LE STRINGHE

Il linguaggio C

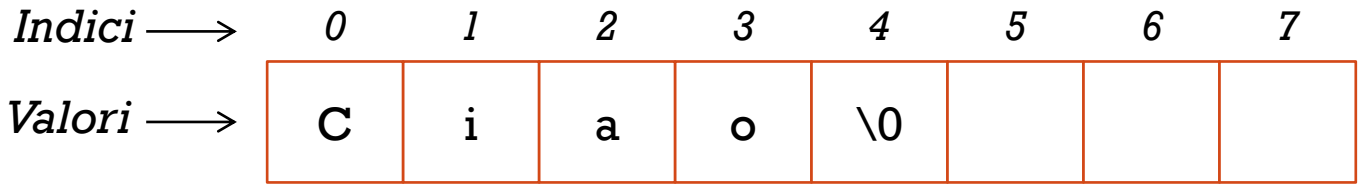
1

DEFINIZIONE DI VETTORE

- Un vettore può essere definito come una "collezione organizzata di oggetti":
  - **"collezione":** implica che gli oggetti contenuti nel vettore siano dello stesso tipo;
  - **"organizzata":** implica che sia possibile identificare univocamente tutti gli oggetti del vettore.

LE STRINGHE IN C

- In C non esiste un tipo primitivo per la gestione delle stringe, ma sopperisce a questa mancanza utilizzando i vettori di caratteri.
- Una stringa è un vettore di tipo "***char***" a cui è aggiunto, come ultimo elemento, un **carattere terminatore** rappresentato dalla sequenza '*\0*'.



COME SI DICHIARA UNA STRINGA?

- La dichiarazione di una stringa è identica a quella di un vettore e si compone di 3 elementi:
  1. il tipo di dati: "***char***";
  2. il nome del vettore (come per le variabili);
  3. il numero di elementi (dimensione) del vettore che deve essere almeno pari alla lunghezza della stringa più lunga che voglio poter memorizzare incrementata di uno (a causa del carattere terminatore).

char stringa[51];

Può contenere una stringa di al più 50 caratteri (inclusi spazi, segni di punteggiatura, ecc.)

INIZIALIZZAZIONE DI UNA STRINGA

- Come per tutti gli altri tipi di vettori, anche le stringhe, all'atto della dichiarazione, **non vengono inizializzate**, a meno che non lo si faccia in modo esplicito.
- L'inizializzazione di una stringa può avvenire in molti modi, i due principali sono:
  - contemporaneamente alla dichiarazione della stessa;
  - utilizzando le funzioni messe a disposizione dalle librerie standard del C, soprattutto dalla libreria "*string.h*".

INIZIALIZZAZIONE DI UNA STRINGA

- Per dichiarare e contemporaneamente inizializzare una stringa si deve procedere come negli esempi seguenti:

```
char str[8] = {'C', 'i', 'a', 'o', '\0'};
```

```
char str[] = {'C', 'i', 'a', 'o', '\0'};
```

```
char str[] = "Ciao";
```

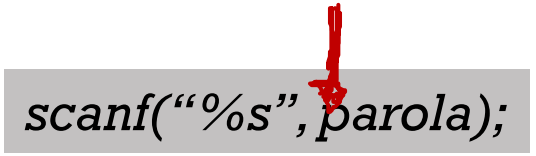

INIZIALIZZAZIONE DI UNA STRINGA

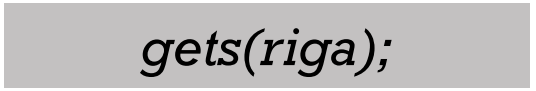
- Utilizzando le funzioni messe a disposizione dalle librerie standard del C, soprattutto dalla libreria "*string.h*".
- La funzione più utilizzata a questo scopo è la funzione ***strcpy(char *str1, char *str2)*** che permette di copiare il contenuto di "*str2*" in "*str1*".

```
strcpy(str,"Ciao");
```


INPUT DI STRINGHE

- Le due principali funzioni, entrambe appartenenti alla libreria "*stdio.h*", utilizzate per acquisire una stringa in input sono la:
  - "*scanf*", utilizzando il segnaposto *"%s"*, che legge una parola;
  - "*gets*" che legge una riga e la memorizza nella stringa passata come parametro.





OUTPUT DI STRINGHE

- Le due principali funzioni, entrambe appartenenti alla libreria "*stdio.h*", utilizzate per eseguire l'output di una stringa sono la:
  - "*printf*", utilizzando il segnaposto "*%s*";
  - "*puts*" che visualizza la stringa passata come parametro.

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

```
puts(riga);
```

LIBRERIA «STRING.H»

- Utilizzando le funzioni messe a disposizione dalla libreria "string.h" si possono manipolare le stringhe in modo semplice e rapido, ecco le principali:
  - *char *strcpy(char *str1, char *str2):* copia il contenuto di "*str2*" in "*str1*";
  - *char *strncpy(char *str1, char *str2, unsigned int n):* copia al più n caratteri di "*str2*" in "*str1*";
  - *char *strcat(char *str1, char *str2):* concatena *"str2"* al termine di "*str1*";
  - *char *strncat(char *str1, char *str2, unsigned int n):* concatena al più n caratteri di *"str2"* al termine di "*str1*";
  - *unsigned int strlen(char *str)*: restituisce il numero di caratteri contenuti nella stringa "*str*". Non considera nel conteggio il terminatore della stringa ('*\0*');

LIBRERIA «STRING.H»

- Utilizzando le funzioni messe a disposizione dalla libreria "string.h" si possono manipolare le stringhe in modo semplice e rapido, ecco le principali:
  - *int *strcmp(char *str1, char *str2):* confronta il contenuto di "*str1*" con quello di "*str2*" e restituisce:
    - un valore minore di zero (<0) se "*str1*" precede in ordine alfabetico "*str2*";
    - zero (0): se "*str1*" è uguale a "*str2*";
    - un valore maggiore di zero (>0) se "*str1*" segue in ordine alfabetico "*str2*".
  - *int *strncmp(char *str1, char *str2, unsigned int n):* confronta al più n caratteri di "*str1*" con quelli di "*str2*" e restituisce un valore di ritorno utilizzando le stesse regole della "*strcmp*".

ESEMPIO

- Leggi due parole, lunghe al più 20 caratteri, e:
  1. visualizzale in ordine alfabetico;
  2. ordinale e visualizzale in ordine crescente rispetto alla loro lunghezza;
  3. concatenale inserendo uno spazio tra loro.

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

int main(int argc, char *argv[])
{
        char parola_1[42], char parola_2[21];

        printf("Inserisci la prima parola: ");
        scanf("%s", parola_1);
        printf("Inserisci la seconda parola: ");
        scanf("%s", parola_2);
```


ESEMPIO

```
if(strcmp(parola_1, parola_2) < 0)
{
    printf("\n%s\n%s\n", parola_1, parola_2);
}
else
{
    printf("\n%s\n%s\n", parola_2, parola_1);
}

if(strlen(parola_1) < strlen(parola_2))
{
    printf("\n%s\n%s\n", parola_1, parola_2);
}
else
{
    printf("\n%s\n%s\n\n", parola_2, parola_1);
}
```

ESEMPIO

```
        strcat(parola_1, " ");
        strcat(parola_1,parola_2);
        puts(parola_1);

        return 0;
}
```

ESERCIZI

- **Esercizio 1:** scrivere un programma che legga una parola di non più di 30 caratteri e visualizzi il numero di vocali in essa contenute.
- **Esercizio 2:** scrivere un programma che legga due parole di non più di 30 caratteri e verifichi se la seconda parola sia o meno contenuta nella prima.
- **Esercizio 3:** scrivere un programma che legga una stringa di non più di 200 caratteri (la stringa contiene una frase e quindi una o più parole) e la visualizzi andando a capo dopo ogni parola.
- **Esercizio 4:** scrivere un programma che legga 10 stringhe di non più di 100 caratteri e visualizzi la più lunga tra esse.