086180 - INFORMATICA APPLICATA
A.A. 2011-12
2° semestre

M07a – stringhe

Un particolare array

- Un array può essere costruito su ogni tipo , semplice o composto.
- Quindi si possono costruire array di char
  - `char  nome[20];`
- In C gli array di char vengono denominati stringhe ed hanno alcune caratteristiche particolari

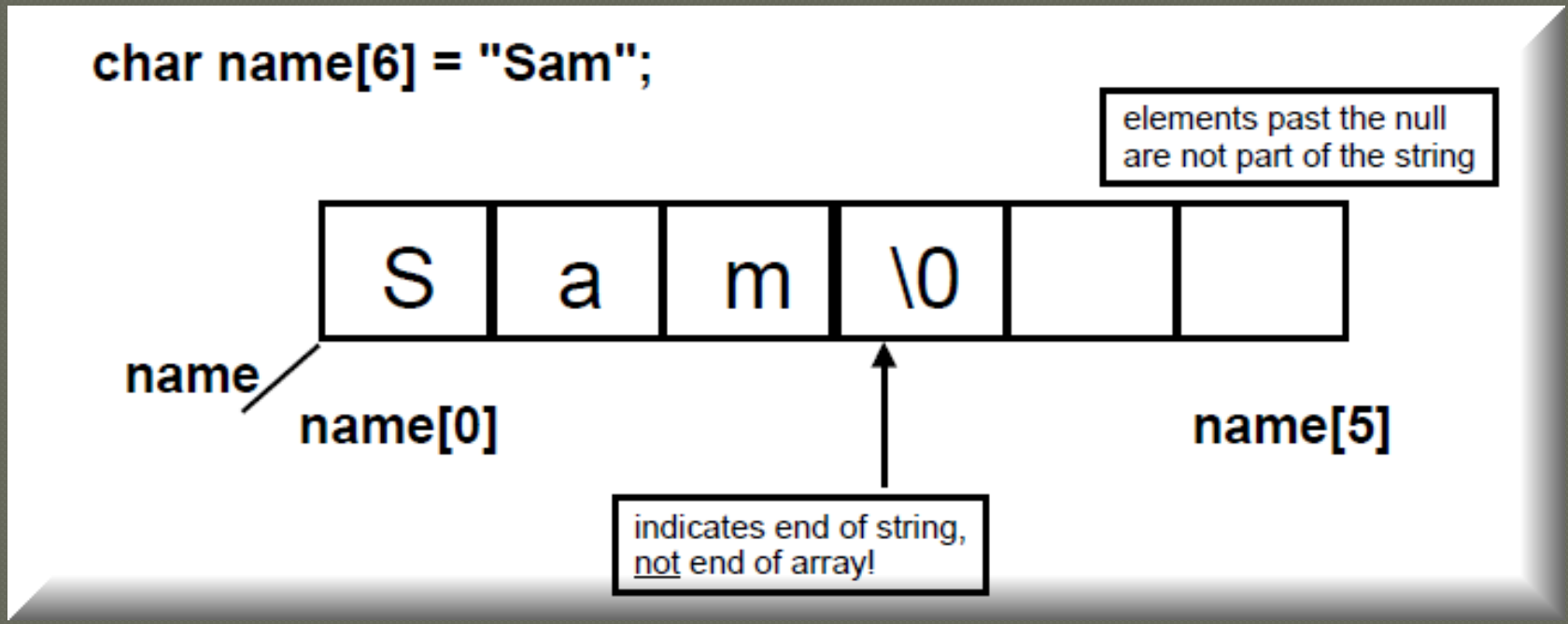
Le stringhe

- Una array di char è un ibrido ed ha alcune caratteristiche peculiari

- Può essere inizializzato con un valore stringa
- Ovvero con una serie di caratteri tra apici doppi
  - char cognome[ ] = { "Rosi" };

- Se i caratteri della stringa sono n la rappresentazione in memoria è di n+1 caratteri
- Il carattere in più è un \0

La rappresentazione di una stringa



Arrays as Strings

- **Initialize**

  can do each char separately or as a string constant

  char name[6] = {'C', 'h', 'u', 'c', 'k', '\0'};

  initialization list of individual chars
  Note null character at end - required for a string!

  char name[6] = "Sam";

  double quotes implies a null character at the end

What would be the result from each? How do they differ?

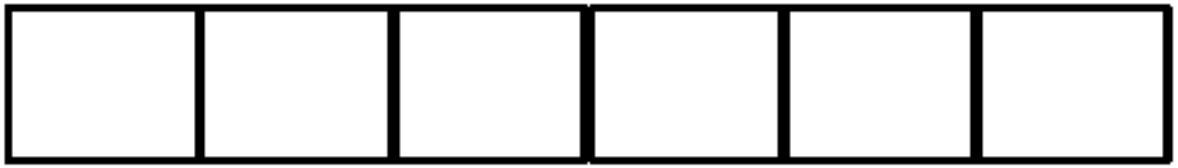
Le stringhe

```c
#include <stdio.h>
int main()
{int i, lungnome, lungcognome;
   char  nome[] = {'A','l','d','o'};
    /* la stringa è di 4 elementi  */
    char cognome[] = { "Rosi"   };
    /* la stringa è di 5 elementi  */

    lungnome=sizeof(nome)/sizeof(nome[1]);
    printf("\n\n dimensioni di nome  %d",lungnome);

    lungcognome=sizeof(cognome)/sizeof(cognome[1]);
    printf("\n\n dimensioni di cosgnome  %d",lungcognome);

   for(i=0;i<lungnome ;i++ )
   {printf("\n  %3d %c",nome[i], nome[i]);
   } ;
   printf("\n\n");
    for(i=0;i<lungcognome ;i++ )
   {printf("\n  %3d %c",cognome[i], cognome[i]);
   } ;
   printf("\n\n");
    return 0;
}
```

```
"C:\Programmi\C-Free Standard\temp\l

 dimensioni di nome  4

 dimensioni di cosgnome  5
   65 A
  108 l
  100 d
  111 o


   82 R
  111 o
  115 s
  105 i
    0

Premere un tasto per continuare .
```

Input di stringhe da tastiera

one char at a time or as a string



name

```
for(ctr = 0; name[ctr] != '\n'; ++ctr)
    name[ctr] = getchar( );
```

reads each character until the \n is encountered; you must insert the null at the end!

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

reads string from keyboard; converts the \n to a null, but will stop at first 'whitespace' character!

```
gets(name);
```

reads string from keyboard, including any whitespace; converts \n to null

Multidimensionali

```
char name[3][8] =  {  {'S', 'a', 'm', '\0'},
                      {'C', 'h', 'u', 'c', 'k', '\0'},
                      {'M', 'a', 'r', 'k', '\0'}  };
```

--- or ---

```
char name[3][8] = {"Sam", "Chuck", "Mark"};
```

ATTENZIONE

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.

Copia di una stringa

- Come per tutti gli array, anche per le stringhe come
  - `char a[ ] , b[ ]`
- il seguente assegnamento **non** effettua una copia
  - `a = b;`
- • È necessario eseguire la copia elemento per elemento
  - `for (i = 0; i<DIM; i++) a[i] = b[i];`

#include <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

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];
strcpy(destinazione, sorgente);
printf("Il valore copiato è: %s",
                              destinazione);
return 0;
}
```

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

Copia di stringhe

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

Concatenamento strcat

- Accoda il contenuto della stringa secondo parametro nella primo parametro
  - `int main () {`
  - `char prima[10]="ciao";`
  - `char seconda[10]="mamma";`
  - `strcat(prima, seconda);`
  - `printf("Il valore risultante è: %s",`
  - `prima);`
  - `return 0;`
  - `}`
- • Devo essere certo che la dimensione massima della stringa destinazione sia sufficiente ad ospitare la nuova stringa

Concatenamento

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

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;
}
```

Lunghezza di una stringa 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

Lunghezza di una stringa

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

Confronto tra stringhe strcmp

- Confronta il contenuto di due stringhe
- • Restituisce:
  - 0 se le stringhe sono identiche
  - <0 se la stringa passata come primo parametro è minore della stringa passata come secondo parametro
  - >0 se la stringa passata come primo parametro è maggiore della stringa passata come secondo parametro
- 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'

Confronto

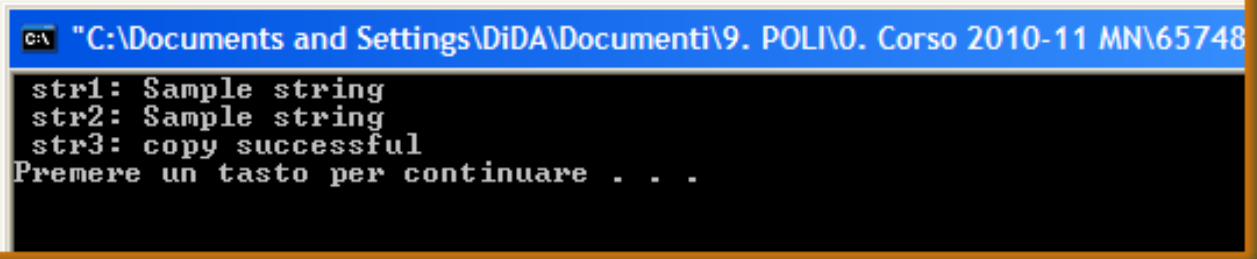
- char S1[10]="bologna";
- char S2[10]="napoli";
- int k;
- …
- k=strcmp(S1,S2); /* k < 0 */
- k=strcmp(S1,S1); /* k=0*/
- k=strcmp(S2,S1); /* k>0 */

```
 1: #include  <stdio.h>
 2: #include  <string.h>
 3:
 4: int main ()
 5:      {
 6: char s1[10]="mamma";
 7: char s2[10]="ciao";
 8: int c;
 9:      c = strcmp(s1, s2);
10:      if (c==0) {
11:      printf("\nle due stringhe sono uguali\n");
12:      } else if (c<0) {
13:      printf("\ns1 < s2\n");
14:      } else {
15:      printf("\ns1 > s2\n");
16:      }
17: return 0;
18: }
```

Scrittura di una stringa sullo schermo

strcpyEXAMPLE.c

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

int main()
{  /* ESEMPIO  strcpy  */

  char str1[]="Sample string";
  char str2[40];
  char str3[40];

  strcpy (str2,str1);
  strcpy (str3,"copy successful");

  printf (" str1: %s\n str2: %s\n str3: %s\n",str1,str2,str3);
  return 0;
}
```



Uso delle funzioni <string.h>

cpyEXAMPLE.c | **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]="seconda";

    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("E1)\t%s\n",strchr (stri1, 'o'));
    printf("E2)\t%s\n",strchr (stri1, 'k'));

    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;
}
```