Tutorato 01 Programmazione

Giulio Umbrella

Contents

1 Reverse **1**

2 Reverse line **1**
2.1 Pseudo codice 1
2.2 Implementazione 2

3 Get line **3**
3.1 Caratteri in input 3
3.2 Nuova riga 3
3.3 Implementazione 4

4 Reverse text **5**

5 Standard input **5**

6 Esercizi aggiuntivi **5**

1 Reverse

L'obbiettivo e' creare un programma `reverse_text` che prenda in input un testo e esegua il reverse di ciascuna riga.

1. Creare una funzione `reverse_line` che dato un array di char ne inverta la posizione
2. Creare una funzione `get_line` che dato un testo in input estragga una singola riga
3. Creare una programma `reverse_text` che combine le due funzioni

2 Reverse line

La funzione `reverse_line` prende in input un array di char di caratteri e ne produce l'inverso; ad esempio dato in input 'CIAO' produce 'OAIC'.

Per prima cosa vediamo come modellare il problema, poi forniamo una soluzione

2.1 Pseudo codice

Per modellare un array di n char possiamo pensare ad un vettore di n elementi con indici che vanno da 0 (primo elemento) a n-1 (ultimo elemento)

Dati due indici i e j, sappiamo che dobbiamo invertire gli **elementi** dell'array a in quella posizione, indichiamo questa operazione con a[i] <-> a[j].

Gli indici i e j **non** possono essere scelti a caso; per invertire il vettore i e j devono avere la stessa distanza da destra e sinistra. Da questa osservazione notiamo che:

- Il valore iniziale di i e' 0 mentre j e' n-1
- Alla fine di ogni aumentiamo i di 1 e decrementiamo j di 1

Infine, dobbiamo pensare ad una condizione di stop. Intuitivamente, sappiamo che dobbiamo proseguire fino alla meta' dell'array. Possiamo considerare due casi separati array di lunghezza pari e dispari:

- Se l'array e' di lunghezza dispari quando i e j hanno lo stesso valore, siamo arrivati all'elemento centrale e ci possiamo fermare.
- Se invece l'array e' pari, superata la meta' il valore di i superera' il valore di j.

Quindi possiamo la condizione di stop si realizza quando i >= j. In altre parole, rimaniamo all'interno del loop fino a che i < j.

Possiamo formalizzare il problema con il seguente **pseudo codice**; un linguaggio ad alto livello che formalizza il problema nei suoi passaggi fondamentali.

```
INPUT: array s di n elementi
OUTPUT: array s con elementi in posizione nverita
1. i <- 0
2. j <- n-1
3. while i < j
4.     a[i] <-> a[i]
5.     i <- i+1
6.     j <- j-1
```

Le righe 1. e 2. rappresentano la nostra condizione iniziale; la riga 3. la condizine di permanenza nel ciclo (quando viene violata invochiamo lo stop); mentre le righe 4.-6. l'operazione che performiano ad ogni iterazione.

2.2 Implementazione

Adesso possiamo implementare la funzione:

```c
void reverse_line(char s[], int len){

    int i,j,tmp;

    i = 0;
    j = len - 1;

    if(s[j] == '\n')
        j--;

    while(i < j){
        tmp = s[i];
        s[i] = s[j];
        s[j] = tmp;
        i++;
        j--;
    }

    return 0;
}
```

Dato un array s, la funzione reverse_line realizza l'inverso *in place*, ossia modifica gli elementi dell'array di input senza creare un nuovo array.

Per eseguire lo scambio usiamo una variabile `tmp` per salvare il valore di `s[i]` prima di modificarlo.

Notiamo che stiamo aggiungendo un controllo sull'ultimo char dell'array; ne rappresenta una nuova riga decrementiamo il valore di j per preservare la formattazione per riga.

3 Get line

Ora possiamo preseguire con la funzione per ottenere la funzione per ottenere una singola riga di testo. Dobbiamo chiarire tre punti:

1. Come ottenere caratteri in input
2. Capire quando inizia una nuova riga di testo
3. Implementare una funzione per ottenere righe

NB vedremo piu' avanti come dare un input al programma; per il momento supponiamo di avere un testo input formato da piu' righe.

3.1 Caratteri in input

Per ottenere dei caratteri dallo standard input, possiamo usare la funzione `getchar` che prende caratteri dallo standard input. Per l'operazione inversa invece usiamo `putchar`

```
#include <stdio.h>
int main()
{
    int c;

    c = getchar();
    while( c!= EOF){
        putchar(c);
        c = getchar();
    }
}
```

Notiamo che la condizione di permanenza del ciclo dipende dalla differenza con il valore `EOF`. Si tratta di un valore restituito da C quando arriva alla fine del file. Ricordiamo che un file di testo corrisponde ad un certo numero di caratteri salvati in memoria; quando ho scorso tutti i caratteri, C segnala la cosa restituendo `EOF`.

Scriviamo una versione piu' compatta della funzione nel seguente modo:

```
#include <stdio.h>
int main()
{
    int c;

    while( (c = getchar()) != EOF)
        putchar(c);
}
```

Con l'espressione `(c = getchar()) != EOF` eseguiamo **due** operazioni:

1. Assegniamo il valore da getchar() in c
2. Il valore assegnato a c viene confrontato con EOF

NB: per salvare dei valori di tipo **char** utilizziamo una variabile di tipo **intero**; nei prossimi tutorati capiremo perche' questo tipo di operazioni e' lecito.

3.2 Nuova riga

Ora che sappiamo come ottenere caratteri dobbiamo capire come spezzare il flusso in input in righe Supponiamo di avere un file con il seguente testo formato da 4 righe.

```
ciao
sono un
testo di
4 righe
```

In realta' il testo viene salvato in **memoria** nel seguente modo:

`ciao\nsono un\ntesto di\n4 righe`

Ossia esiste un carattere speciale `\n` che rappresenta la nuova riga.

Quindi possiamo mettere insieme quanto visto fino ad esso per implementare la funzione. Scorriamo l'input con `getchar` e salviamo i caratteri in un array. Appena incontriamo un carattere di tipo `\n` interrompiamo la procedura.

3.3 Implementazione

La funzione get_line ha due parametri in input: - Un array di char - ossia un puntatore - Un intero che rappresenta il massimo numero di caratteri

La funzione ha il seguente comporatamento: - modifica l'array in input aggiungendo caratteri - restituisce la lunghezza della linea

```
int get_line(char s[], int lim){

    int c, i;

    i = 0;
    while(i < lim -1){
        c = getchar();

        if(c == EOF){
            break;
        }
        else if (c == '\n'){
            s[i] = '\n';
            i++;
            break;
        } else {
            s[i] = c;
        }
            i++;
    }

    s[i] = '\0';
    return i;

}
```

La funzione quindi scorre l'input e continua a salvare i valori all'interno del array di char. Ricordiamoci che un array di char ha come ultimo elemento il null byte `\0`, che inseriamo come ultima operazione.

Inizialmente possiamo pensare a due casi: 1. La riga in input e' piu' grande dell'array di char 2. La riga in input e' minore o uguale dell'array di char

Nel primo caso, continuiamo ad inserire valori fino a quando il valore di i non raggiunge lim e inseriamo il null byte alla fine.

Nel secondo caso invece la condizione di uscita e' determinata dal corpo del ciclo while. Abbiamo quindi due casi per l'uscita.

1. Raggiungiamo una nuova riga
2. Raggiungiamo `EOF`

Se ragginguiamo una nuova riga dobbiamo ricordardi di incrementare il valore di i e assegnare manualmente il valore di *s[i]*.

4 Reverse text

Ora possiamo combiare le due funzione e ottere il programma per la reverse del testo.

```
#include <stdio.h>
#define MAXLINE 1000

int main()
{
    int len;
    char line[MAXLINE];

    while( (len = get_line(line, MAXLINE)) > 0){
        reverse_line(line, len);
        printf("%s",line);
    }

    return 0;
}
```

La funzione e' implementata nel seguente modo:

1. La funzione `get_line` inserisce caratteri dallo standard input e restituisce la lunghezza nella variabile len.
2. Il valore len viene controllato, se maggiore di zero vuol dire che ci sono caratteri nella riga.
3. La riga viene invertita e stampata
4. Il while riprende

Possiamo sottolineare che lo standard input viene consumtato in blocchi diversi. Se prendiamo il testo

```
ciao\nsono un\ntesto di\n4 righe
```

Nella prima iterazione, la funzione `get_line` prende la porzione di array `ciao\n` e poi passa l'esecuzione al corpo del ciclo while. Nella seguente interzione `get_line` ripartira' a partire dal sesto carattere.

5 Standard input

Per fornire un input al programma possiamo usare la funzione di *input redirection* fornita dalla shell.

```
gcc -o reverse_text reverse_text.c
./reverse_text < myinput.txt
```

In questo modo forniamo come input al programma il contenuto di myinput, senza doverlo inserire manualmente.

6 Esercizi aggiuntivi

1. Scrivere una funzione line_count che dato un file di testo conta il numero di righe
2. Scrivere una funzione line_length che data una stringa di testo conta il numero di caratteri
3. Scrivere un programma che stampa il valore di `EOF`