Fondamenti di Informatica T-1 modulo 2

Laboratorio 04: stringhe

Stringhe

- In C le stringhe ben formate sono in realtà ***array di caratteri terminati sempre da un carattere speciale, '\0',*** detto anche "terminatore di stringa"
- Le stringhe possono quindi essere gestite:
  - come array di caratteri (quindi accesso a basso livello)
  - come ADT stringa, con l'aiuto delle funzioni della libreria **`string.h`**

Esempio 1 Stringhe come ADT

Stampa di numeri reali con “dettaglio” a piacimento

- ***Si realizzi una funzione stampaDettagli(…) che riceva come parametri un numero reale e due interi (che indicano, rispettivamente, il numero di cifre per la parte intera e per la parte decimale)***
- ***La funzione stampi a video il numero secondo le indicazioni ricevute come parametri***
- Come? Tramite una opportuna stringa di formato (es. “%6.2f” significa stampare un float con 6 cifre per la parte intera e due per la parte decimale)
- La funzione componga dinamicamente una opportuna stringa di formato, utilizzando la funzione sprintf(…) ed una stringa allocata staticamente (si controlli di non eccedere la dimensione della stringa già allocata)

Esempio 1 - Note

- Pseudo algoritmo:
  - Creo una stringa di dimensione prefissata MAX, dove comporrò il formato
  - Calcolo quanti caratteri sono necessari per comporre la stringa di formato
  - Se ho “spazio a sufficienza”, con sprintf() scrivo nella stringa di formato il formato che desidero
  - Altrimenti restituisco un apposito codice di errore

Esempio 1 - Soluzione (1)

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


#define TRUE    1
#define FALSE   0
#define BOOLEAN int;

#define MAX_FORMATO 128
#define RESULT_OK 1
#define RESULT_TOO_MANY_CHARS -1
#define RESULT int;
```


Esempio 1 - Soluzione (2)

```
RESULT stampaDettagli(      float num,
                int cifreParteIntera,
                int cifreParteDecimale)
{
  char stringaFormato[MAX_FORMATO];
  int nCarInt, nCarDec;

  nCarInt = ((int) log10(cifreParteIntera)) + 1;
  nCarDec = ((int) log10(cifreParteDecimale)) + 1;

  if ((nCarInt + nCarDec + 6) >= MAX_FORMATO)
      return RESULT_TOO_MANY_CHARS;
  …
```

Esempio 1 - Soluzione (3)

```
    else
    {
        sprintf(    stringaFormato,
                    "%%%d.%df\\n",
                    cifreParteIntera,
                    cifreParteDecimale);
        printf("Formato: %s\n",stringaFormato);
        printf(stringaFormato, num);
    }
    return RESULT_OK;
}
```


Esempio 1 - Soluzione (4)

```
void handleError(RESULT result, char *str)
{
   switch (result)
   {
       case RESULT_OK:
             strcpy(str, “”);
             break;
       case RESULT_TOO_MANY_CHARS:
             strcpy(str, "Stringa di formato con troppi caratteri\n");
             break;
       default:
             strcpy(str, ”Risultato non gestito.\n");
   }
}
```

Esempio 1 - Soluzione (4)

```
int main(void)
{
  RESULT result;
  char resultStr[MAX_FORMATO];
  result = stampaDettagli(2.0, 3, 8);

  if (result!=RESULT_OK)
     handleError(result, resultStr);
  printf("errore %s\n", resultStr);
     return (0);
}
```


Esercizio 1

Composizione del nome

- Al fine di stampare degli indirizzi su delle buste, è necessario comporre la prima parte dell'indirizzo come "Cognome Nome" o "Cognome N."
- Si realizzi una funzione che riceva come parametri:
  - cognome
  - nome
  - una stringa in cui andare a inserire l'indirizzo
  - lunghezza massima della stringa indirizzo
- La funzione deve copiare/concatenare nell'indirizzo il cognome seguito dal nome, avendo cura di rispettare le dimensioni della stringa indirizzo
- Qualora la stringa indirizzo sia troppo piccola per contenere entrambi, la funzione provi a comporre la stringa come "Cognome N."
- Qualora neanche ciò sia possibile, la funzione ritorni un codice di errore opportuno

Esercizio 1 - Note

- Si usino:
  - **`strlen()`** per determinare la lunghezza di una stringa
  - **`strcat()`** o **`strcpy()`** per comporre in indirizzo la nuova stringa
- Si provi a implementare una variante ricorsiva di **`strlen`** (senza appoggiarsi sulla libreria **`string.h`**)

Esercizio 2: array di caratteri

Codice segreto nella pagina di un libro

- Sono date due stringhe, una denominata msg e una denominata secret (non più lunga di msg) di caratteri tutti minuscoli
- Si vuole sapere se tutti i caratteri di secret sono contenuti nello stesso ordine (ma eventualmente intervallati da altri caratteri) nella stringa msg
- In caso positivo, il programma deve restituire una copia del msg originale, dove però ad ogni lettera riconosciuta come facente parte di secret viene sostituita la lettera maiuscola
  - Es: msg = “ma che bel castello”, secret = “cestello”
  - Risultato: **SI** e stringa **“ma ChE bel caSTELLO”**

Esercizio 3: calcolo modulo 2

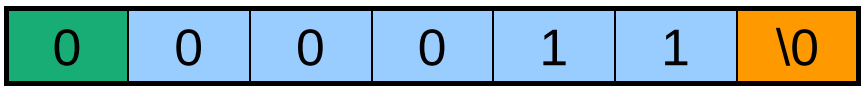
Conversione di numeri interi in rappresentazione a modulo 2 e operazioni

- Si realizzi un programma capace di
  - Effettuare la conversione da numeri interi (con segno!) alla corrispondente rappresentazione in modulo 2 e viceversa
  - Effettuare la somma e sottrazione di due numeri interi utilizzando la rappresentazione in modulo 2
- Si utilizzino le stringhe per la rappresentazione in modulo 2

Esercizio 3 - Requisiti

In particolare, si definiscano le seguenti funzioni

- **`int convertiBin(char* bin);`**
  - Ottenere la codifica binaria del valore assoluto e convertire quella
- **`RESULT convertiInt(int value, char* res);`**
  - Dove RESULT può valere
    - OK se la conversione è stata effettuata con successo
    - SIZE_OVERFLOW se la dimensione fisica di res non è abbastanza grande per contenere la conversione
      - » si utilizzi il logaritmo in base 2 per calcolare lo spazio necessario, e si considerino segno e terminatore
- Facciamo in modo che le stringhe siano riempite sempre per tutta la dimensione fisica
  - Es: se la dimensione fisica è 7, la rappresentazione del numero 3 dovrà essere



Esercizio 3 - Requisiti (2)

- **`void invertiSegno(char *num, char *res);`**
  - Inverte il segno di num (ovvero realizza l'algoritmo del modulo 2)
    - PASSO 1: inversione dei bit
    - PASSO 2: aggiunta di uno
      - » Suggerimento: si converta l'intero 1 in binario e si utilizzi la funzione sum (vedi sotto)
- **`RESULT sum(char *add1, char *add2, char *res);`**
  - Somma i due numeri binari (controllare che abbiano la stessa lunghezza)
  - Restituisce SIZE_OVERFLOW se l'operazione causa overflow (x-or dei due carry più significativi)
- NOTA: si faccia in modo che le funzioni calcolino un risultato corretto anche passando come risultato una delle variabili date in input
  - Es: **`invertiSegno(num, num)`** deve invertire correttamente il segno di **`num`** (dopo l'invocazione, **`num`** contiene il risultato)

Esercizio 3 - Testing

- Si includa il modulo di test fornito sul sito Web del corso
  - main deve semplicemente includere header file e invocare la funzione **`test()`**
- Nota sul test

Il caso '-' si riconduce al caso '+' invertendo il segno del secondo operando

```
switch(operatore)
{
    case '-':
          invertiSegno(str2, str2);
          printf("%s + %s = ", str1, str2);
    case '+':
          resOpe = sum(str1, str2, strRes);
          …
```