

Lezione 10

Definire le funzioni seguenti (esercizio del lucido n. 7):

```
/* getline: legge e carica in s una linea, ritorna la lunghezza */

int getline(char s[], int lim) {
    int i=0;
    char c;

    while((c=getchar())!='\n' && c!=EOF && i<lim-1)
        s[i++]=c;

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

/* copy: copia from in to; assume che to sia sufficientemente ampio */
void copy(char to[], char from[]) {
    int i=0;

    while(from[i]!='\0')
        to[i]=from[i++];

    to[i]='\0';
}
```

Le versioni di `getline` e `copy` che fanno uso di dichiarazioni globali sono definibili come segue (esercizio del lucido n. 9):

```
#define MAXLINE 1000

char line[MAXLINE];
char longest[MAXLINE];

int getline(void) {
    int i=0;
    char c;
    extern char line[MAXLINE];

    while((c=getchar())!='\n' && c!=EOF && i<MAXLINE-1)
        line[i++]=c;

    line[i]='\0';
    return i;
}

void copy(void) {
    int i=0;
    extern char line[MAXLINE];
    extern char longest[MAXLINE];
```

```

while(line[i]!='\0')
    longest[i]=line[i++];

longest[i]='\0';
}

```

Scrivete le seguenti funzioni:

1. `reverse(s)`, che inverte la stringa di caratteri `s`. Usate tale funzione per scrivere un programma che inverte le linee di un testo in input.

```

#include <stdio.h>
#include <string.h>

#define MAXLENGTH 256

void reverse(char s[]) {
    int i,l=strlen(s);
    char c;

    for(i=0;i<l/2;i++) {
        c=s[i];
        s[i]=s[l-(i+1)];
        s[l-(i+1)]=c;
    }
}

main() {
    char c,linea[MAXLENGTH];
    int i=0;

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

        if(c=='\n') {
            linea[i]='\0';
            i=0;
            reverse(linea);
            printf("%s\n",linea);
        }
        else
            linea[i++]=c;

    }
}

```

2. `strindex(s,t)` che restituisce la posizione, cioè l'indice di inizio della stringa `t` nel vettore `s`, oppure `-1` se `t` non compare in `s`.

```

int strindex(char s[],char t[]) {
    int i=0,j,index=-1;

    if(strlen(s)<strlen(t))
        return -1;

    while(s[i]!='\0' && index==-1) {

        while(s[i]!=*t && s[i]!='\0')
            i++;

        j=1;

        while(s[i+j]==t[j] && s[i+j]!='\0' && t[j]!='\0')
            j++;

        index=t[j]=='\0' ? i : -1;
    }

    return index;
}

```

3. `strrindex(s,t)` che restituisce la posizione dell'occorrenza più a destra di t in s , oppure -1 se t non compare in s .

```

int strrindex(char s[],char t[]) {
    int i=0,j,index=-1;

    if(strlen(s)<strlen(t))
        return -1;

    while(s[i]!='\0') {

        while(s[i]!=t[0] && s[i]!='\0')
            i++;

        j=1;

        while(s[i+j]==t[j] && s[i+j]!='\0' && t[j]!='\0')
            j++;

        if(t[j]=='\0')
            index=i;

        i++;
    }

    return index;
}

```