Caruso Davide 4° B 23/12/03

RELAZIONE DI SISTEMI

Testo del problema:

Scrivere un programma in C++ che ,dopo aver acquisito due stringhe , indichi se la seconda stringa è una sottostringa della prima. Utilizzare per la ricerca della sottostringa una funcion esterna scritta in linguaggio Assembly.

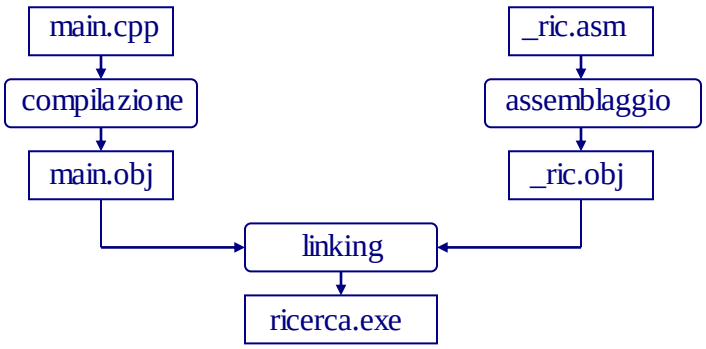
Risposta:

Individuazione dei dati di input e di output:

Dati di input: **DC**(vettore di byte) ,**s** (vettore di byte)
Dati di output: **al**(trovato=1,non trovato =0).

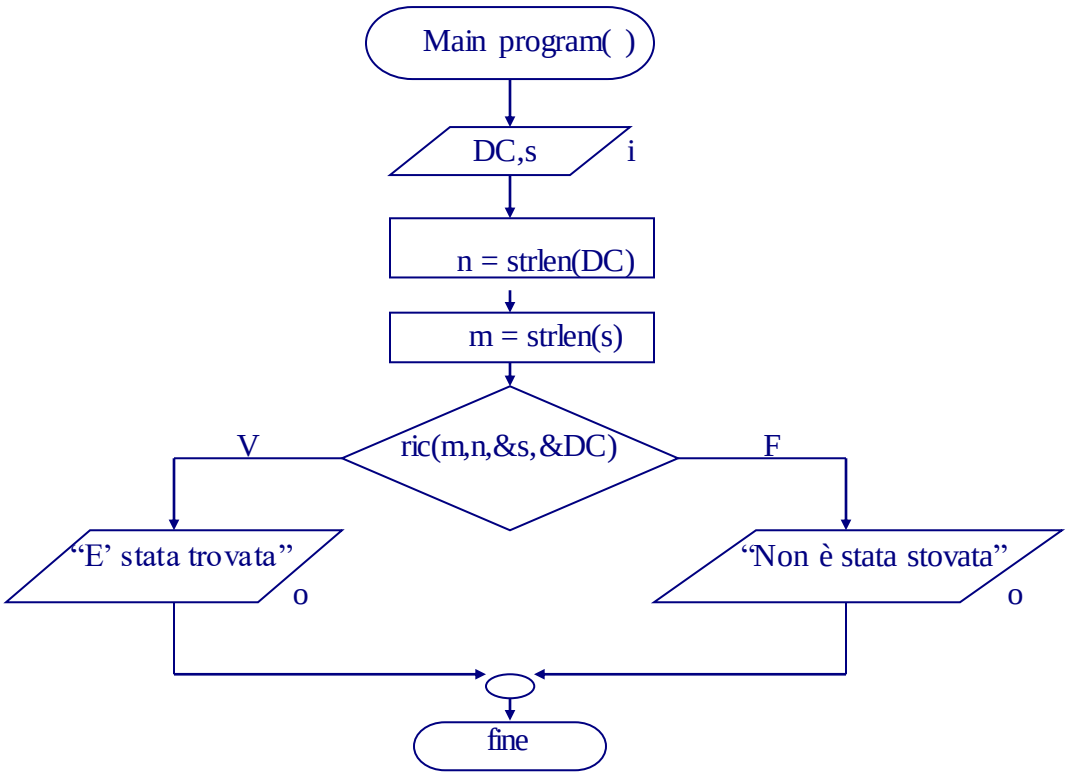
Analisi del problema:

In base a quanto chiesto dal problema , il programma sarà costituito da due file sorgenti :dal file dove verrà implementato il main program in linguaggio c++ , che chiamerò main.cpp , e dal file dove verrà implementata la funcion in linguaggio assembly , che chiamerò _ric.asm. Entrambi i file , dopo la fase di compilazione , verranno linkati insieme , per generare un unico file eseguibile(vedi schema sottostante).



Adesso inizierò a descrivere il main program che ,come potete vedere dal diagramma di flusso sottostante, ha il compito di leggere le due stringhe da tastiera e di identificare se la seconda stringa è una sottostringa della prima ,richiamando la procedura esterna ric ,che vuole come parametri attuali l'indirizzo delle due stringhe con le loro rispettive dimensioni e restituisce 1 se la sottostringa è stata trovata , cioè è contenuta nella prima stringa almeno una volta , e 0 se la sottostringa non è stata trovata.

Diagramma di flusso:



Traduzione dell'algoritmo in C++

Il risultato della traduzione del precedente algoritmo in linguaggio C++ nel file main.cpp è stato il seguente:

```
/* file main.cpp */
#include<iostream.h>
#include<STRING.h>
#include<conio.h>
extern "C" unsigned char far ric(int ,int ,char [],char []);
void main()
{
 int n,m;
 char DC[50],s[10];
 clrscr();                    //viene cancellata la videata
 cout << "\nInserisci un testo :";
 cin.getline(DC,50); //questa funzione , riconosce lo spazio come un carattere ,e non come un terminatore
 cout << "Inserisci la parola da cercare:";
 cin.get(s,10);
 n = strlen(DC);
 m = strlen(s);
 if(ric(m,n,s,DC)) cout << "E' stata trovata!"<<endl;
    else cout << "Non è stata trovata!"<<endl;
 getch();
}
```

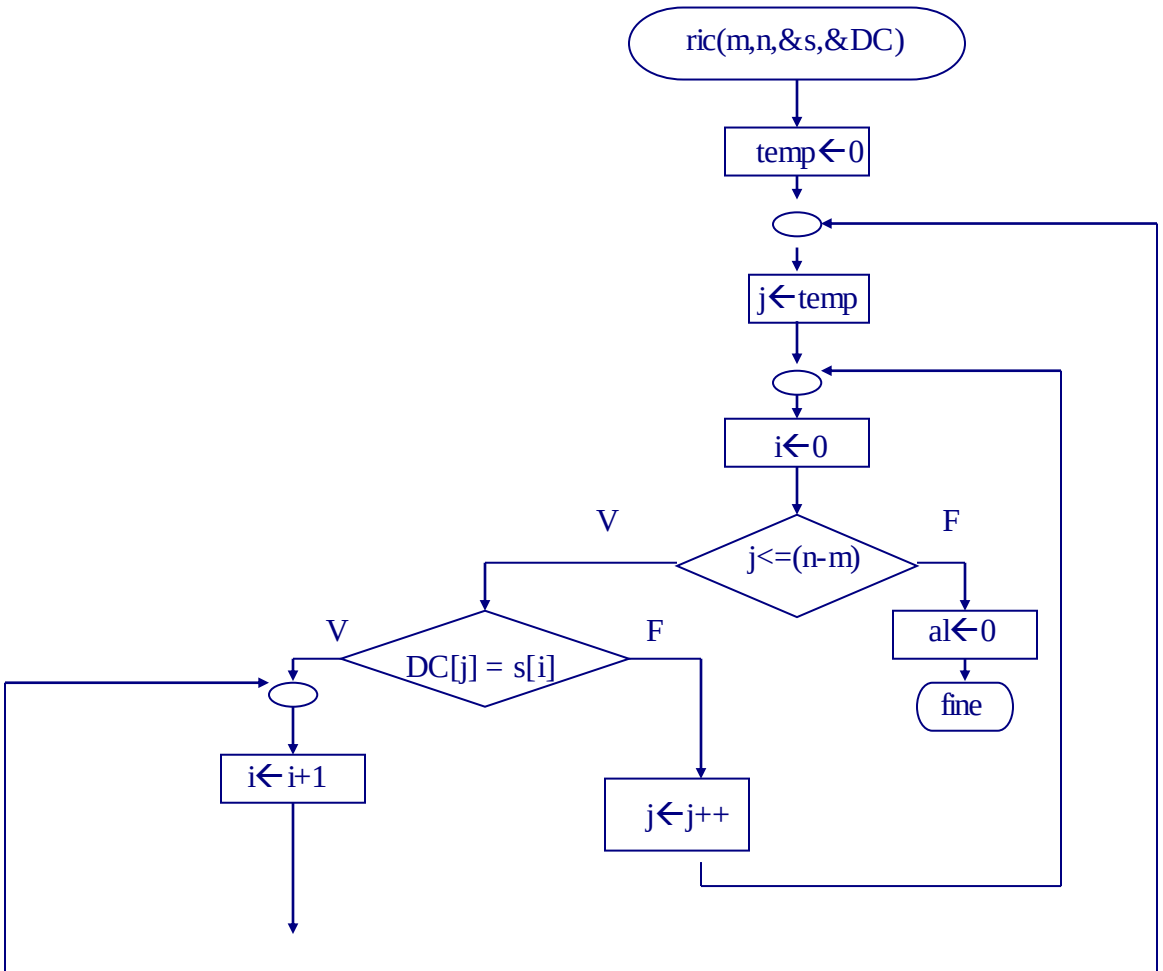
E' da notare che per il momento è possibile soltanto a compilare il programma , perché , naturalmente , nella fase di linkaggio la function _ric(..) non viene trovata .

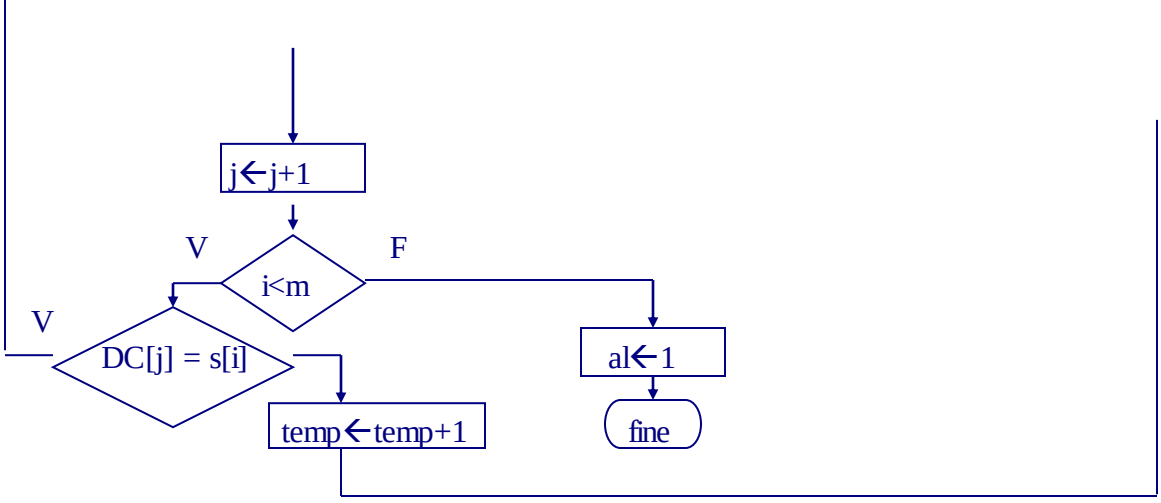
Per realizzare la function _ric(..)ho costruito un semplice programma interamente in assembly(nel file ricerca.asm) che farà da supporto alla function , in modo da agevolare la fase di debuging.

La funcion in questione è basata sul seguente algoritmo:

Diagramma di flusso:

Variabili di lavoro: **temp**(bp[6]) ,**i** (Di),**j**(SI)





L'algoritmo risulta essere costituito da un ciclo **for** in cui viene cercato l'elemento **s**[0] nella stringa **DC**, a partire dal posto **DC**[**temp**]; se in questa fase l'elemento non viene trovato , cioè se il contatore **j** , inizializzato a zero, risulta maggiore di n-m , la funzione restituirà in al zero, se invece viene trovato ,si passerà ad un altro ciclo **for** in cui vengono confrontate le stringhe **s** e **DC** fin quando saranno uguali o fin quando **i** sarà minore di **m** .Se si uscirà dal ciclo perché il contatore **i** è uguale a **m** , significa che si è trovata la sottostringa e quindi si dovrà uscire dalla funcion con restituzione in al di 1 , se invece si uscirà dal ciclo perché l'uguaglianza **DC**[**j**]=**s**[**i**] non è più vera , invece , si dovrà ripristinare **j** al valore successivo a quello che aveva all'uscita del primo ciclo **for** ed eseguire una nuova ricerca.

Traduzione dell'algoritmo in linguaggio assembly

```
;******************
;** file ricerca.asm **
;******************
include d:\tasm\stdlib.lib          ;in questa libreria si trova la definizione della macro print_mess che
                                    ;ha la funzione i stampare una stringa situata in ds passata come parametro
Dati segment
        stringa_1  db "Qui si trova stringa_2"      ;dichiarazione e inizializzazione della stringa
        n dw 22                                     ;n viene inizializzato con la dimensione di stringa_1
        stringa_2  db "stringa_2"                   ;dichiarazione e inizializzazione della sottostringa stringa_1
        m dw 9                                      ;m viene inizializzato con la dimensione di stringa_2
        messaggio1  DB "La sottostringa e' stata trovata!",0ah,0dh,'$'
        messaggio2  DB "La sottostringa non e' stata trovata!",0ah,0dh,'$'
        messEXIT  db "Premere un tasto per uscire",0ah,0dh,'$'
Dati ends

Sistema segment stack 'stack'
        .....
Sistema ends

Codice segment
        ;definizione dei segmenti
            ……
inizio:
        ;inizializzazione dei registri di segmento
            ……
;inizio programma
        jmp start
;spazio riservato alla procedura
```

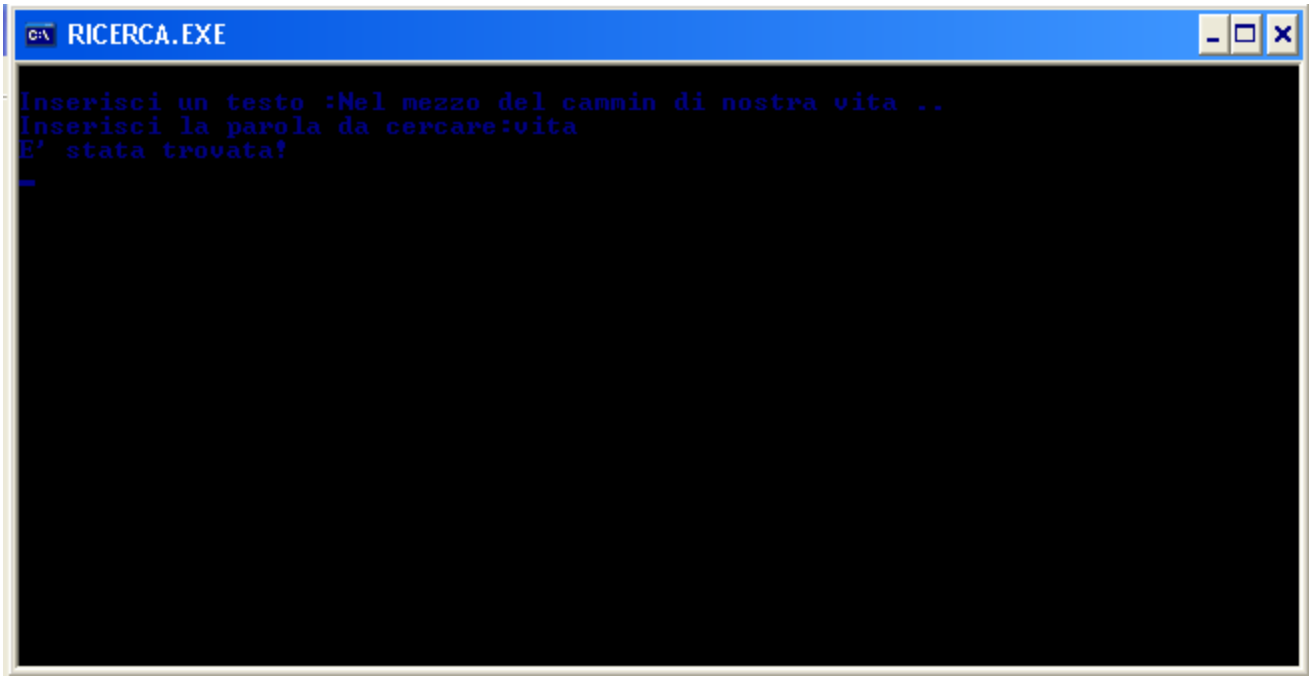
```
_ric  proc
          push bp
          mov bp,sp
          mov cx,bp[6]                    ;viene recuperato n della stack
          sub cx,bp[4]
          add cx,1                        ;cx viene inizializzato a n-m+1
          mov bp[6],cx                    ;bp[6] rappresenterà da questo momento la variabile temp
ciclo:
          mov di,bp[10]                   ;viene assegnato a di l'offset si stringa_1
          mov si,bp[8]                    ;viene assegnato a di l'offset si stringa_2
          mov al,[si]                     ;viene assegnato ad al il valore di s[0]
          cmp cx,0
          je Stri_non_trovata
          repnz scasb                     ;viene effettuato il confronto tra i byte al e [es:di] fin quando non
                                          ;si verifica un'uguaglianza o cx è uguale a 0
          jne Stri_non_trovata            ;viene effettuato il salto se l'uguaglianza non si è mai varificata
          inc si
          mov bx,bp[4]                    ;bx (rappresentante i)viene inizializzato a m
ciclo2:
          dec bx                          ;viene ripetuto il confronto tra i byte [ds:si] e [es:di] bx volte
          je Stri_trovata                 ;se l'uguaglianza si verifica
          cmpsb
          je  ciclo2
          inc  bp[10]                     ;se l'uguaglianza non si verifica viene incrementato l'offset di
          dec bp[6]                       ;stringa_1 , decrementato n
          mov cx,bp[6]                    ;e ripristinato cx
          jmp  ciclo
Stri_non_trovata:
          xor al,al
          jmp  fine
Stri_trovata:
          mov al,1
fine:
          pop bp
          ret 8                               ;viene "pulito" lo stack
          endp
start:
          push offset  stringa_1              ;passaggio dei parametri e chiamata
          push offset  stringa_2              ;alla procedura near  ric,
          push n                              ;che metterà in al zero , se stringa_2
          push m                              ;non è presente in stringa_1, o 1 ,se
          call  _ric                          ;è presente.
trovata:
     print_mess  messaggio1
finecode:
    ;blocco del programma fin quando non viene premuto un tasto
          print_mess  messEXIT
          MOV AH,01H
          INT 21H
;ritorno  al sistema  operativo
          mov al,00h
          mov ah,4ch
          int  21h
codice ends
end inizio
```

Dopo aver ultimato le operazioni di deguging al programma appena descritto , ho implementato nel file _ric.asm la procedura _ric , apportandole le seguenti modifiche:

```
public _ric
Codice segment
         Assume CS:codice
inizio:
_ric  proc far
       push bp di si   ;vengono copiati nello stack i registri che saranno modificati
       mov bp,sp
       mov cx,bp[12]
       sub cx,bp[10]
       add cx,1
       mov bp[12],cx
ciclo:
       mov di,bp[16]
       mov si,bp[14]
       mov al,[si]
       cmp cx,0
       je Stri_non_trovata
       repnz scasb
       jne Stri_non_trovata
       inc si
       mov bx,bp[10]
ciclo2:
       dec bx
       je Stri_trovata
       cmpsb
       je  ciclo2
       inc bp[16]
       dec bp[12]
       mov cx,bp[12]
       jmp ciclo
Stri_non_trovata:
       xor al,al
       jmp fine
Stri_trovata:
       mov al,1
fine:
       pop si di bp   ;ripristino dei registi modificati dalla funcion
       ret           ;lo stack viene pulito dal compilatore
       endp
codice ends
end inizio
```

Note:
Come potete vedere la procedura near è stata convertita in far ed inoltre è stata definita public .

In fine ho inserito i due file (main.cpp e _ric.asm) in un progetto cpp (ricerca.prj) e ho lincato il file main.cpp , ottenendo il risultato riportato in basso:



Commento:

Il programma , dopo gli ultimi controlli in fase di debuging , non ha presentato errori ; ho riscontrato soltanto due messaggi di warning durante la fase di compilazione del file _ric.asm tralasciabili , nelle rige 27 e 28, indicanti la perdita di valore dei parametri n e m salvati nello stack prima della chiamata alla procedura.