Stringhe

Prof. Lorenzo Porcelli

definizione

Una stringa è un vettore di caratteri terminato dal carattere nullo ‘\0’.

Il carattere nullo finale permette di determinare la lunghezza della stringa.

```
char vet[32];
char corso[ ] = {‘i’, ‘n’, ‘f’, ‘o’, ‘r’, ‘m’, ‘a’, ‘t’, ‘i’, ‘c’, ‘a’, ‘\0’ };
char nome[ ] = “Lorenzo”;
char errore[ ] = {‘e’, ‘r’, ‘r’, ‘o’, ‘r’, ‘e’ };
```

utilizzo

```
char corso[ ] = “informatica”;
int x = 5;

printf(“%s”, corso);
corso[7] = ‘\0’;
printf(“%s”, corso);

scanf(“%s”, corso);
printf(“%s %d”, corso, x);
```

Output prodotto:

informatica

informa

inseriamo la parola

elettrotecnica

Output prodotto:

elettrotecnica 24931

Perché?

Operazioni su stringhe

Trasformare in maiuscolo i caratteri di una stringa.

Prototipo:

```
char* upper( char * );
```

La funzione restituisce lo stesso indirizzo avuto in ingresso. Perché?

```
char *upper( char s[] ) {
   int i=0;

   for(i=0; s[i]!=0; ++i)
        if( minuscolo(s[i]) )
                s[i]+= 'A'-'a';

   return s;
}
```

string.h

Contiene i prototipi delle funzioni che operano sulle stringhe.

Deve essere incluso nei programmi che operano con le stringhe.

Maggiori dettagli si trovano nell'help del compilatore utilizzato.

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

strcat

char ***strcat**(char *s1, const char *s2);

Copies the string s2, including its null character, to successive elements of the array of *char* that stores the string s1, beginning with the element that stores the null character of s1. It returns s1.

```
char c[100]=“Porcelli”;
char n[100]=“Lorenzo”;
printf(“%s\n”,strcat(c,n);
printf(“%s”, c);
```

strchr

char ***strchr**(const char *s, int c);

It searches for the first element of the string s that equals (char)c. It considers the terminating null character as part of the string. If successful, the function returns the address of the matching element; otherwise, it returns a null pointer.

```
char x[128]="Asinello";
printf("%s",strchr(x, 'n'));
```

strcmp

int **strcmp**(const char *s1, const char *s2);

The function compares successive elements from two strings, s1 and s2, until it finds elements that are not equal.

- If all elements are equal, the function returns zero.
- If the differing element from s1 is greater than the element from s2 the function returns a positive number.
- Otherwise, the function returns a negative number.

```
char s1[32]=“politecnico”;
char s2[32]=“polinomio”;
printf(“%d”, strcmp(s1, s2) );
```

strcpy

char ***strcpy**(char *s1, const char *s2);

The function copies the string s2, including its terminating null character, to successive elements of the array of *char* whose first element has the address s1. It returns s1.

```
char c[100]=“Porcelli”;
char n[100]=“Lorenzo”;
printf(“%s\n”,strcpy(c,n);
printf(“%s”, c);
printf(“%s”, strcpy(&c[1], n) );
```

strlen

size_t **strlen**(const char *s);

The function returns the number of characters in the string s, *not* including its terminating null character.

```
char s1[32]=“politecnico”;
printf(“%d”, strlen(s1) );
```

strncat

char ***strncat**(char *s1, const char *s2, size_t n);

It copies the string s2, *not* including its null character, to successive elements of the array of *char* that stores the string s1, beginning with the element that stores the terminating null character of s1. The function copies no more than n characters from s2. It then stores a null character, in the next element to be altered in s1, and returns s1.

strncmp

int **strncmp**(const char *s1, const char *s2, size_t n);

- The function compares successive elements from two strings, s1 and s2, until it finds elements that are not equal or until it has compared the first n elements of the two strings.
- If all elements are equal, the function returns zero.
- If the differing element from s1 is greater than the element from s2 the function returns a positive number.
- Otherwise, it returns a negative number.

strncpy

char ***strncpy**(char *s1, const char *s2, size_t n);

The function copies the string s2, *not* including its terminating null character, to successive elements of the array of *char* whose first element has the address s1. It copies no more than n characters from s2. The function then stores zero or more null characters in the next elements to be altered in s1 until it stores a total of n characters. It returns s1.

strtok

char ***strtok**(char *s1, const char *s2);

- If s1 is not a null pointer, the function begins a search of the string s1. Otherwise, it begins a search of the string whose address was last stored in an internal static-duration object on an earlier call to the function, as described below. The search proceeds as follows:
- The function searches the string for begin, the address of the first element that equals *none* of the elements of the string s2 (a set of token separators). It considers the terminating null character as part of the search string only.
- If the search does not find an element, the function stores the address of the terminating null character in the internal static-duration object (so that a subsequent search beginning with that address will fail) and returns a null pointer. Otherwise, the function searches from begin for end, the address of the first element that equals *any one* of the elements of the string s2. It again considers the terminating null character as part of the search string only.
- If the search does not find an element, the function stores the address of the terminating null character in the internal static-duration object. Otherwise, it stores a null character in the element whose address is end. Then it stores the address of the next element after end in the internal static-duration object (so that a subsequent search beginning with that address will continue with the remaining elements of the string) and returns begin.

strtok

```
int main(){
char s[128]=
"Lorenzo:Porcelli:18:01:1957:Alessandria:docente";
    char *x;
    x=strtok(s,":");
    while( x ) {
          printf("%s\n", x);
          x=strtok(NULL, ":");
    }
    return 0;
}
```

```
Lorenzo
Porcelli
18
01
1957
Alessandria
docente
```