Introduzione alla programmazione
Esercizi di Laboratorio – Parte 2+

1 Riduzione di una frazione ai minimi termini

L'obiettivo di questo esercizio è quello di scrivere una funzione che riduca ai minimi termini numeratore e denominatore di una frazione.

```
void minimi_termini(int &numeratore, int &denominatore)
{
  // da completare...
}

int main()
{
  int a, b;

  std::cout << "Dammi␣il␣numeratore:␣";
  std::cin >> a;
  std::cout << "Dammi␣il␣denominatore:␣";
  std::cin >> b;

  minimi_termini(a, b);
  std::cout << a << "/" << b << std::endl;

  return 0;
}
```

Completare la funzione `minimi_termini` implementando in C++ l'algoritmo di Euclide che calcola il massimo comun divisore di due numeri naturali m ed n:

1. se $m = 0$ ho finito, il massimo comun divisore di m e n è n;
2. se $m < n$ scambia m e n;
3. rimpiazza m con $m - n$ e torna al passo 1.

Dotare il programma di opportuni controlli per segnalare l'inserimento di "frazioni" il cui denominatore è nullo.

Questa descrizione dell'algoritmo di Euclide si presta a una realizzazione immediata tramite un ciclo `while`. È possibile implementare lo stesso algoritmo senza cicli, ma con una funzione ricorsiva?

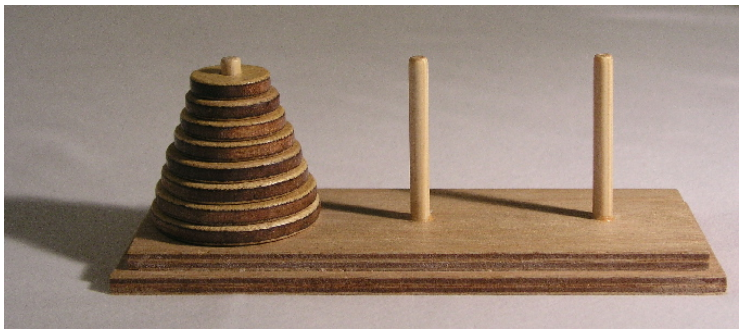

Figura 1: Configurazione iniziale del gioco delle torri di Hanoi

2 Il gioco delle torri di Hanoi

Lo scopo di questo esercizio è quello di scrivere una funzione che risolva il problema delle torri di Hanoi con n dischi (Figura 1) formulato come segue:

- **OBIETTIVO**
  - spostare tutti i dischi dal paletto A al paletto C
- **REGOLE**
  1. è consentito spostare un solo disco alla volta
  2. se due dischi sono sovrapposti, quello sottostante è più grande

La strategia per risolvere il gioco ha una naturale formulazione come algoritmo ricorsivo, tenuto conto del fatto che sul disco più grande può essere collocato uno qualsiasi degli altri dischi.

- Quando $n = 0$ la soluzione è immediata, perché non ci sono dischi da spostare!
- Quando $n > 0$ l'idea è di usare il paletto B come "magazzino temporaneo" per $n - 1$ dischi. Più precisamente:
  1. Si spostano $n - 1$ dischi dal paletto A al paletto B.
  2. Si sposta il disco più grande, ora esposto alla base del paletto A, nel paletto C (ciò è possibile perché quest'ultimo è vuoto).
  3. Si spostano $n - 1$ dischi dal paletto B al paletto C.

Usando i numeri 1, 2 e 3 per rappresentare i tre paletti, scrivere una funzione (ricorsiva) `hanoi` con tre argomenti `n`, `from` e `to` che stampi la sequenza di mosse per spostare `n` dischi dal paletto `from` al paletto `to`. **Attenzione:** il paletto temporaneo cambia a seconda di `from` (il paletto di partenza) e `to` (il paletto di destinazione).

Una volta completato, il seguente programma

```
#include <iostream>

void hanoi(int n, int from, int to)
{
  // da completare...
}

int main()
{
  int n;

  std::cout << "Quanti␣dischi?␣";
  std::cin >> n;
  hanoi(n, 1, 3);

  return 0;
}
```

dato, ad esempio, l'input 3 deve stampare

```
1 -> 3
1 -> 2
3 -> 2
1 -> 3
2 -> 1
2 -> 3
1 -> 3
```

La strategia adottata sopra esegue complessivamente $2^n - 1$ spostamenti. La dimostrazione di questo fatto segue da una semplice induzione su n:

- Quando $n = 0$, l'algoritmo esegue $2^0 - 1 = 0$ spostamenti, che è consistente con il fatto che non ci sono dischi da spostare.
- Quando $n > 0$, l'algoritmo esegue $2^{n-1} - 1$ mosse per spostare $n - 1$ dischi dal paletto di partenza a quello temporaneo (questo segue dall'ipotesi induttiva), una mossa per spostare il disco più grande dal paletto di partenza al paletto di destinazione, ed altre $2^{n-1} - 1$ mosse per spostare $n - 1$ dischi dal paletto temporaneo a quello di destinazione (anche questo segue dall'ipotesi induttiva). Complessivamente vengono effettuati $2(2^{n-1} - 1) + 1 = 2^n - 2 + 1 = 2^n - 1$ spostamenti.

Non paghi di leggere questa dimostrazione sulla carta (o sullo schermo), occorre verificarla sperimentalmente. A tal scopo:

1. Modificare il programma in modo da "contare" il numero di spostamenti effettuati da `hanoi` usando una variabile globale come già fatto per altri esercizi su funzioni ricorsive.
2. Ripetere la verifica, ma *senza* usare una variabile globale (ogni altra caratteristica del C++ nota fino a questo punto può essere utilizzata).

Sulla pagina successiva viene visualizzata la struttura delle chiamate ricorsive nel caso di n=3.

Struttura delle chiamate ricorsive e le mosse da effettuare nel caso n=3.
La chiamata hanoi(n,from,to) è indicata con h(n,from,to).
Per esempio, h(2,1,2) vuole dire “sposta due dischi dal primo paletto al secondo paletto”.