Laboratorio di Elementi di Architetture e Sistemi Operativi

Soluzioni del Compitino del 3 Aprile 2013

Esercizio 1.

- Scrivere una funzione con il seguente prototipo:
  `int fattoriale(int n)`
  che calcoli il fattoriale di un numero `n`.
- Scrivere un programma che prenda in input un numero intero e ne calcoli il fattoriale.

```
#include <stdio.h>

int fattoriale(int n) {
  if(n <= 1) return 1;
  return n * fattoriale(n-1);
}

int main() {
  int n;

  printf("Inserire un numero intero: ");
  scanf("%d", &n);
  printf("Il fattoriale di %d e' %d\n", n, fattoriale(n));

  return 0;
}
```

Esercizio 2.

- Scrivere una funzione con il seguente prototipo:
  `void ribalta(char s1[], char s2[])}`
  che rovescia il testo contenuto nella stringa `s2`, salvando il risultato nella stringa `s1`.
- Scrivere un programma C che utilizzi la funzione `ribalta` per eseguire le seguenti operazioni:
  1. leggere una riga di testo (di massimo 100 caratteri) dallo standard input;
  2. stampare sullo standard output la riga scrivendola alla rovescia;
  3. ripetere le operazioni finché non si legge una riga vuota.

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

void ribalta(char s1[], char s2[]){
  int i=strlen(s2)-1;
  int j;

  for(j = 0; i >= 0; i--, j++){
    s1[j]=s2[i];
  }
  s1[j]='\0';
}

int main(){
  char s1[101], s2[101];
  char *res;

  res=gets(s2);
```

```
  while(res != NULL && strlen(s2) > 0){
    ribalta(s1,s2);
    puts(s1);
    res=gets(s2);
  }
}
```

Esercizio 3.

- Realizzare un insieme di funzioni per gestire una pila (stack) di caratteri. In particolare, si implementino le operazioni di inserimento (`push`) ed estrazione (`pop`) di un valore nella pila, ed il controllo di pila vuota (`isempty`). Si assuma che la pila possa contenere al massimo 80 valori.
- Utilizzare le funzioni di gestione della pila per controllare il bilanciamento delle parentesi tonde, quadre e graffe di una stringa di testo. Il programma deve:
  1. leggere una riga di testo dallo standard input (massimo 80 caratteri);
  2. scorrere la riga ed operare come segue:
     - se il carattere corrente è una parentesi aperta (`(, [, {`) inserirlo in cima alla pila e proseguire con il carattere successivo;
     - se il carattere corrente è una parentesi chiusa (`), ], }`), estrarre il carattere in cima alla pila e controllare che sia una parentesi aperta *dello stesso tipo*. In caso negativo, il bilanciamento delle parentesi non è corretto ed il programma termina. In caso positivo si continua a scorrere la stringa;
     - se il carattere non è una parentesi lo si ignora e si prosegue con il carattere successivo;
  3. terminato il controllo della stringa, il programma controlla se la pila è vuota. In caso positivo, il bilanciamento delle parentesi è corretto. In caso negativo, il bilanciamento delle parentesi non è corretto.

```
#include <stdio.h>

int isempty();
void push(char c);
int pop();
int controllo(char s[]);

int main(){
  char s[81];
  char *res;

  printf("Inserire una riga di testo: ");
  res=gets(s);
  if(res == NULL) {
    printf("Errore nella lettura dell'input!\n");
    return 2;
  }

  if(controllo(s)) {
    printf("Le parentesi sono bilanciate correttamente.\n");
    return 0;
  }
  printf("Le parentesi non sono bilanciate correttamente.\n");
  return 1;
}

// Gestione dello stack
float stack[80];
int cima = -1;
```

```
int isempty() {
  return (cima < 0);
}

void push(char c) {
  cima++;
  stack[cima] = c;
}

int pop() {
  if(cima >= 0) {
    cima--;
    return stack[cima+1];
  } else {
    return EOF;
  }
}

int controllo(char s[]) {
  int i, c;
  for(i = 0; s[i] != '\0'; i++) {
    if(s[i] == '(' || s[i] == '[' || s[i] == '{') {
      push(s[i]);
    }
    if(s[i] == ')') {
      c = pop();
      if(c != '(') return 0;
    }
    if(s[i] == ']') {
      c = pop();
      if(c != '[') return 0;
    }
    if(s[i] == '}') {
      c = pop();
      if(c != '{') return 0;
    }
  }
  return isempty();
}
```