Laboratorio di Elementi di Architetture e Sistemi Operativi

Esercizi dell'11 Aprile 2012

Esercizio 1. *Modificare l'Esercizio 4 della lezione scorsa (la calcolatrice) in modo che chieda all'utente se vuole fare un'altra operazione.*

Esistono vari modi in cui si può risolvere l'esercizio senza creare i problemi visti a lezione. Il più semplice è quello di aggiungere una chiamata a `putchar` per eliminare il carattere `'\n'` spurio che rimane nel buffer di tasitera:

```
#include <stdio.h>

main()
{
  int a, b;
  char op;

  do {
    printf("Inserire l'operazione da eseguire: ");
    scanf("%d %c %d", &a, &op, &b);
    switch(op) {
        case '+':
            printf("%d + %d = %d\n", a, b, a + b);
            break;
        case '-':
            printf("%d - %d = %d\n", a, b, a - b);
            break;
        case '*':
            printf("%d * %d = %d\n", a, b, a * b);
            break;
        case '/':
            printf("%d / %d = %d\n", a, b, a / b);
            break;
        default:
            printf("Operazione sconosciuta!\n");
            break;
    }
    getchar();
    printf("Eseguire una nuova operazione? (s/n) ");
    op = getchar();
  } while (op == 's' || op == 'S');
}
```

Questa soluzione, tuttavia, non risolve tutti i problemi. In particolare, può non comportarsi correttamente quando l'utente inserisce un input che non rappresenta un espressione. Per poter gestire correttamente anche questi casi, è necessario cambiare il modo con cui si legge l'input dell'utente, usando `gets` invece di `scanf`:

```
#include <stdio.h>

main()
{
  int a, b;
  char s[80];
```

```
  char op;

  do {
    printf("Inserire l'operazione da eseguire: ");
    gets(s);
    sscanf(s, "%d %c %d", &a, &op, &b);
    switch(op) {
        case '+':
            printf("%d + %d = %d\n", a, b, a + b);
            break;
        case '-':
            printf("%d - %d = %d\n", a, b, a - b);
            break;
        case '*':
            printf("%d * %d = %d\n", a, b, a * b);
            break;
        case '/':
            printf("%d / %d = %d\n", a, b, a / b);
            break;
        default:
            printf("Operazione sconosciuta!\n");
            break;
    }
    printf("Eseguire una nuova operazione? (s/n) ");
    gets(s);
  } while (s[0] == 's' || s[0] == 'S');

}
```

Esercizio 2. *Scrivere un programma C che trasferisca l'input sull'output sopprimendo le duplicazioni contigue di righe.*

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

main()
{
  char s[100], t[100];
  char *res;

  res = gets(s);
  if(res != NULL)
    puts(s);
  res = gets(t);

  while(res != NULL) {
    if(strcmp(s,t) != 0)
      puts(t);
    strcpy(s,t);
    res = gets(t);
  }
}
```

Esercizio 3. *Realizzare un programma C che stampi il numero di occorrenze di ogni lettera dell'alfabeto (maiuscola o minuscola) presente nel suo input.*

```c
#include <stdio.h>

main()
{
  int f[26];
  int c;

  for(c = 0; c < 26; c++)
    f[c] = 0;

  for(c = getchar(); c != EOF; c = getchar()) {
    c = tolower(c);
    if(c >= 'a' && c <= 'z')
      f[c-'a']++;
  }

  for(c = 'a'; c <= 'z'; c++)
    printf("%c: %d\n", c, f[c-'a']);
}
```

Esercizio 4. *I commenti in C possono essere di due tipi:*

- *commenti su una sola linea, preceduti da //*
- *commenti su più linee, racchiusi tra /* e */*

Scrivere un programma C per rimuovere tutti i commenti da un programma C. Fare attenzione alla corretta gestione delle stringhe fra apici singoli e doppi. I commenti, in C, non possono essere nidificati. Il programma è un filtro: legge l'input dallo standard input e produce il risultato sullo standard output.

```c
/*
 * decommenta.c
 * ------------------------------------------
 * programma che toglie i commenti dal codice C
 */

#include <stdio.h>

main()
{
  // stringa per testare la corretta gestione delle stringhe
  char str[] = "stringa con /* commenti */, // e caratteri di escape \" \' \\";
  int c1, c2, stampa;

  c2 = getchar();
  while(c2 != EOF) {
    c1 = c2;
    c2 = getchar();
    if(c1 == '/' && c2 == '/') {
      // commento su linea singola
      do{
        c1 = c2;
        c2 = getchar();
      } while(c2 != '\n' && c2 != EOF);
    } else if(c1 == '/' && c2 == '*') {
      // commento multilinea
      do{
        c1 = c2;
        c2 = getchar();
```

```
      } while (c2 != EOF && !(c1 == '*' && c2 == '/'));
      if(c2 != EOF) c2 = getchar();
    } else if(c2 == '"') {
      // costante stringa
      do {
        putchar(c1);
        c1 = c2;
        c2 = getchar();
        if(c1 == '\\' && c2 != EOF){
          // carattere di escape, scrivo e leggo un altro carattere
          putchar(c1);
          c1 = c2;
          c2 = getchar();
        }
      } while(c2 != EOF && c2 != '"');
      putchar(c1);
    } else if(c2 == '\'') {
      // costante carattere
      do {
        putchar(c1);
        c1 = c2;
        c2 = getchar();
        if(c1 == '\\' && c2 != EOF){
          // carattere di escape, scrivo e leggo un altro carattere
          putchar(c1);
          c1 = c2;
          c2 = getchar();
        }
      } while(c2 != EOF && c2 != '\'');
      putchar(c1);
    } else {
      // codice C normale
      putchar(c1);
    }
  }
}
```

Esempio di esecuzione del programma: `./decommenta < decommenta.c`