

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

```
#define MAXPAROLA 30
#define MAXRIGA 80
```

```
int main(int argc, char *argv[])
{
    int freq[MAXPAROLA]; /* vettore di contatori
delle frequenze delle lunghezze delle parole */
    char riga[MAXRIGA];
    int i, inizio, lunghezza;
    FILE *f;
```

```
for(i=0; i<MAXPAROLA; i++)
    freq[i]=0;
```

```
if(argc != 2)
```

```
{
    fprintf(stderr, "ERRORE, serve un parametro con il nome del file\n");
    exit(1);
}
```

```
f = fopen(argv[1], "r");
if(f==NULL)
```

```
{
    fprintf(stderr, "ERRORE, impossibile aprire il file %s\n", argv[1]);
    exit(1);
}
```

```
while( fgets( riga, MAXRIGA, f ) != NULL )
```



Caratteri e stringhe

Vettori di caratteri

Vettori di caratteri

- Il tipo stringa
- Terminatore nullo
- Input/output di stringhe

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

```
#define MAXPAROLA 30
#define MAXRIGA 80
```

```
int main(int argc, char *argv[])
{
    int freq[MAXPAROLA]; /* vettore di contatori
delle frequenze delle lunghezze delle parole */
    char riga[MAXRIGA];
    int i, inizio, lunghezza;
    FILE *f;
```

```
    for(i=0; i<MAXPAROLA; i++)
        freq[i]=0;
```

```
    if(argc != 2)
    {
        fprintf(stderr, "ERRORE, serve un parametro con il nome del file\n");
        exit(1);
    }
```

```
    f = fopen(argv[1], "r");
    if(f==NULL)
    {
        fprintf(stderr, "ERRORE, impossibile aprire il file %s\n", argv[1]);
        exit(1);
    }
```

```
    while( fgets( riga, MAXRIGA, f ) != NULL )
```



Vettori di caratteri

Il tipo stringa

Stringhe in C

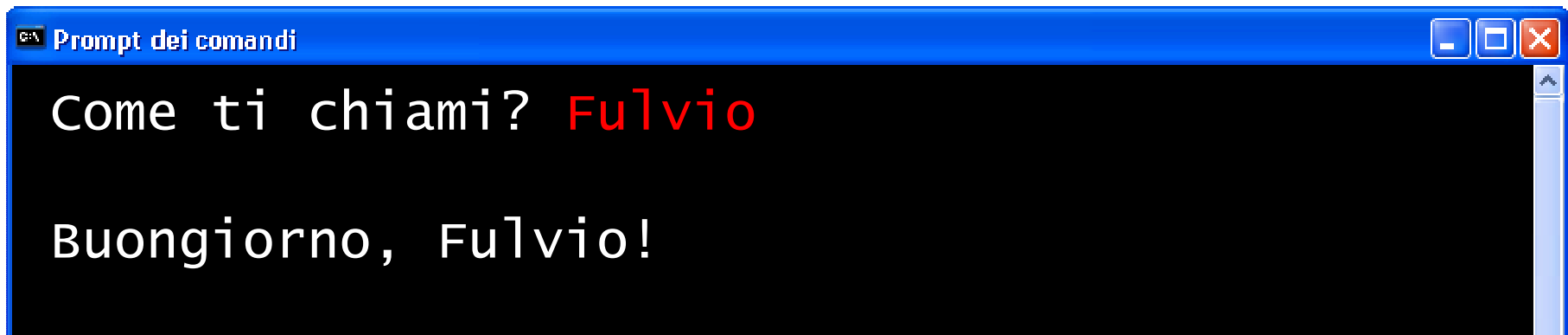
- Nel linguaggio C non è supportato esplicitamente alcun tipo di dato "stringa"
- Le informazioni di tipo stringa vengono memorizzate ed elaborate ricorrendo a semplici **vettori di caratteri**

```
char saluto[10] ;
```

B	u	o	n	g	i	o	r	n	o
---	---	---	---	---	---	---	---	---	---

Esempio

- Si realizzi un programma in linguaggio C che acquisisca da tastiera il nome dell'utente (una stringa di max 20 caratteri), e stampi a video un saluto per l'utente stesso



```
Prompt dei comandi
Come ti chiami? Fulvio
Buongiorno, Fulvio!
```

Soluzione (1/3)



saluti.c

```
const int MAX = 20 ;  
char nome[MAX] ;  
int N ;  
char ch ;  
int i ;  
  
printf("Come ti chiami? ") ;  
  
N = 0 ;
```

Soluzione (2/3)



saluti.c

```
ch = getchar() ;

while( ch != '\n' && N<MAX )
{
    nome[N] = ch ;
    N++ ;
    ch = getchar() ;
}
```

Soluzione (3/3)



saluti.c

```
printf("Buongiorno, ") ;  
  
for(i=0; i<N; i++)  
    putchar( nome[i] ) ;  
  
printf("!\n") ;
```


Commenti (1/2)

- Qualsiasi operazione sulle stringhe si può realizzare agendo opportunamente su vettori di caratteri, gestiti con occupazione variabile
- Così facendo, però vi sono alcuni svantaggi
 - Per ogni vettore di caratteri, occorre definire un'opportuna variabile che ne indichi la lunghezza
 - Ogni operazione, anche elementare, richiede l'uso di cicli `for/while`

- Alcune convenzioni ci possono aiutare
 - Gestire in modo standard i vettori di caratteri usati per memorizzare stringhe
 - Apprendere le tecniche solitamente utilizzate per compiere le operazioni più frequenti
- Molte funzioni di libreria seguono queste convenzioni
 - Conoscere le funzioni di libreria ed utilizzarle per accelerare la scrittura del programma

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

```
#define MAXPAROLA 30
#define MAXRIGA 80
```

```
int main(int argc, char *argv[])
{
    int freq[MAXPAROLA]; /* vettore di contatori
delle frequenze delle lunghezze delle parole */
    char riga[MAXRIGA];
    int i, inizio, lunghezza;
    FILE *f;
```

```
    for(i=0; i<MAXPAROLA; i++)
        freq[i]=0;
```

```
    if(argc != 2)
    {
        fprintf(stderr, "ERRORE, serve un parametro con il nome del file\n");
        exit(1);
    }
```

```
    f = fopen(argv[1], "r");
    if(f==NULL)
    {
        fprintf(stderr, "ERRORE, impossibile aprire il file %s\n", argv[1]);
        exit(1);
    }
```

```
    while( fgets( riga, MAXRIGA, f ) != NULL )
```



Vettori di caratteri

Terminatore nullo

Lunghezza di una stringa

- Vi sono due tecniche per determinare la lunghezza di una stringa
 1. utilizzare una variabile intera che memorizzi il numero di caratteri validi

```
char nome[10] ;  
int lungh_nome ;
```

F u l v i o z ! \$.

6

Lunghezza di una stringa

- Vi sono due tecniche per determinare la lunghezza di una stringa
 1. utilizzare una variabile intera che memorizzi il numero di caratteri validi

```
char nome[10] ;  
int lungh_nome ;
```

F u l v i o z ! \$.

6

2. utilizzare un carattere "speciale", con funzione di **terminatore**, dopo l'ultimo carattere valido

```
char nome[10] ;
```

F u l v i o Ø ! \$.

Carattere terminatore

- Il carattere “terminatore” deve avere le seguenti caratteristiche
 - Fare parte della tabella dei codici ASCII
 - Deve essere rappresentabile in un char
 - Non comparire mai nelle stringhe utilizzate dal programma
 - Non deve confondersi con i caratteri “normali”
- Inoltre il vettore di caratteri deve avere una posizione libera in più, per memorizzare il terminatore stesso

Terminatore standard in C

- Per convenzione, in C si sceglie che tutte le stringhe siano rappresentate mediante un carattere terminatore
- Il terminatore corrisponde al carattere di codice ASCII pari a zero
 - `nome[6] = 0 ;`
 - `nome[6] = '\0' ;`

F	u	l	v	i	o	Ø	!	\$.
---	---	---	---	---	---	---	---	----	---

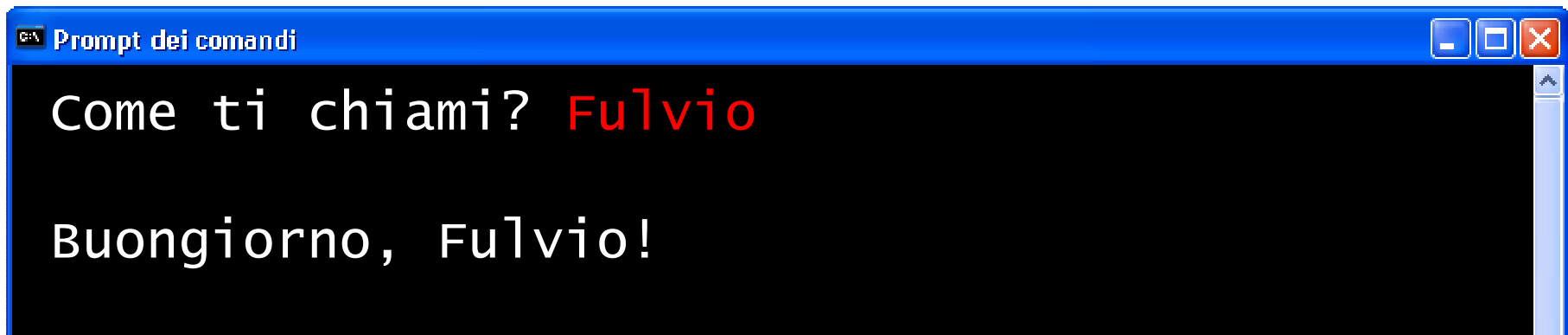
Vantaggi

- Non è necessaria un'ulteriore variabile intera per ciascuna stringa
- L'informazione sulla lunghezza della stringa è interna al vettore stesso
- Tutte le funzioni della libreria standard C rispettano questa convenzione
 - Si aspettano che la stringa sia terminata
 - Restituiscono sempre stringhe terminate

- Necessario 1 byte in più
 - Per una stringa di N caratteri, serve un vettore di N+1 elementi
- Necessario ricordare di aggiungere sempre il terminatore
- Impossibile rappresentare stringhe contenenti il carattere ASCII 0

Esempio

- Si realizzi un programma in linguaggio C che acquisisca da tastiera il nome dell'utente (una stringa di max 20 caratteri), e stampi a video un saluto per l'utente stesso



```
Prompt dei comandi
Come ti chiami? Fu1vio
Buongiorno, Fu1vio!
```

Soluzione (1/3)



saluti0.c

```
const int MAX = 20 ;  
char nome[MAX+1] ;  
char ch ;  
int i ;  
  
printf("Come ti chiami? ") ;  
  
i = 0 ;
```

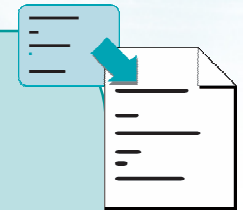
Soluzione (2/3)

```
i = 0 ;  
ch = getchar() ;  
while( ch != '\n' && i<MAX )  
{  
    nome[i] = ch ;  
    i++ ;  
    ch = getchar() ;  
}  
/* aggiunge terminatore nullo */  
nome[i] = '\0' ;
```



saluti0.c

Soluzione (3/3)



saluti0.c

```
printf("Buongiorno, ") ;  
  
for(i=0; nome[i]!='\0'; i++)  
    putchar( nome[i] ) ;  
  
printf("!\n") ;
```

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

```
#define MAXPAROLA 30
#define MAXRIGA 80
```

```
int main(int argc, char *argv[])
{
    int freq[MAXPAROLA]; /* vettore di contatori
delle frequenze delle lunghezze delle parole */
    char riga[MAXRIGA];
    int i, inizio, lunghezza;
    FILE *f;
```

```
for(i=0; i<MAXPAROLA; i++)
    freq[i]=0;
```

```
if(argc != 2)
{
    fprintf(stderr, "ERRORE, serve un parametro con il nome del file\n");
    exit(1);
}
```

```
f = fopen(argv[1], "r");
if(f==NULL)
{
    fprintf(stderr, "ERRORE, impossibile aprire il file %s\n", argv[1]);
    exit(1);
}
```

```
while( fgets( riga, MAXRIGA, f ) != NULL )
```



Vettori di caratteri

Input/output di stringhe

I/O di stringhe

- Diamo per scontato di utilizzare la convenzione del terminatore nullo
- Si possono utilizzare
 - Funzioni di lettura e scrittura carattere per carattere
 - Come nell'esercizio precedente
 - Funzioni di lettura e scrittura di stringhe intere
 - scanf e printf
 - gets e puts

Lettura di stringhe con scanf

- Utilizzare la funzione `scanf` con lo specificatore di formato **"%s"**
- La variabile da leggere deve essere il nome di un vettore di caratteri
 - Non utilizzare le parentesi quadre
 - Non utilizzare la `&`
- Legge ciò che viene immesso da tastiera, fino al primo spazio o fine linea (esclusi)
 - Non adatta a leggere nomi composti (es. "Pier Paolo")

Esempio

```
const int MAX = 20 ;  
char nome[MAX+1] ;  
  
printf("Come ti chiami? ") ;  
  
scanf("%s", nome) ;
```

Lettura di stringhe con gets

- La funzione `gets` è pensata appositamente per acquisire una stringa
- Accetta un parametro, che corrisponde al nome di un vettore di caratteri
 - Non utilizzare le parentesi quadre
- Legge ciò che viene immesso da tastiera, fino al fine linea (escluso), e compresi eventuali spazi
 - Possibile leggere nomi composti (es. "Pier Paolo")

Esempio

```
const int MAX = 20 ;  
char nome[MAX+1] ;  
  
printf("Come ti chiami? ") ;  
  
gets(nome) ;
```

Scrittura di stringhe con printf

- Utilizzare la funzione `printf` con lo specificatore di formato **"%s"**
- La variabile da stampare deve essere il nome di un vettore di caratteri
 - Non utilizzare le parentesi quadre
- È possibile combinare la stringa con altre variabili nella stessa istruzione

Esempio

```
printf("Buongiorno, ") ;  
printf("%s", nome) ;  
printf("!\n") ;
```

```
printf("Buongiorno, %s!\n", nome) ;
```

Scrittura di stringhe con puts

- La funzione `puts` è pensata appositamente per stampare una stringa
- La variabile da stampare deve essere il nome di un vettore di caratteri
 - Non utilizzare le parentesi quadre
- Va a capo automaticamente
 - Non è possibile stampare altre informazioni sulla stessa riga

Esempio

```
printf("Buongiorno, ") ;  
puts(nome) ;  
  
/* No!! printf("!\n") ; */
```

Conclusione

- Utilizzare sempre la convenzione del terminatore nullo
- Ricordare di allocare un elemento in più nei vettori di caratteri
- Utilizzare quando possibile le funzioni di libreria predefinite
 - In lettura, prediligere `gets`
 - In scrittura
 - `printf` è indicata per messaggi composti
 - `puts` è più semplice se si ha un dato per riga