

Titolo unità didattica: Stringhe ed elaborazione di testi [09]

Titolo modulo : Function in C per l'elaborazione di testi [05-C]

Sviluppo di function in C per azioni su stringhe ed elaborazione di testi

Argomenti trattati:

- ✓ function in C per l'analisi delle componenti di un testo
- ✓ function in C per lo string matching
- ✓ function in C per il matching migliore

Prerequisiti richiesti: AP-09-02-T, AP-09-04-C

analisi componenti di un testo, string matching, best matching

✓ trattamento testi:

word-processor

correttori ortografici

estrazione parole chiave di un libro

✓ antivirus:

ricerca della firma del virus

✓ siti web:

classificazione dei siti in funzione di
determinate parole rilevate nel contenuto
delle pagine

✓ bioinformatica:

ricerca di determinate sequenze all'interno del
DNA

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

```
#define TRUE 1
```

```
#define FALSE 0
```

```
void main()
```

```
{
```

```
    char c;
```

```
    int conta_c, conta_p, conta_l, in_p;
```

```
    conta_c = 0; conta_p = 0; conta_l = 0; in_p = FALSE;
```

```
    while ((c = getchar()) != EOF) {
```

```
        conta_c++;
```

```
        if (c == '\n')
```

```
            conta_l++;
```

```
        if (c == ' ' || c == '\n' || c == '\t')
```

```
            in_p = FALSE;
```

```
        else if (in_p == FALSE) {
```

```
            in_p = TRUE;
```

```
            conta_p++;
```

```
        }
```

```
    }
```

```
    printf ("numero di caratteri= %d\nnumero di parole =  
%d\nnumero di linee = %d\n", conta_c, conta_p, conta_l);
```

```
}
```

conteggio del numero di caratteri,
del numero di parole e
del numero di linee di un testo

conteggio del numero di caratteri,
del numero di parole e
del numero di linee di un testo

```
void conta_tutto(char *testo,
                 int *conta_c,
                 int *conta_p,
                 int *conta_l)
{
    int i, in_p;
    i = 0; *conta_c = 0; *conta_p = 0; *conta_l = 0;
    in_p = FALSE;
    while (testo[i] != '\0')
    {
        (*conta_c)++;
        if (testo[i] == '\n')
            (*conta_l)++;
        if (testo[i] == ' ' || testo[i] == '\n' || testo[i] == '\t')
            in_p = FALSE;
        else if (in_p == FALSE)
        {
            in_p = TRUE;
            (*conta_p)++;
        }
        i++;
    }
    (*conta_l)++;
}
```

conteggio del numero di caratteri, del numero di parole e del numero di linee ed estrazione di **token** di un testo

```
#include <stdio.h>
#include <string.h>
#define TRUE 1
#define FALSE 0

void conta_tutto(char *testo, int *numero_caratteri,
                 int * numero_parole, int * numero_linee);

void main()
{
    char testo[100], separatori[]={' ', '\n', '\t', '\0'}, *token;
    int numero_caratteri, numero_parole, numero_linee, j;
    gets(testo);
    conta_tutto(testo, &numero_caratteri, &numero_parole, &numero_linee);
    printf("conta_tutto:\n numero di caratteri= %d\n numero di parole = %d\n numero di linee = %d\n", numero_caratteri, numero_parole, numero_linee);
    /* uso di strtok */
    printf("elenco delle parole (token estratti con strtok):\n");
    j = 1;
    printf("1-sima parola = %s\n", strtok(testo, separatori));
    while((token = strtok('\0', separatori)) != '\0')
    {
        j++;
        printf("%d-sima parola = %s\n", j, token);
    }
}
```

Nel mezzo del cammin di nostra vita mi ritrovai in una selva oscura
conta_tutto:

numero di caratteri = 67

numero di parole = 13

numero di linee = 1

elenco delle parole (token estratti con strtok):

1-sima parola = Nel

2-sima parola = mezzo

3-sima parola = del

4-sima parola = cammin

5-sima parola = di

6-sima parola = nostra

7-sima parola = vita

8-sima parola = mi

9-sima parola = ritrovai

10-sima parola = in

11-sima parola = una

12-sima parola = selva

13-sima parola = oscura

string (pattern) matching

(occorrenze di una sottostringa in una stringa)

```
int string_matching(char chiave[], char testo [])
{
    int n, m, i, conta_chiave;
    n = strlen(chiave);
    m = strlen(testo);
    conta_chiave = 0;
    for (i=0; i<m-n; i++)
        if (strncmp(chiave, &testo[i], n) == 0)
            conta_chiave++;
    return conta_chiave;
}
```

string (pattern) matching

(occorrenze di una sottostringa in una stringa)

```
#include <stdio.h>
#include <string.h>
int string_matching(char *chiave, char *testo);
void main()
{
    char miotesto[100], miachiave[20];
    int n_occorrenze;
    puts("inserire il testo:\n");
    gets(miotesto);
    puts("\ninserire la chiave:\n");
    gets(miachiave);
    n_occorrenze = string_matching(miachiave, miotesto);
    printf("\nla stringa %s appare %d volte nel testo\n\n%s\n", miachiave, n_occorrenze, miotesto);
}
```


string (pattern) matching

(occorrenze di una sottostringa in una stringa)

```
#include <stdio.h>
#include <string.h>
int string_matching(char *chiave, char *testo);
void main()
{
    char *miotesto, *miachieve;
    int n_occorrenze;
    puts("inserire il testo:\n");
    gets(miotesto);
    puts("\ninserire la chiave:\n");
    gets(miachieve);
    n_occorrenze = string_matching(miachieve, miotesto);
    printf("\nla stringa %s appare %d volte nel testo\n\n%s\n", miachieve, n_occorrenze, miotesto);
}
```

string (pattern) matching

(occorrenze di una sottostringa in una stringa)

inserire il testo:

ATTCAGGCACGAGTTACAGGCGAGGAGTCAGGCTTGTGT

inserire la chiave:

AGG

la stringa AGG appare 4 volte nel testo

ATTCAGGCACGAGTTACAGGCGAGGAGTCAGGCTTGTGT

string (pattern) matching

(individuazione di una sottostringa in una stringa)

```
#include <stdio.h>
#include <string.h>
void main()
{
    char miotesto[100], miachiave[10], *punt, sstringa[10];
    int n_occorrenze;
    puts("inserire il testo: ");
    gets(miotesto);
    puts("\n inserire la chiave: ");
    gets(miachiave);
    punt = strstr(miotesto, miachiave);
    strncpy(sstringa, punt, strlen(miachiave));
    printf("\n usando strstr: %s\n", sstringa);
}
```

```
inserire il testo:
ATTCAGGCACGAGTTACAGGCGAGGAGTCAGGCTTGTGT
inserire la chiave: AGG
usando strstr: AGG
```

best matching

(allineamento di stringhe)

```
#include <stdio.h>
#include <string.h>
int matching_migliore(char *chiave, char *testo) ;
int punteggio_matching(char *a, char *b, int n) ;
void main()
{
    int indice_bm, i;
    char miotesto[50], miachiave[10] ;
    printf("inserire la chiave: ") ;
    gets(miachiave) ;
    printf("inserire il testo:\n") ;
    gets(miotesto) ;
    indice_bm = matching_migliore(miachiave, miotesto) ;
    puts("Risultato del best matching\n") ;
    printf(" inizio sottostringa: %d\n sottostringa di
        best matching: ", indice_bm) ;
    for (i=0; i<strlen(miachiave); i++)
        putchar(miotesto[indice_bm+i]) ;
}
```

best matching

(allineamento di stringhe)

```
int matching_migliore(char *chiave, char *testo)
{
    int i, n, m, punteggio_max, punteggio, indice=0;
    n = strlen(chiave);
    m = strlen(testo);
    punteggio_max = 0;
    for (i=0; i<m-n; i++)
    {
        punteggio = punteggio_matching(chiave, &testo[i], n);
        if (punteggio > punteggio_max)
        {
            punteggio_max = punteggio;
            indice = i;
        }
    }
    return indice;
}
```

best matching

(allineamento di stringhe)

```
int punteggio_matching(char *a, char *b, int n)
{
    int i, n_caratteri_uuali;
    n_caratteri_uuali = 0;
    for (i=0; i<n; i++)
        if (a[i] == b[i])
            n_caratteri_uuali++;
    return n_caratteri_uuali;
}
```

inserire la chiave: dal cammino

inserire il testo:

Nel mezzo del cammin di nostra vita mi ritrovai in una
selva oscura

Risultato del best matching

inizio sottostringa: 10

sottostringa di best matching: del cammin