Classe III^a Inf - Informatica - Le Stringhe in C

Le stringhe come vettori di caratteri

Una dei tipi di variabili più utilizzate in moltissimi linguaggi è appunto quella delle stringhe. Esse rappresentano per i dati alfanumerici (ossia quelli che sono considerati sotto forma simbolica o di carattere), la stessa cosa che le variabili numeriche sono per i dati numerici, interi, float o di altro tipo[1]. In alcuni linguaggi ciò è supportato direttamente in modo nativo, ossia le variabili stringa risultano direttamente delle variabili come tutte le altre a tutti gli effetti. Questo è il caso del BASIC, fin dalle prime versioni, del C++, e di altri linguaggi. Viceversa alcuni linguaggi, nelle loro implementazioni "standard", come il Pascal ed il C non hanno previsto il tipo stringa come tipo nativo, ma hanno preferito indicarlo ed implementarlo come sequenza di caratteri, ossia vettore di char.
Operativamente quindi ogni linguaggio ha sue sfumature e regole per trattare le stringhe.
In ogni caso tutti prevedono opportune funzioni per trattare le stringe e poterne ricavare parti di esse, lunghezza e altri dati.

Variabili e costanti stringa

Come accadeva per i valori interi, è possibile definire delle costanti stringa che possono essere utilizzati all'interno del programma in modo opportuno. Un esempio di ciò è la definizione della costante SCRITTA contenente la stringa "Ciao Mondo !":

```
#define SCRITTA "Ciao Mondo !"
void main()
{
      printf(SCRITTA);
}
```

in pratica la stringa definita in SCRITTA viene sostituita direttamente dal compilatore alla indicazione SCRITTA[2] della costante stessa. Al posto di SCRITTA, il compilatore inserisce la stringa "Ciao Mondo !" completa dei suoi delimitatori. Questa definizione di costante quindi presume una semplice sostituzione dell'indicatore[3].
In alcuni casi abbiamo già incontrato stringhe costanti, anche senza saperlo, come ad esempio nel caso del famoso programma Ciao mondo:

```
void main()
{
      printf("Ciao mondo !");
      getch();

}
```

1 Si ricorda, ad esempio, che il Fortran, prevede ad esempio anche un tipo nativo di variabile contenenti numeri complessi.
2 E' buona norma indicare sempre le costanti tutte in maiuscolo.
3 In C++ esiste un possibile, diverso modo, di definire le costanti.

In questi casi ciò che definiamo costanti stringa qualsiasi cosa che stà fra virgolette, in quanto il ruolo di questi delimitatori è proprio quello di delimitare serie di caratteri costituenti una stringa.

Il C++ definisce un apposito tipo stringa (String)[4], mentre il C non prevede l'uso di tale tipo di variabile. Tuttavia è chiaro che anche il C prevede metodiche per trattare le stringhe riconducendole al trattamento di vettori di caratteri, e affiancando a ciò opportuni indicatori e funzioni per l'input e l'output.

Quindi in C una possibile dichiarazione di stringa, sarà della forma:

```
char st[20];
```

ossia la stringa sarà costituita da un vettore di caratteri di una certa lunghezza. Ovviamente la stringa non potrà avere lunghezza che debordi l'area di memoria allocata dal vettore ossia non potrà superare in questo caso i 19 caratteri[5].

La stringa quindi non potrà essere più lunga della massima lunghezza della variabile, ma potrà viceversa essere più corta, ossia non occupare tutto il vettore. Per indicare quale parte del vettore sia occupata dalla stringa vera e propria e quale parte sia eccedente in C viene inserito un cosiddetto carattere terminatore di stringa, che indica ove termini la stringa. Tale carattere è corrispondente all' ASCII 0, ossia il codice 0 ASCII, usualmente ovviamente non utilizzato per i contenuti alfanumerici della stringa stessa.

Si osservi anche che esso non deve essere inserito dall'utente, ma viene inserito in modo automatico dalle funzioni preposte a leggere o copiare le stringhe, di modo che sia presente in modo opportuno. Nello stesso modo esso viene opportunamente considerato dalle funzioni preposte a stampare stringhe, che lo considerano, ovviamente in modo corretto, delimitatore delle stesse. Si consideri anche che il nome della variabile rappresenta, in questo caso, un puntatore che punta al punto di inizio del vettore di caratteri (e a cui quindi non può essere assegnato un valore stringa, come intuitivamente si sarebbe portati a fare), ed in generale l'accesso alla variabile avviene in modalità carattere per carattere. E' solo grazie ad opportune funzioni come la gets(...), strcpy(...) ed altre, è possibile in certe circostanze considerare la variabile stringa come un tutt'uno.

Assegnazione, input e output

Per assegnare una stringa ad una variabile vettore di char, data la natura del modo in cui è conformata la variabile che la contiene, non è possibile effettuare una assegnazione diretta della stessa, ossia è scorretta una assegnazione di questo tipo:

```
char st[20];
...
```

4 Analogamente a quanto fa l'Object Pascal

5 L'ultimo carattere della stringa serve come delimitatore della stessa (carattere ASCII 0) e quindi il numero utile di caratteri diminuisce di uno.

```
st = "Ciao Mondo !";
```

E' invece necessario utilizzare una apposita funzione strcpy(...) a due parametri preposta a copiare il contenuto di una stringa (sorgente) in un'altra (destinazione). La stringa sorgente potrà essere sia una stringa costante, sia un' altra variabile stringa.

Avremo quindi che potremo correttamente assegnare ad st una stringa in questo modo:

```
char st[20];
...
strcpy(st,"Ciao Mondo !");
```

oppure sarà possibile passare i contenuti di st in st2 tramite:

```
char st[20], st2[30];
...
strcpy(st,st2);
```

ovviamente sempre rispettando sempre i limiti di lunghezza della stringa destinazione.

Si fa anche presente che siccome le variabili stringa così definite sono a lunghezza fissa, la parte di memoria non sfruttata non viene recuperata per altri scopi, ma non viene semplicemente utilizzata, al pari di qualunque altro vettore o matrice non del tutto utilizzato in una elaborazione.

Modi alternativi per assegnare ad una variabile stringa (vettore di char) un valore, ma deprecati dato la ridondanza del metodo di assegnazione sono, quelli di assegnare il vettore carattere per carattere (in questo caso inserendo anche il carattere terminatore '\0'), o assegnare nello stesso modo il valore come inizializzazione C:

```
char st[20];
...
st[0] = 'C';  st[1] = 'i'; st[2] = 'a'; st[3] = 'o';
...
st[11] = '\0';
...
```

oppure

```
char st[20]= {'C','i','a','o',' ','m','o','n','d','o',' ','!','\0'};
```

La possibilità di usare questi metodi mette in evidenza la natura della stringa come vettore di caratteri, infatti è possibile operare su essa assegnazioni utilizzando una metodica per singolo carattere, come logico dalla struttura stessa della variabile (vettore di caratteri = variabile composta da una sequenza di caratteri).

Usualmente le due forme qui sopra riportate di assegnazione non vengono utilizzate.

Per la lettura dei valori delle stringhe da console[6] è consigliabile utilizzare comunque la gets(...), funzione preposta alla lettura di una variabile stringa. Un esempio di suo uso è il seguente:

6 Ricordiamo che stiamo operando con programmi che operano tramite console a caratteri.

```
char st[20];
...
gets(st);
```

tale funzione vuole che sia inclusa la libreria (header file) string.h.
La funzione prevede la conferma del dato inserito tramite la pressione di Invio e permette di leggere sia stringhe vuote che stringhe contenenti spazi.
Un modo alternativo per leggere strighe può essere quello di utilizzare la funzione scanf con il descrittore di formato %s. Tuttavia questa funzione soffre di alcune limitazioni (ad esempio riesce a leggere correttamente solo stringhe che non abbiano spazi (altrimenti vengono considerate analogamente a due stringhe separate e non legge stringhe vuote).
E' quindi sconsigliato utilizzare questa funzione per input su stringhe.
Per quanto riguarda l'output è possibile stampare stringhe con printf(....), abbinato al descrittore di formato %s, ed il nome del vettore di caratteri che forma la stringa.
Un esempio di una simile operazione è:

```
char st[20];
...
printf("La stringa risulta:  %s",st);
```

è possibile combinare tale formato con quello di altri dati quali interi e numeri float, senza problemi.
Infine si tenga conto che esistono numerose funzioni atte a permettere di svolgere celermente le più comuni operazioni con le stringhe. Queste funzioni sono contenute anch'esse nella libreria string.h. Tra di esse ricordiamo:

- la funzione strlen(...) che, presa in input una stringa, ne calcola la lunghezza

  ad es.:

  ```
  char st[20];
  int l;
  ...
  printf("La lunghezza della stringa risulta:  %i",strlen(st));
  ```

  il calcolo della lunghezza di una stringa è spesso richiesto per permettere poi di scandire i singoli caratteri della stringa stessa, per elaborazioni di vario tipo.

- la funzione strcpy(...,...) che come abbiamo visto in precedenza permette la copia di stringhe, e che è utile per le assegnazioni delle stringhe.

- La funzione strcat(...,...), che permette di concatenare una stringa ad un altra. La funzione accetta due parametri e rende il valore concatenato nel primo di essi e alternativamente in uscita. In altri linguaggi esistono appositi concatenatori di stringa, come ad esempio il simbolo + in Object Pascal.

- La funzione strcmp(st1,st2), sempre a due parametri stringa. Essa rende:
  - il valore 0 se le due stringhe sono uguali
  - un valore positivo se st1 è maggiore di st2
  - un valore negativo se st1 è minore di st2

  Da quanto detto si comprende che non è corretto comparare due stringhe direttamente come si potrebbe intuitivamente fare con confronti del tipo:

  ```
  if (st1 > st2)...
  while (st1 != "pippo")...
  ```

  in realtà questi confronti attengono puntatori alla stringa e quindi un confronto corretto non è possibile tra questi elementi.

  A proposito di questa funzione c'è da osservare che, come esiste un ordinamento alfabetico dei caratteri, esiste anche un noto ordine alfabetico per le stringhe[7], che considerano progressivamente il valore del primo carattere di due stringhe, e nel caso di uguaglianza passano al secondo carattere e così via fino, fino a quando uno dei caratteri indichi una differenza tra le due stringhe. Si ricorda che nel caso di uguaglianza di tutti i caratteri, la stringa più lunga risulta maggiore di quella più corta, mentre se le due stringhe sono di lunghezza uguale si conclude la coincidenza delle due e quindi l'uguglianza (valore 0 reso dalla funzione strcmp(..,..)).

Esempi di elaborazione su stringhe

Ovviamente molte di queste funzionalità sono realizzabili anche con del normale codice C direttamente sulle stringhe stesse. Può essere un utile esercizio cercare di realizzare ciò. Ad esempio la funzione strlen(...) corrisponde all' equivalente codice C:

```
#include "string.h"
#include "conio.h"
#include "stdio.h"

void main()
{
       char st1[20];
       int scan;

       // legge la stringa
       gets(st1);

       // individua la sua lunghezza
       scan = 0;
       while (st1[scan] != '\0')  scan++;

      // stampa la lunghezza trovata
       printf("La lunghezza della stringa e':  %i", scan);

       getch();
```

7 Quello usato, ad esempio, in tutti i vocabolari.

```
}
```

La funzione strcat è possibile realizzarla con il codice:

```
#include "string.h"
#include "conio.h"
#include "stdio.h"

void main()
{
       char st1[20], st2[30];
       int i, scan1, scan2;

        // lettura delle due stringhe
       gets(st1);
       gets(st2);

        // individua la lunghezza della prima stringa
       scan1 = 0;
       while (st1[scan1] != '\0')  scan1++;

        // individua la lunghezza della seconda stringa
       scan2 = 0;
       while (st2[scan2] != '\0')  scan2++;

        // effettua la concatenazione
       for (i = 0; i < scan2; i++)  st1[i + scan1] = st2[i];

      // aggiunge il terminatore
       st1[i + scan1] = '\0';

       printf("La stringa concatenata e':  %s", st1);

       getch();

}
```

ed in modo analogo abbiamo la possibilità di riscrivere in modo adeguato la funzione strcmp, sempre effettuando opportune scansioni dei vettori e opportune operazioni sui singoli caratteri:

```
#include "stdio.h"
#include "conio.h"

void main()
{
       char st1[20], st2[30];
       int scan1, scan2, lim, cc;

      // lettura delle stringhe
       gets(st1);
       gets(st2);

      // calcolo lunghezza della stringa 1
       scan1 = 0;
       while (st1[scan1] != '\0')  scan1++;

      // calcolo lunghezza della stringa 2
       scan2 = 0;
       while (st2[scan2] != '\0')  scan2++;

        // individua come limite del controllo la lunghezza della stringa più breve
       if (scan1 >= scan2)
          lim = scan2;
       else
```

```
            lim = scan1;

         // individua la prima lettera che determina se una delle stringhe
         // superi l'altra
         for (cc = 0; cc < lim; cc++)
         {
            if (st1[cc] > st2[cc]) { printf("%i", (st1[cc] - st2[cc]));  break; }
            if (st1[cc] < st2[cc])  { printf("%i",-(st2[cc] - st1[cc]));  break; }
         }

         // se il controllo e' arrivato alla fine della stringa più breve
         // senza un esito allora...
         if (cc == lim) {
                // se le due stringhe hanno la stessa lunghezza...
                if (scan1 == scan2)   printf("0");
                // una risulta piu' lunga dell'altra....
                if (scan1 > scan2)    printf("%i",st1[cc]);
                if (scan1 < scan2)    printf("%i",-st2[cc]);
                }

         getch();
}
```

si noti in questi esempi, l'uso della scansione delle stringhe per caratteri, e la ripetuta, necessaria, determinazione della lunghezza delle stringhe.