Lab Lesson 01

Giacomo Bergami

October 24, 2017

Exercises

- Scrivere due programmi, sia per byte che per integer, che sono interessati dall'overflow[1].
- Scrivere un programma che stampa i secondi in un formato leggibile. Questo significa che ogni secondo deve mostrare il numero di anni, giorni, ore, minuti e secondi. Deve essere utilizzato il seguente formato di output:

  ```
  0y 2d 3h 1m 20s
  ```

- Un anno con 366 giorni si dice che sia un anno bisestile. Dopo l'adozione del calendario gregoriano (1582), al fine di determinare se un anno è bisestile, è possibile attenersi alla seguente procedura:
  1. Se l'anno è divisibile per quattro, vedi 2) e 5) altrimenti
  2. Se l'anno è divisibile per cento, vedi 3) e 4) altrimenti
  3. Se l'anno è divisibile per quattrocento, vedi 4) e 5) altrimenti
  4. restituisci **true**
  5. restituisci **false**

[1] da Wikipedia.it: "*Un overflow aritmetico è un problema relativo alle operazioni sui numeri all'interno di un computer. Il problema è dovuto al fatto che il computer non è in grado di memorizzare qualsiasi tipo di numero, o meglio, non è in grado di memorizzare un qualsiasi numero di cifre, ma solo tante quante sono i bit a disposizione nella memoria per quel tipo di dato. Ogni bit può assumere due soli valori, 1 e 0; se abbiamo a disposizione n bit (per esempio 4), possiamo memorizzare un numero massimo pari a* $(2^n) - 1$ *(nell'esempio 1111, ovvero 15 nel sistema decimale). Se cercassimo di modificare questo numero aggiungendo 1 il numero diventerebbe 10000(16 in decimale), ma la macchina può memorizzare solo 4 bit, quindi memorizzerebbe il numero 0000 il che non corrisponde a quanto avremmo voluto. L'overflow di solito viene raggiunto quando vengono sommati 2 numeri estremamente grandi entrambi positivi e il computer ci riporta un risultato negativo (cosa impossibile!). Ugualmente nel caso stessimo lavorando con una somma di 2 numeri negativi, quindi estremamente piccoli, e il risultato fosse un numero positivo (in questo caso viene chiamato underflow). Il bit in più che viene "buttato" dalla macchina spesso rappresenta il segno del numero (1 nel caso sia negativo) quindi nel trabocco il numero perde il segno diventando positivo quando magari ci si aspettava un negativo (o viceversa)*".

Risolvere questo esercizio senza utilizzare alcuna istruzione if-then-else.

- Creare un programma che accetta una stringa con il formato seguente come input:

  ```
  1024356 h S
  ```

  e converte 1024356 ore (h) in secondi (S). In particolare, l'unità di tempo di origine è in minuscolo, e l'unità di tempo in maiuscolo è l'unità di tempo di destinazione verso cui applica la conversione. Le stringhe che possono essere usate per esprimere tale unità di tempo sono le seguenti:

  - y/Y = Anno
  - d/D = Giorno
  - h/H = Ora
  - m/M = Minuti
  - s/S = Secondi

  Creare un programma che analizzi la stringa e fornisca la conversione di ora desiderata.

- Scrivere un programma che associa ad una coppia di numeri interi (i, j) un integer univoco c, tale che $(i, j) \to c$ è una biiezione. Alla fine, controllare che il risultato della funzione inversa corrisponde all'input fornito all'inizio i e j.

  Per fare questo, possiamo usare la funzione di dovetailing definita come segue:

  $$c = \frac{(i+j)(i+j+1)}{2} + j$$

  Quindi, vogliamo assicurarci che il numero risultante sia associato in modo univoco a (i, j). Allora, possiamo chiederci se c'è una funzione inversa. Tale funzione inversa è:

  $$j = c - \frac{1}{2}\left(\left\lfloor \frac{\sqrt{8c+1}-1}{2} \right\rfloor + 1\right) \cdot \left(\left\lfloor \frac{\sqrt{8c+1}-1}{2} \right\rfloor\right)$$

  $$i = \left(\left\lfloor \frac{\sqrt{8c+1}-1}{2} \right\rfloor\right) - j$$