Stringhe in C

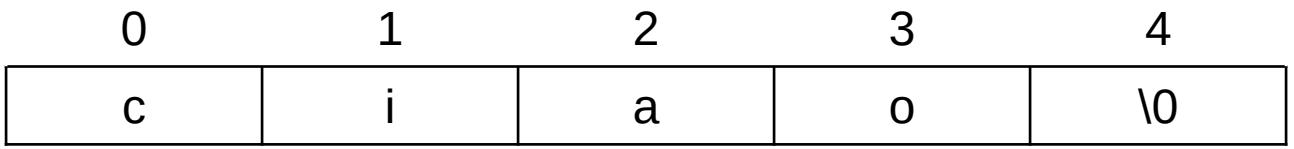
Emilio Di Giacomo

Stringhe

- Una stringa è una sequenza finita di caratteri
- Le stringhe sono un tipo di dati talmente importante e utile che fanno parte di quasi tutti i linguaggi di programmazione
- Nel linguaggio C una stringa è in realtà un array di caratteri
- Tutte le stringhe in C terminano con un carattere speciale detto carattere nullo (o terminatore): *'\0'*

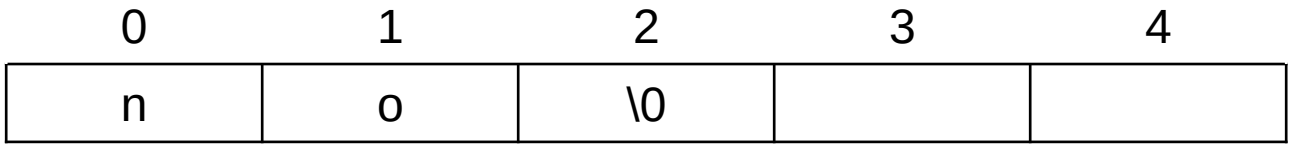
Stringhe

- Un array di *n* caratteri può dunque ospitare stringhe lunghe al più *n-1* caratteri, perché una cella è destinata al terminatore *'\0'*



Stringhe

- Un array di *n* caratteri può naturalmente essere usato per memorizzare stringhe più corte
- In questo caso, se *k* è la lunghezza della stringa, le celle con indice maggiore di *k* sono concettualmente vuote
  - praticamente sono inutilizzate e contengono un valore casuale.



Stringhe: inizializzazione

- Una stringa si può inizializzare, come ogni altro array, tramite un letterale array

  char s[] = {'c', 'i', 'a', 'o', '\0'};

  – In questo caso bisogna indicare anche *'\0'*

- Oppure anche, più brevemente, tramite un letterale stringa

  char s[] = "ciao";

  – In questo caso il carattere nullo *'\0'* è automaticamente incluso in fondo.

Leggere e stampare stringhe

- E' possibile leggere una stringa da tastiera (memorizzandola in un array di *char*) utilizzando la *scanf* e lo specificatore di conversione *%s*

```
char str[20];

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

- Nota: il nome dell'array va passato alla *scanf* senza farlo precedere da *&*
  - l'operatore *&* è utilizzato per indicare alla *scanf* la locazione di memoria in cui memorizzare il valore
  - *str* è il nome di un array e quindi è in realtà l'indirizzo del suo primo elemento: *&* non è necessario

Leggere e stampare stringhe

- Analogamente un array di caratteri che rappresenti una stringa può essere stampato utilizzando la *printf* e lo specificatore di conversione *%s*

```
char str[20];

…

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

- Gli array di caratteri costituiscono un'eccezione
  - gli altri tipi di array non si possono leggere e scrivere interamente, ma devono essere letti/scritti elemento per elemento

Leggere e stampare stringhe

- La *scanf* legge la stringa fino ad un carattere di spazio
- È possibile indicare il numero di caratteri che si vuole leggere usando lo specificatore di conversione *%xs*, dove *x* è il numero di caratteri da leggere (escluso il terminatore)

```
char str[20];

scanf("%19s", str);
```

Leggere e stampare stringhe

- Poiché una stringa è un array di caratteri, si può anche leggere/scrivere elemento per elemento

```
char str[20];
for (int i=0; i < 19; i++)
        scanf("%c", &str[i]);
str[19] = '\0';
...
for (int i=0; str[i]!='\0'; i++)
        printf("%c", str[i]);
```

Altre funzioni di libreria per l'I/O

- Nella libreria stdio.h sono presenti altre funzioni per la lettura/scrittura di stringhe e caratteri
- *char *gets(char *s)*
  - Legge i caratteri dallo standard input e li memorizza nell'array *s* fino a che non incontra un carattere newline (*'\n'*) o un indicatore di fine del file. Il terminatore viene inserito automaticamente nell'array
- *int puts(const char *s)*
  - Visualizza la stringa *s* seguita da un carattere newline

Altre funzioni di libreria per l'I/O

- *int getchar()*
  - Legge e restitisce il prossimo carattere dallo standard input
- *int putchar(int c)*
  - Visualizza il carattere memorizzato in *c*

- Nota: *gets* legge una stringa fino al carattere newline, mentre *scanf* legge una stringa fino a che non incontra un carattere di spazio

Esempio: puts & getchar

```
#include <stdio.h>

int main() {
      char frase[100];
      char c;
      puts("immetti una frase");
      int i = 0;
      while((c=getchar()) != '\n'){
            frase[i++] = c;
      }
      frase[i]='\0';
      puts("La frase immessa e':");
      puts(frase);
}
```

Esempio: gets & putchar

```
#include <stdio.h>
int main() {
      char frase[100];
      printf("immetti una frase:\n");
      gets(frase);
      printf("La frase immessa e':\n");
      int i = 0;
      while(frase[i] != '\0'){
            putchar(frase[i]);
            i++;
      }
}
```

Esempi sulle stringhe

- Scrivere le seguenti funzioni.
  - *int lunghezza(char s[])*: restituisce la lunghezza della stringa *s*
  - *void copiaStringa(char s1[], char s2[])*: copia la stringa *s1* nella stringa *s2* (di lunghezza non minore di *s1*)
  - *int confronta(char s1[], char s2[])*: confronta *s1* ed *s2* dal punto di vista lessicografico; restituisce *0* se le due stringhe sono uguali, *-1* se *s2* segue *s1* nell'ordinamento lessicografico, *1* se *s2* precede *s1* nell'ordinamento lessicografico

lunghezza(...)

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

- Nota: non viene indicate la dimensione dell'array perché può essere dedotta dalla posizione dello *'\0'*

copiaStringa(...)

```
void copiaStringa(char s1[], char s2[]) {
     for(int i=0; s1[i]!='\0'; i++)
          s2[i] = s1[i];
     s2[i] = '\0';
}
```

- Nota: Occorre esplicitamente inserire il carattere nullo

confronta(...)

```
int confronta(char s1[], char s2[]){
   int confronto;
   int i=0;
   while(s1[i]!='\0' && s2[i]!='\0' && s1[i]==s2[i])
      i++;
   if(s1[i]<s2[i])
       confronto=-1;
   else if(s1[i]>s2[i])
       confronto=1;
   else
       confronto=0;
   return confronto;
}
```

Librerie per gestire caratt./stringhe

- Nella libreria standard del C esistono diverse funzioni per la gestione di caratteri e stringhe
- In particolare la libreria *ctype.h* contiene funzioni per gestire caratteri
- La libreria *string.h* invece contiene funzioni per gestire le stringhe
- Vedremo alcune delle funzioni di tali librerie

La libreria ctype.h

Funzione	Descrizione	Esempio
int isblank(int c)	Restituisce un valore positivo (vero) se c è un carattere blank (spazio o tab), 0 (falso) altrimenti	isblank('\t') è positivo
int isdigit(int c)	Restituisce un valore positivo (vero) se c è una cifra, 0 (falso) altrimenti	isdigit('3') è positivo
int isalpha(int c)	Restituisce un valore positivo (vero) se c è una lettera, 0 (falso) altrimenti	isalpha('Z') è positivo
int isalnum(int c)	Restituisce un valore positivo (vero) se c è una lettera o una cifra, 0 (falso) altrimenti	isalnum('Z') è positivo
int isxdigit(int c)	Restituisce un valore positivo (vero) se c è una cifra esadecimale, 0 (falso) altrimenti	isxdigit('A') è positivo

La libreria ctype.h

Funzione	Descrizione	Esempio
int islower(int c)	Restituisce un valore positivo (vero) se c è una lettera minuscola, 0 (falso) altrimenti	islower('a') è positivo
int isupper(int c)	Restituisce un valore positivo (vero) se c è una lettera maiuscola, 0 (falso) altrimenti	isupper('A') è positiva
int tolower(int c)	Se c è una lettera maiuscola viene restituita la corrispondente lettera minuscola, altrimenti viene restituito c inalterato	tolower('D') è uguale a 'd'
int toupper(int c)	Se c è una lettera minuscola viene restituita la corrispondente lettera minuscola, altrimenti viene restituito c inalterato	toupper('d') è uguale a 'D'

La libreria ctype.h

Funzione	Descrizione	Esempio
int isspace(int c)	Restituisce un valore positivo (vero) se c è un carattere di spaziatura, 0 (falso) altrimenti	isspace('\t') è positivo
int iscntrl(int c)	Restituisce un valore positivo (vero) se c è un carattere di controllo, 0 (falso) altrimenti	iscntrl('\a') è positivo
int ispunct(int c)	Restituisce un valore positivo (vero) se c è un carattere di punteggiatura, 0 (falso) altrimenti	ispunct('!') è positivo
int isprint(int c)	Restituisce un valore positivo (vero) se c è un carattere stampabile, 0 (falso) altrimenti	isprint('a') è positivo
int isgraph(int c)	Restituisce un valore positivo (vero) se c è un carattere stampabile diverso da spazio, 0 (falso) altrimenti	isgraph('A') è positivo

La libreria string.h

Funzione	Descrizione
char *strcpy(char *s1, const char *s2)	Copia la stringa s2 nell'array s1. Restituisce il valore di s1
char *strcat(char *s1, const char *s2)	Concatena la stringa s2 alla fine di s1. Il primo carattere di s2 si sostituisce al carattere nullo di s1. Restituisce il valore di s1
int strcmp(const char *s1, const char *s2)	Confronta le stringhe s1 e s2. La funzione restituisce 0, un valore minore di 0 o maggiore di 0 qualora s1 sia rispettivamente uguale, minore o maggiore di s2
char *strchr(const char *s, int c)	Individua la prima occorrenza del carattere c nella stringa s. Se c viene trovato, viene restituito un puntatore alla locazione di c in s, altrimenti viene restituito un puntatore NULL

La libreria string.h

Funzione	Descrizione
char *strstr(const char *s1, const char *s2)	Individua la prima occorrenza della stringa s2 in s1. Se s2 viene trovata, sarà restituito un puntatore alla locazione di s2 in s1, altrimenti viene restituito un puntatore NULL
int strlen(const char *s)	Restituisce la lunghezza della stringa s

Esempio

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

int main(){
   char s1[50], s2[50];
   printf("Immetti due stringhe\n");
   gets(s1);
   gets(s2);
   printf("Lunghezza di s1: %d, lunghezza di s2: %d\n",
          strlen(s1), strlen(s2));
   if(!strcmp(s1, s2))
       printf("Le due stringhe sono uguali\n");
   else
       printf("Le due stringhe non sono uguali\n");
   strcat(s1, s2);
   puts(s1);
   strcpy(s1,"ho copiato tutto");
   puts(s1);
}
```