Laurea in Ingegneria dell'Informazione

Esercitazioni Guidate di Tecniche della Programmazione

Note introduttive:

1) Nel titolo di ogni sezione di questo documento è specificato tra parentesi il nome del (o dei) file in cui è proposta una soluzione (se disponibile nella directory "`programmi`" di questa EG).
2) I programmi che scriveremo dovranno essere in accordo con la definizione standard **ANSI C** del linguaggio C.
    a. Se usate un sistema diverso dal DEV, provvedete a che la compilazione avvenga con il compilatore standard C.
    b. Ricordate che un programma C e' in un file con estensione "`.c`"
    c. Se usate il DEVC++, per configurare bene la compilazione bisogna
        i. andare nel menù "Tools", selezionare "Compiler Options", scegliere "Settings" e poi "C Compiler"; poi selezionare almeno "Support all ANSI Standard C Programs")
        ii. (se l'interfaccia è in italiano …) andare nel menu' "Strumenti", selezionare "Opzioni di compilazione", "Compilatore", "Generazione di Codice …", "Compilatore C" e poi far apparire "Yes" almeno accanto a "Supporto programmi ANSI standard C.

NB Spesso ci sono svariati esercizi proposti nelle slide delle lezioni; questi esercizi potrebbero essere affrontati durante la EG, anche se non vi compaiono. A meno che non li abbiate già fatti …. Se li avete affrontati potete sempre farmeli vedere per chiarire eventuali dubbi. Se non li avete affrontati provate a farli …

9. Esercitazione 9 - ricorsione

9.1. La funzione fattoriale(`FATT.C`)

Scrivere una funzione **`int fatt(int)`** che, ricevendo un numero intero `n` calcola e restituisce il valore di **`n!`**.

Scrivere inoltre un programma capace di testare questa funzione e verificare che, usando nella definizione della funzione il tipo **`int`** (per parametro e valore ritornato) la funzione produce risultati troppo grandi anche con numeri relativamente piccoli. (Fino a che valore di **`n`** si riesce ad arrivare senza andare in *overflow*?)

9.2. La funzione fattoriale (`FATTLONG.C`)

Ripetere l'esercizio precedente, ma usando il tipo **`long int`** per il risultato della funzione. Verificare che questa funzione permette di fare calcoli su numeri (un po') più grandi rispetto alla precedente.

9.3. La funzione LeggiEInvertiInput() (`INVINP.C`)

Scrivere una funzione che esegue la lettura di una sequenza di caratteri fornita da input e terminata dal carattere '.', producendo in output una stampa dei medesimi caratteri (escluso il punto) in ordine inverso rispetto a quello con ui sono stati inseriti in input. Ad esempio, se l'input fosse TOPI. l'output dovrebbe essere IPOT

Si vuole che la funzione implementi un algoritmo ricorsivo per risolvere il problema. Provare la funzione con un opportuno programma.

…

Suggerimento:

L'idea di quanto visto a lezione era la seguente: l'algoritmo da realizzare deve

- leggere il primo carattere che si presenta in input e memorizzarlo in una variabile **`ch`**; (ad esempio T viene messo in **`ch`**)
- chiedere che il resto dell'input venga letto e stampato al contrario
- se l'operazione indicata al punto precedente è stata realizzata, ora in output ci sono, stampati in ordine inverso, i caratteri che erano forniti in input dopo quello che abbiamo memorizzato in **`ch`**

ad esempio ora in output c'è **IPO**

quindi è sufficiente stampare il carattere attualmente in **`ch`** e l'output assume la forma richiesta

IPO**T**

(seguono altri suggerimenti)

Suggerimento 2 di 3:

Chiamiamo **`LEGGIEINVERTI`** la sequenza di operazioni che vogliamo implementare (insomma, l'algoritmo ...)

Lo schema base da realizzare e' il seguente:

- lettura di un carattere in **`ch`**
- **`LEGGIEINVERTI`**
  (così viene letto e invertito il resto dell'input (quello dopo il carattere memorizzato in **`ch`**)
- stampa **`ch`**

Suggerimento segue

Suggerimento 3:

Ovviamente se il carattere letto e memorizzato in **`ch`** durante l'esecuzione di **`LEGGIEINVERTI`** e' **`'.'`**, non deve essere richiesto di leggere e invertire "il resto dell'input" (non c'e'!), ne' di stampare il carattere **`ch`**.
Queste due operazioni vanno eseguite solo se il carattere letto è diverso da **`'.'`**.

Quindi il caso in cui **`ch!='.'`** è un CASO RICORSIVO, in cui l'algoritmo **`LEGGIEINVERTI`** decide di eseguire se stesso sul resto dell'input e poi stampare **`ch`**.

Invece il caso in cui **`ch=='.'`** è un CASO BASE (in cui non c'è attivazione ricorsiva e, in questo caso particolare, non bisogna fare nulla).

- **`lettura ch`**
- **`se ch!='.'`**
  - **`LEGGIEINVERTI`**
  - **`stampa ch`**

  `(altrimenti niente)`

Poi segue il prossimo esercizio

9.4. La funzione `palin()` *(*`PALIN1.C`*)*

Scrivere una funzione **`palin()`** che, ricevendo una parola (stringa di caratteri) e due indici interi, **`i`**, **`j`**, restituisca 1 se la porzione di parola compresa tra i caratteri di indice **`i`** e **`j`** è palindroma.

Provare la funzione con un opportuno programma.

Attenzione, segue suggerimento …

Suggerimento:

Una parola è palindroma se è leggibile indifferentemente da sinistra a destra o da destra a sinistra (anilina, asor-rosa, ingegni, anna, ada, radar, kayak, level, rotor, civic).

Attenzione, segue suggerimento …

Suggerimento 2 di 5:

Sia p la parola (un array di N caratteri, occupato dal carattere di indice 0 al carattere di indice `k` (con **`strlen(p)`** uguale a **`k+1`** e **`p[strlen(p)]=='\0'`** e quindi escluso dalle nostre considerazioni).

La porzione **`p[0]---p[k]`** (cioe' la stringa)

- non è palindroma se **`p[0]!=p[k]`**
- **potrebbe** essere palindroma se **`p[0]==p[k]`**(e in tal caso sarebbe effettivamente palindroma **solo se** fosse palindroma la porzione **`p[1]---p[k-1]`**)

Attenzione, segue suggerimento …

Suggerimento 3 di 5:

Dal suggerimento precedente viene uno schema come il seguente

`palin(p, i, j)` è (ritorna come risultato)

- 0 se **`p[0]!=p[k]`**
- il valore di `palin(p, i+1, j-1)` se **`p[0]==p[k]`**

Segue suggerimento

Suggerimento 4 di 5:

Lo schema visto al punto precedente vale se i<j;
infatti,

- se **`i==j`** non c'è nulla da controllare! E se siamo arrivati ad eseguire la valutazione di **`palin(p,i,j)`** con **`i==j`** vuol dire che tutte le coppie **`p[i], p[j]`** con i valori precedenti di **`i`** e **`j`** erano a caratteri uguali. A questo punto la stringa si rivela palindroma e possiamo restituire 1 senza ulteriori attivazioni ricorsive;

- se **`i>j`**, vale un discorso analogo: tutte le coppie **`p[i], p[j]`** con i valori precedenti di **`i`** e **`j`** (cioè con valori per cui **`i<j`**) erano a caratteri uguali. Continuando a fare verifiche su coppie di caratteri **`p[i], p[j]`** con **`i > j`** controlleremmo coppie già controllate (e che erano uguali). Quindi anche in questo caso la stringa si è rivelata palindroma e possiamo restituire 1 senza ulteriori attivazioni ricorsive.

Segue suggerimento

Suggerimento 5:

Dal suggerimento precedente viene uno schema come il seguente

`palin(p, i, j)` e' (ritorna come risultato)

- **`se i<j`**
  - **`0`** se **`p[0]!=p[k]`**
  - **`palin(p, i+1, j-1)`** se **`p[0]==p[k]`**

- **`altrimenti 1`**

9.5. cosa fa?

Osservare il seguente programma:

- la **`main()`** contiene due errori, facilmente evidenziabili. Quali sono?
- cosa fa la funzione **`cosaFa()`** e, quindi, cosa fa il programma, una volta corretto? Giustificare la risposta **e poi** vedere se è giusta, eseguendo il programma.

```
#include <stdio.h>
#define MAX 6

int cosaFa(int a[], int k, int dim) {
     if (k < dim)
          return (a[k] + cosaFa(a, k+1, dim));
     else return 0;
}

int main(){
  int io[MAX], j;
     printf("\nInserire le %d cifre del numero di matricola con
                          (uno spazio tra l'una e l'altra): ", MAX);

     for(j=0; j<=MAX; j++)
          scanf("%d",io+j);

     printf("ecco qua: %d\n", cosaFa(io,2,MAX));

printf("\nFINE\n");
return 0;
}
```

9.6. mahattama sāmānya vibhājaka

Qui ci occupiamo del massimo comun divisore, una volta per tutte.

Nella lezione sulla ricorsione abbiamo richiamato i metodi di Euclide (per sottrazione e per divisione) e abbiamo visto come si puo` interpretare il secondo metodo come algoritmo ricorsivo.

Scrivere un programma in cui

1) Vengano confrontate le esecuzioni dei due algoritmi sopra menzionati, nella versione nota da tempo, iterativa. Quale dei due sembra più efficiente?

2) Venga realizzata l'implementazione ricorsiva dell'algoritmo "per divisioni"

3) Venga realizzata, alla fine, anche una versione ricorsiva, ottenuta da quella precedente, in cui viene usata una espressione condizionale.

Seguire i passi elencati qui sotto.

9.6.1. Confronto tra i due algoritmi

Scrivere il programma in modo che, dopo la lettura dei due numeri, **`num1`** e **`num2`**, vengano mostrati i valori del MCD calcolati dalle due funzioni, opportunamente chiamate: una funzione realizzerebbe il primo algoritmo, l'altra l'altro …

Entrambe le chiamate avranno come parametri attuali i due numeri richiesti dall'utente.

Suggerimento segue

Suggerimento

Nelle funzioni aggiungere una parte in cui viene stampata qualche informazione che permetta di capire su cosa sta lavorando la funzione in quel momento.

L'idea è che, confrontando gli output delle due funzioni, ci si possa rendere conto di quanti passaggi vengono eseguiti dall'una e dall'altra. Dovrebbe apparire evidente che un metodo raggiunge il risultato in molti meno passi dell'altro.

(Ora, è giusto che una divisione costa più di una sottrazione, ma questa differenza impallidisce se pensiamo a quanto grandi e distanti potrebbero essere n ed m. Se per esempio n fosse un milione ed m 10000, potrebbero esserci 100 sottrazioni contro una dozzina di divisioni … Il numero di sottrazioni cresce linearmente rispetto al rapporto n/m, mentre il numero di divisioni necessarie cresce mooooolto più lentamente. Sì, è più efficiente l'algoritmo per divisioni).

Suggerimento su cosa stampare segue …

Suggerimento
Beh, dato che i due algoritmi modificano ciclicamente i valori di **`n`** e di **`m`**, e immaginando che le funzioni facciano la medesima cosa …
Si potrebbero stampare i valori assunti da **`n`** ed **`m`**, volta per volta.
Nella soluzione proposta si fa proprio così (il codice relativo è però commentato).

9.6.2. MCD ricorsivo

Seguire le linee proposte nella lezione svolta su questi argomenti.

Il programma sarà una estensione del precedente. Solo che ora ci sarà anche la chiamata ad una funzione ricorsiva, **`mcdEuclideRic()`**, che riceve due numeri interi, **`n`** ed **`m`**.

9.6.3. MCD ricorsivo, con l'espressione condizionale

Aggiungere nel programma precedente una seconda versione della funzione **`mcdEuclideRic()`**, che però usa una singola espressione condizionale per restituire il risultato.

Basta fissare a lungo l'algoritmo (vedi lezione) e poi il codice della funzione **`mcdEuclideRic()`** precedente per capire che dall'**`if`** che in pratica costituisce tutta la funzione, si può ottenere una riscrittura in forma di espressione condizionale.

La soluzione completa è nel file **`mcd_risorsivo.c`**.

9.7. Boom Frequency in una centrale nucleare (`speriamoBene.c`)

In una centrale nucleare particolarmente "ottimizzata", le pompe di refrigerazione del nocciolo non sono esattamente il massimo dell'affidabilità.

Ogni **`DP`** minuti viene effettuata una previsione delle prossime **`N`** frequenze di vibrazione (in Hz) di una delle pompe di refrigerazione, valide per i prossimi **`DP - 1`** minuti.

Le frequenze previste sono memorizzate in un array di **`N`** double, **`frequenzePompa`**, ordinate in senso crescente.

Purtroppo, per motivi di budget, la pompa è stata progettata male ed entra in risonanza se viene sollecitata ad una particolare frequenza **`FB`** (Boom Frequency). Se tra le frequenze previste compare esattamente **`FB`**, la situazione diventa antipatica.

`DP`, **`N`**, **`FB`** sono costanti (**`%define`**).

Scrivere un programma C che:

- Inizializzi o legga da input l'array
- analizzi l'array **`frequenzePompa`** per verificare se tra i prossimi valori previsti ci sarà la frequenza **`FB`**;
    - esegua questa verifica utilizzando una funzione: **`int speriamoBene()`** che riceva l'array, la frequenza da cercare, ed anche la dimensione dell'array ed usi l'algoritmo di ricerca binaria per restituire l'indice dell'elemento dell'array che contiene **`FB`**, oppure **`-1`**
- il programma principale chiamerà questa funzione, per stabilire il prossimo passo
    - stampare **`"PANIC! PANIC! PANIC! PANIC! RUN! RUN! RUN!"`**
    - oppure **`"Per ora è andata, forse ..."`**

Si può iniziare svolgendo l'esercizio con l'algoritmo di **ricerca esaustiva** …

Poi si può espandere il programma, **sostituendo la ricerca esaustiva con quella binaria**, versione iterativa.

La soluzione proposta e nel file **`speriamoBeneIterativo.c`**

La **soluzione completa**, presente nel file **`speriamoBeneRic.c`**, prevede però l'uso della **versione ricorsiva della ricerca binaria**. (E` un invito a realizzarla …)

9.8. Il Quadrato del Sator

Quello che leggerai non è un semplice resoconto: è la **trascrizione** di un documento sgualcito, ritrovato tra casse polverose nell'archivio governativo della *Direzione Misteri Perduti*, dove *Martin E'nigme*, noto archeologo avventuroso, lo ha depositato, dopo averlo trovato durante una delle sue spedizioni. Il documento proviene dalla sala più segreta di un monastero abbandonato nei pressi di Genzano.

Inchiostro scolorito, pagine macchiate dal tempo e, qua e là, da ciò che *sembra* guano di volatile.

9.8.1. Il Documento di Martin E'nigme

«Molti hanno visto il Quadrato del Sator.
Pochi lo hanno letto.
Nessuno lo ha capito *senza pagarne il prezzo.»*

Così si apre il testo, e già l'aria sembra farsi più pesante.

Il **Quadrato del Sator** è noto da secoli:

S A T O R
A R E P O
T E N E T
O P E R A
R O T A S

Esso mantiene la sua forma se letto da ogni direzione, da sopra, da sotto, da sinistra, da destra, come se il tempo stesso, davanti a queste cinque parole, fosse costretto a girare in cerchio.

Cronache e lettere di studiosi, persi nel fervore della loro stessa passione, o bramosia, descrivono il quadrato come **una serratura maligna**. E chi tenta di aprirla senza la chiave rischia di smarrirsi.

9.8.2. La lunga scia degli sconfitti

Gli archivi parlano di:

- eruditi che passarono decenni davanti a quelle cinque righe, fino a dimenticare il proprio nome;
- alchimisti che cercarono di estrarne formule fino a ridurre la propria mente a un labirinto di numeri e simboli;
- avventurieri, convinti che il quadrato fosse una mappa, poi ritrovati mesi dopo, vaganti come anime confuse, mormorando sillabe spezzate: «ro… tas… te… net…»;
- Professori persi per le vie di Latina con l'aria allucinata, la voce rotta da un lamento *"L'algoritmo … L'algoritmo"*, alla ricerca spasmodica di un sempre sfuggente passo 0.

Si mormora che **antichi templi zoroastriani** dell'altopiano iranico conservino ancora sale nascoste, dove le pareti sono incise di segni che ricordano il quadrato, tracciati quando il **Fuoco Sacro** era considerato un ponte tra i mondi. **Monasteri medievali europei**, sepolti sotto strati di restauri e silenzi, mostrano corridoi in cui ogni pietra reca una lettura diversa, frutto di monaci che vegliarono per notti intere, finendo per confondere la preghiera con la geometria del verbo. E persino nelle roccaforti d'alta montagna un tempo legate alla **setta degli Assassini**, si dice che certe camere

rupestri siano rivestite di trascrizioni febbrili, come se ogni muro fosse un tentativo fallito di arrivare al centro del segreto.

Interpretazioni infinite. Nessuna conclusiva. La maggior parte… sul margine tremante tra illuminazione e follia.

9.8.3. Il segreto impossibile

Ma Martin, durante un viaggio in Egitto, riferisce di aver consultato la misteriosa **Tabula Smaragdina"** (Tavola Smeraldina), un testo alchemico fondamentale attribuito a Ermete Trismegisto, che descrive i principi della Grande Opera (l'alchimia spirituale e materiale). Qualcuno chiama questa opera **"Tabula Aurea dell'Opera Suprema",** titolo probabilmente apocrifo, in cui l'uso dei termini "Aurea" (Oro) e "Opera Suprema" suggerisce un riferimento al raggiungimento della perfezione alchemica (la pietra filosofale e l'elisir di lunga vita).

Sì, è noto, Martin E'nigme è logorroico.

In un passaggio criptico, il testo allude a un processo che unisce:
- ciclicità,
- parola,
- trasformazione della materia.

E qui il documento si fa audace: **una complessa decrittazione del Quadrato del Sator indicherebbe le istruzioni per la trasmutazione del *guano di piccione in oro*.**

Sì, proprio così. Una pietra filosofale, su un cuscino di piume. Martin annota a margine:

«Non è elegante.
Non è poetico.
Ma se funziona, funziona.»

9.8.4. Il problema

Martin E'nigme ha bisogno di una consulenza di programmazione. E si rivolge **a te**.
Perché solo con la tua collaborazione la chiave può essere trovata.

9.8.5. L'Algoritmo di Decrittazione per Tentativi

(dettato da Martin, da implementare in C)

1) Tutto ciò che il programma stamperà a video deve essere presentato come se fosse Martin a parlare.
2) All'avvio del programma, **stampare il Quadrato del Sator** nella sua forma classica, parola per parola, una sopra all'altra.
3) Avviare un ciclo di tentativi:
    1. Chiedere all'utente di indicare una lettera del quadrato da sostituire (`daSostituire`) e una nuova lettera (`conCuiSostituire`).
    2. Sostituire *tutte* le occorrenze della lettera selezionata nel quadrato.
    3. Stampare nuovamente il quadrato risultante, e chiedere all'utente di **leggerlo ad alta voce**. *Dice Martin … «La presenza di una piccola quantità di guano di piccione vicino al dispositivo può aiutare a verificare eventuali reazioni.»* … Fai tu.

4. Controllare con una funzione se il quadrato è ancora palindromo.
    i. Se NON lo è: stampare “Comunque questo non era palindromo.”
    ii. Se lo è: stampare “Oh, questo era palindromo: ha funzionato?”
    Se l’utente risponde **no**, allora stampare:
    “... non che mi aspettassi qualcosa ...”
    iii. Mentalmente, cosi`, *en passant* … chiedersi, *“ma può essere che il risultato non sia palindromo?”*

4) Chiedere: *“Tentiamo un’altra lettera?”*
Se sì, ripetere il ciclo.
Se no, il programma termina con un commento finale di Martin.

9.8.6. Ultima nota nel documento

Scritta a mano, tremolante, forse durante una notte febbrile, passata al buio lungo un canale di scolo, e molto lunga.

«Non essere spaventato dal quadrato.
Temi semmai ciò che potresti scoprire quando smetterà di girare.»