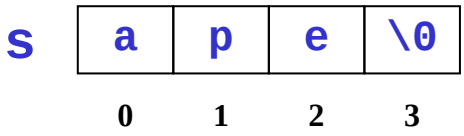
STRINGHE: ARRAY DI CARATTERI

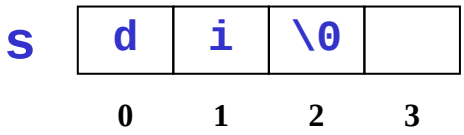
- Una *stringa di caratteri in C* è un array di caratteri *terminato dal carattere* `'\0'`



- Un vettore di N caratteri può dunque ospitare stringhe *lunghe al più N-1 caratteri,* perché una cella è destinata al terminatore `'\0'`.

STRINGHE: ARRAY DI CARATTERI

- Un array di N caratteri può ben essere usato per memorizzare *stringhe più corte*



- In questo caso, *le celle oltre la k-esima* (k essendo la lunghezza della stringa) *sono concettualmente vuote:* praticamente sono inutilizzate e contengono un valore casuale.

STRINGHE

- Una stringa si può *inizializzare,* come ogni altro array, elencando le singole componenti:

```
char s[4] = {'a', 'p', 'e', '\0'};
```

oppure anche, più brevemente, *con la forma compatta* seguente:

```
char s[4] = "ape" ;
```

Il carattere di terminazione '\0' è *automaticamente incluso* in fondo. *Attenzione alla lunghezza!*

STRINGHE: LETTURA E SCRITTURA

- Una stringa si può *leggere da tastiera e stampare,* come ogni altro array, elencando le singole componenti:

```
...char str[4]; int i;
  for (i=0; i < 3; i++)
  scanf("%c", &str[i]); str[4] = "\0" ...
```

- oppure anche, più brevemente, *con la forma compatta* seguente:

```
...char str[4]; scanf("%s", str);
```

NOTA: E' un'eccezione !!! Gli array non si possono leggere e scrivere interamente, ma elemento per elemento.

ESEMPIO

Problema:

Date due stringhe di caratteri, decidere quale precede l'altra in ordine alfabetico.

Rappresentazione dell'informazione:

- poiché vi possono essere *tre* risultati (s1<s2, s1==s2, s2<s1), *un boolean non basta*
- possiamo usare:
  - due boolean (**`uguale`** e **`precede`**)
  - tre boolean (**`uguale`**, **`s1precedes2`**, **`s2precedes1`**)
  - un intero (negativo, zero, positivo)

scegliamo la terza via.

ESEMPIO

Specifica:

- scandire uno a uno gli elementi *di egual posizione* delle due stringhe, *o fino alla fine delle stringhe, o fino a che se ne trovano due diversi*
  - *nel primo caso, le stringhe sono uguali*
  - *nel secondo, sono diverse*
- nel secondo caso, confrontare i due caratteri così trovati, e determinare qual è il minore
  - la stringa a cui appartiene tale carattere precede l'altra

ESEMPIO

Codifica:

```
main() {
 char s1[] = ”Maria";
 char s2[] = ”Marta";
 int i=0, stato;
 while(s1[i]!='\0' && s2[i]!='\0' &&
       s1[i]==s2[i]) i++;
 stato = s1[i]-s2[i];
 ……
 }
```

negativo ↔ s1 precede s2
positivo ↔ s2 precede s1
zero ↔ s1 è uguale a s2

ESEMPIO

Problema:

Data una stringa di caratteri, copiarla in un altro array di caratteri (di lunghezza non inferiore).

Ipotesi:

La stringa è “ben formata”, ossia correttamente terminata dal carattere ‘\0’.

Specifica:

- scandire la stringa elemento per elemento, fino a trovare il terminatore ‘\0’ (che esiste certamente)
- *nel fare ciò, copiare l’elemento nella posizione corrispondente dell’altro array.*

ESEMPIO

Codifica:copia della stringa carattere per carattere

```
main() {
 char s[] = "Nel mezzo del cammin di";
 char s2[40];
 int i=0;
 for (i=0; s[i]!='\0'; i++)
     s2[i] = s[i];
 s2[i] = '\0';
}
```

La dimensione deve essere tale da garantire che la stringa non ecceda

Al termine, occorre garantire che anche la nuova stringa sia "ben formata", inserendo esplicitamente il terminatore.

ESEMPIO

Perché non fare così?

```
main() {
 char s[] = "Nel mezzo del cammin di";
 char s2[40];
     s2 = s;
}
```

ERRORE DI COMPILAZIONE:
incompatible types in assignment !!

PERCHÉ GLI ARRAY NON POSSONO ESSERE MANIPOLATI NELLA LORO INTEREZZA !

ESEMPIO

Problema:

Data una stringa di caratteri, *scrivere una funzione* che ne calcoli la lunghezza.

Nel caso delle stringhe, la dimensione non serve perché può essere dedotta dalla posizione dello '\0'

Codifica:

```
int lunghezza(char s[]) {
  int lung=0;
  for (lung=0; s[lung]!='\0'; lung++);
  return lung;
}
```

LIBRERIA SULLE STRINGHE

Il C fornisce una nutrita libreria di funzioni per operare sulle stringhe:

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

Include funzioni per:

- copiare una stringa in un'altra (**strcpy**)
- concatenare due stringhe (**strcat**)
- confrontare due stringhe (**strcmp**)
- cercare un carattere in una stringa (**strchr**)
- cercare una stringa in un'altra (**strstr**)
- ...