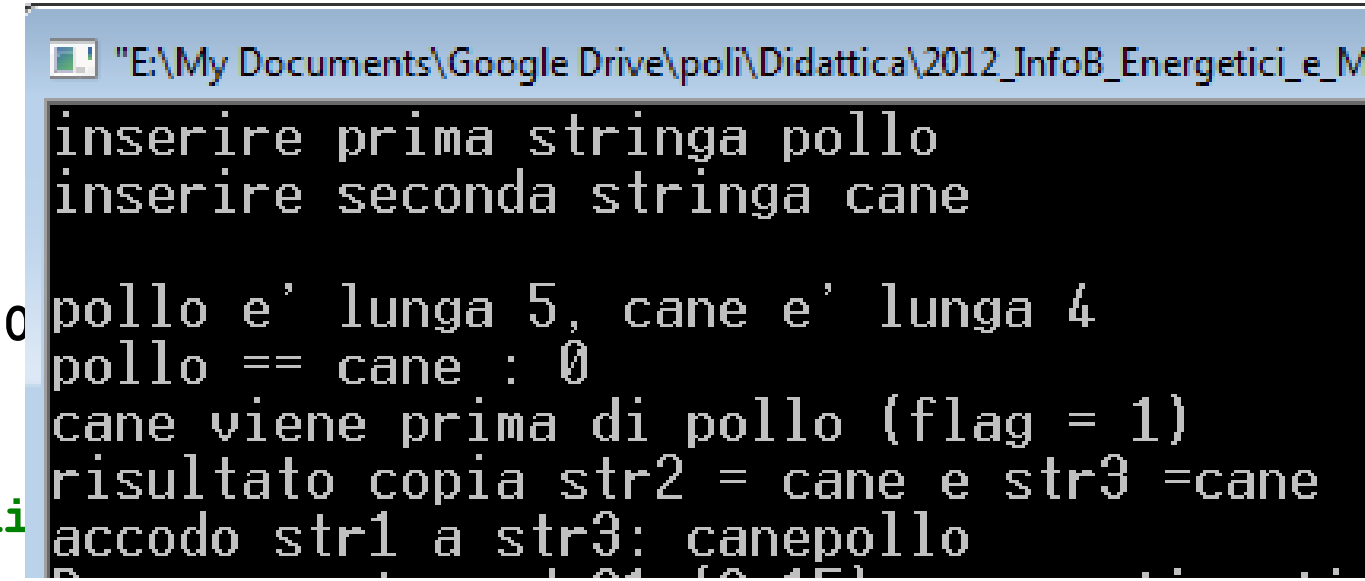
Accoda il di **`s2`** in **`s1`** (il **`'\0'`** appare solo alla fine)

Esempio di Copia Tra Stringhe

```
#include<string.h>
void main()
{
int coincidono,len1,len2,flag;
char str1[30], str2[30], str3[30];
…

// copia in str3 il contenuto di str2
strcpy(str3,str2);
printf("\nrisultato copia str2 = %s e str3 =%s", str2, str3);

// accoda
strcat(str3,str1);
printf("\naccodo str1 a str3: %s ", str3);
}
```

Esempio

```
#include<string.h>
void main()
{
int coincidono,len1,len2,flag;
char str1[30], str2[30], str3[30
…

// copia in str3 il contenuto di
strcpy(str3,str2);
printf("\nrisultato copia str2 = %s e str3 =%s", str2, str3);

// accoda
strcat(str3,str1);
printf("\naccodo str1 a str3: %s ", str3);
}
```



```c
#include<string.h>

char str1[32]; /* str1 ha spazio per 32 char. */
char str2[64]; /* str2 ha spazio per 64 char. */

/* inizializza str1 con la stringa "alfa" */
strcpy(str1, "alfa");    /* str1 contiene "alfa"  */

/* copia str1 in str2 */
strcpy(str2, str1);      /* str2 contiene "alfa"  */

/* lunghezza di str1 */
x = strlen(str1);        /* x assume valore 4     */

/* scrivi str1 su standard output */
printf("%s", str1);      /* scrive str1 su stdout */

/* leggi str1 da standard input */
scanf("%s", str1);       /* str1 “riceve” da stdin */
```

```c
char str1[32];
char str2[64];

scanf("%s", str1);
> ciao⍰ /* ora str1 contiene "ciao" */

strcpy(str2, str1);  /* str2 riceve "ciao"*/
val = strlen(str2);  /* val  = 4          */

printf("%s\n", str2);
> ciao  /* stampa "ciao" */
```

Attenzione: `strlen("")` vale `0`

Esercizio

Scrivere un programma che acquisisce una stringa **`s1`** e copia in una seconda stringa **`s2`** solo le vocali contenute in **`s1`**

il programma quindi stampa **`s2`**

`Esempio:`

`ciao -> iao`

Esercizio

Scrivere un programma che traduce una stringa inserita dall'utente in alfabeto farfallino

Esempio:

ciao -> cifiafaofo

```c
#include<string.h>
#include<stdio.h>
#define L 30
int main()
{
char s[L], f[3*L];
int i, j = 0, n;
printf("inserire testo da tradurre: ");
scanf("%s", s);
n = strlen(s);
for(i = 0; i<=n; i++)
{
    f[j] = s[i]; j++;
    if (s[i] == 'a' || s[i] == 'e' || s[i] == 'i' || s[i] == 'o' || s[i] == 'u')
    {
        f[j] = 'f'; j++;
        f[j] = s[i]; j++;
    }
}
printf("%s -> %s", s,f);
}
```