Laboratorio di Programmazione a.a. 2023-24

Esercitazione sulla codifica in Java

27 Marzo 2024

Esercizio – Parte I

Dato un intero *non negativo*[1] in notazione ternaria bilanciata (stringa di cifre –/./+), la seguente procedura in Scheme restituisce la rappresentazione dell'intero successivo, calcolata operando direttamente sulle cifre a livello testuale:

```
(define btr-succ                              ; val: stringa di -/./+
  (lambda (btr)                               ; btr: stringa di -/./+
    (let ((n (string-length btr)))            ; (brt = "." oppure inizia con "+")
      (let ((lsb (string-ref btr (- n 1))))
        (if (= n 1)
            (if (char=? lsb #\+)
                "+-"
                "+")
            (let ((pre (substring btr 0 (- n 1))))
              (if (char=? lsb #\+)
                  (string-append (btr-succ pre) "-")
                  (string-append pre (if (char=? lsb #\-) "." "+"))
                  ))
            )))
    ))
```

Traduci in *Java* la procedura `btr-succ` e verifica sperimentalmente che i risultati siano consistenti con quelli ottenuti applicando la procedura riportata sopra nell'ambiente *DrRacket*.

Esercizio – Parte II

Considera il programma in Scheme per realizzare la procedura `ones-complement` che, data una stringa di bit, restituisce la rappresentazione del corrispondente complemento a uno, in cui le cifre `0` e `1` sono "scambiate" fra loro (sorgenti Scheme: `recursion_strings.scm` di cui è riportato il link anche a fianco di questo testo).

Prova a trasformare il programma per realizzarne una versione imperativa in Java, che non si avvalga della ricorsione ma applichi il costrutto `for` per l'iterazione.

Sperimenta infine il programma nell'ambiente *BlueJ* (oppure *DrJava*) e verificane i risultati.

1 La definizione di `btr-succ` riportata qui non si applica alle rappresentazioni ternarie bilanciate di numeri interi negativi, che richiederebbero un trattamento più complesso.