



Stringhe in C

Alessandra Giordani

agiordani@disi.unitn.it

Lunedì 30 maggio 2011

<http://disi.unitn.it/~agiordani/>



Stringhe

- Un altro tipo di insieme che vorremmo poter rappresentare è quello delle stringhe di caratteri, cioè sequenze di caratteri, per poter comporre parole e frasi (ad esempio, per visualizzare messaggi sullo schermo).
- In C la rappresentazione delle stringhe è un argomento complesso. Per il momento limitiamoci ad considerare una stringa costante in C come una sequenza di caratteri compresi tra doppi apici ("). Ad esempio:
 - "ciao mondo!"
 - "questa è una stringa"
 - "per favore, inserisci un numero"



Stringhe: Sequenze di Escape

- Alcuni caratteri non sono rappresentabili con un singolo carattere (es. carattere di fine riga)
- Delle sequenze speciali (escape sequences) ci consentono di inserire questi caratteri.
- Ad esempio, il carattere di fine riga può essere rappresentato con `'\n'` (non `"\n"`!)
- Quindi, la stringa `"questa è una stringa\n"` termina con un carattere di fine riga.
- In ogni caso una stringa termina sempre con il carattere di fine stringa `'\0'` (non `"\0n"`!)

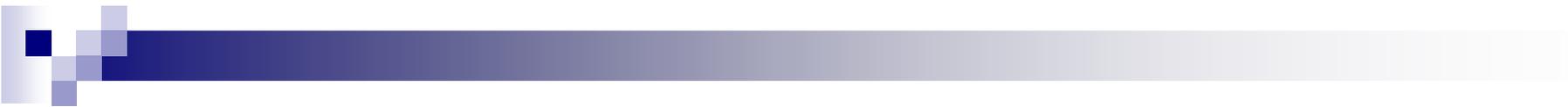
Le funzioni di libreria per le stringhe

`strlen()`

- La funzione `strlen()`, restituisce il numero di caratteri che compongono una stringa (escluso il carattere nullo)
- Poiché nell'espressione `*str++` i due operatori hanno la stessa precedenza ed associatività destra, l'espressione viene analizzata dal compilatore nel modo seguente:
 - Valutazione dell'operatore di incremento postfisso; il compilatore passa `str` all'operatore successivo e lo incrementa solo al termine della valutazione dell'espressione
 - Valutazione dell'operatore `*`, applicato a `str`
 - Completamento dell'espressione, con l'incremento di `str`

```
int strlen(str)
char *str;
{
    int i;

    for (i=0; *str++; i++)
        ; /* istruzione vuota */
    return i;
}
```



printf() di stringhe

- printf può anche essere usata per stampare il contenuto di una variabile un array di caratteri o parte di esso
- All'interno della stringa template dobbiamo usare %s.
- Come altro parametro dobbiamo passare il nome della variabile, che usato da solo è il puntatore al primo elemento della stringa
- printf() stamperà quel elemento fino al carattere '\0'
 - `char str[] = "stringa";`
 - `printf("%s", str); // stampa stringa`
 - `printf("%s", str+2); // stampa ringa`
 - `printf("%s", str[2]); // non corretto!`
 - `printf("%c", str[2]); // stampa r`



scanf() di stringhe

- Allo stesso modo per leggere una stringa non carattere per carattere ma per intero
 - template “%s”
 - indirizzo del primo elemento della stringa che è proprio il nome della stringa! (no &)
- Però in questo caso quando si incontra uno spazio la stringa di input si considera finita...



scanf() di stringhe con spazi

- Nel template possiamo specificare...
 - quanti caratteri leggere:
 - “%20s” leggere massimo 20 caratteri (no spazi)
 - quali carattere accettare
 - “%[A-Za-z]” accettare solo caratteri alfabetici maiuscolo o minuscolo e lo spazio
 - “%[0-9-]” accettare solo caratteri numerici e il -



Buffer di lettura - accorgimenti

- Ogni volta che si legge una stringa devo aver preallocata un buffer sufficiente
- Se leggo una stringa di massimo 20 char devo aver allocato un buffer di 21 char (e impedire che ne vengano letti di più!)

- `char string[21];`

- `scanf("%20s", string);`

- oppure

- `scanf("%20[a-b0-9]", string);`



Stringa MAIUSCOLA

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

main()
{
    char s[100], t[100];
    int i;

    printf("Inserisci una stringa: ");
    scanf("%[A-Za-z ]", s);
    for (i=0; i<strlen(s); i++)
    {
        if ((s[i] >= 'a') && (s[i] <= 'z'))
            t[i] = s[i] - 32;
        else
            t[i] = s[i];
    }
    t[i]='\0';
    printf("Stringa maiuscola: %s\n", t);
    exit(0);
}
```

Parole palindrome

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

```
main()
{
    char parola[32], i=0, n;

    printf("Inserisci una parola (lunga al max 31 caratteri): ");
    scanf("%31s", parola);
    n = strlen(parola);
    while((i <= n/2) && (parola[i] == parola[n-1-i]))
        i++;
    if(i > n/2)
        printf("La parola %s e' palindroma.\n", parola);
    else
        printf("La parola %s non e' palindroma.\n", parola);
    exit(0);
}
```

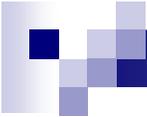
Es. ANNA,
ONORARONO, ...

Frase palindrome (a meno di spazi)

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

main()
{
    char parola[52], app[52], i = 0, j = 0, n;
    printf("Inserisci una frase (lunga al max 51 caratteri): ");
    scanf("%51[A-Za-z ']", parola);
    while(parola[i]!='\0')
    {
        if ((parola[i]!=' ') && (parola[i]!='\'))
        {
            app[j]=parola[i];
            j++;
        }
        i++;
    }
    app[j]='\0';
    n = j;
    i = 0;
    while((i <= n/2) && (app[i] == app[n-1-i]))
        i++;
    if(i > n/2)
        printf("La frase %s e' palindroma.\n", parola);
    else
        printf("La frase %s non e' palindroma.\n", parola);
    exit(0);
}
```

Es. i topi non avevano
nipoti, ai lati d italia...



Conversione da stringa ad intero

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

main()
{
    char anno_nascita[5], anno_corrente[5];
    int anni;

    printf("Inserire l'anno di nascita: ");
    scanf("%s", anno_nascita);
    printf("Inserire l'anno corrente: ");
    scanf("%s", anno_corrente);

    /* atoi() converte una stringa in un intero */
    anni = atoi(anno_corrente) - atoi(anno_nascita);
    printf("Eta': %d\n", anni);
    exit(0);
}
```