Specifica di Algoritmi in Pseudocodice

La specifica ad alto livello di un algoritmo viene solitamente fatta tramite pseudocodice che fornisce una descrizione chiara e sintetica dei passi elementari eseguiti dall'algoritmo per risolvere una data istanza di input. Lo pseudocodice utilizza solitamente costrutti tipici dei linguaggi di programmazione. In alcuni casi, per semplificare e rendere più efficace la descrizione, è possibile utilizzare il linguaggio naturale, purché da esso sia facile ricavare una specifica in un linguaggio di programmazione [1, Sect.1.8.2]. Lo pseudocodice di un algoritmo è strutturato come segue

Algoritmo *nome_algoritmo*(*parametri di input*)
input *descrizione dell'istanza di input*
output *descrizione della soluzione restituita*

descrizione dei passi elementari dell'algoritmo tramite mix di costrutti dei linguaggi di programmazione e di linguaggio naturale

L'intestazione dello pseoudocodice non deve contenere la descrizione del problema computazionale che l'algoritmo risolve, ma solo il nome dell'algoritmo e la relazione input-output. Si ricordi che un problema computazionale è definito da un insieme $\mathcal{I}$ di istanze, un insieme $\mathcal{S}$ di soluzioni, e un insieme $\Pi \subseteq \mathcal{I} \times \mathcal{S}$ che, per ogni istanza di input, definisce la/le soluzioni corrette. Un algoritmo che risolve un tale problema calcola una *funzione* che associa a ciascuna istanza $i \in \mathcal{I}$ una soluzione $s \in \mathcal{S}$, tale che $(i, s) \in \Pi$. Se in Π esistono più soluzioni per la stessa istanza, l'algoritmo ne calcola una sola. L'intestazione dello pseudocodice deve solo evidenziare la funzione $i \to s$ calcolata dall'algoritmo, specificando i ed s.

I costrutti dei linguaggi di programmazione più frequentemente utilizzati sono

- Assegnamento di un valore a una variabile, indicato con il simbolo "$\leftarrow$".
- Indicizzazione di array (ad es., $A[i]$)
- **if** (*condizione*) **then** $\{\cdots\}$ **else** $\{\cdots\}$. Il ramo "else" è opzionale.
- **for** *variabile* $\leftarrow$ *valore iniziale* **to/downto** *valore finale* **do** $\{\cdots\}$.
- **foreach** *elemento di un iteratore* **do** $\{\cdots\}$.
- **while** (*condizione*) **do** $\{\cdots\}$.
- **repeat** $\{\cdots\}$ **until** (*condizione*) .
- **return**, seguito, opzionalmente, da un valore o un insieme di valori.
- Invocazione di un algoritmo, magari lo stesso algoritmo, se ricorsivo.

Vi sono alcune regole di buon senso e convenzioni che è bene seguire.

- Bisogna dare dei nomi significativi agli algoritmi.
- Bisogna evitare di usare traduzioni in italiano dei costrutti dei linguaggi di programmazione.
- Il ricorso al linguaggio naturale deve essere limitato a quei casi in cui esso semplifica la descrizione e aiuta la comprensione dell'algoritmo, senza però offuscare la sequenza di operazioni. Negli altri casi, l'uso dei costrutti sopra elencati risulta essere più standard ed efficace.
- Nei cicli **for**, l'incremento della variabile di ciclo è automatico e non va indicato esplicitamente, a differenza dei cicli **while** e **repeat**, in cui le variabili che determinano il succedersi delle iterazioni vanno esplicitamente aggiornate.
- Ci sono tre tipologie di variabili utilizzate da un algoritmo:
  - *Variabili locali*, che devono essere definite e inizializzate dall'algoritmo e che non sopravvivono all'esecuzione dell'algoritmo.
  - *Variabili globali*, che si assumono definite nel processo chiamante e che sopravvivono all'esecuzione dell'algoritmo.
  - *Parametri di input* riportati tra parentesi vicino al nome dell'algoritmo. Solitamente, i valori iniziali di questi parametri sono forniti all'algoritmo dal processo chiamante. In alcuni casi, i parametri di input sono variabili globali che sopravvivono alla esecuzione dell'algoritmo. (Questa scelta è più flessibile rispetto a Java dove i parametri di input sono passati *by value*, ovvero diventano variabili locali che non sopravvivono alla esecuzione dell'algoritmo. Tuttavia, anche in Java, nel caso di parametri di input che contengono riferimenti a oggetti, modifiche a tali oggetti sopravvivono alla esecuzione dell'algoritmo.)
- Se un algoritmo non restituisce esplicitamente dei valori ma ha un effetto su dati globali, nella riga di descrizione dell'output nella intestazione si descrive tale effetto (ad es., si pensi all'ordinamento di una sequenza S).
- Un algoritmo ricorsivo contiene al suo interno una o più chiamate a se stesso su istanze più piccole. È importante che le chiamate vengano fatte utilizzando il nome dell'algoritmo, e che la scelta dei parametri di input da indicare tra parentesi vicino al nome dell'algoritmo definisca senza ambiguità l'istanza sulla quale l'algoritmo è invocato.

References

[1] M.T. Goodrich, R. Tamassia, M.H. Goldwasser. *Data Structures and Algorithms in Java*, Sixth Edition. John Wiley and Sons, 2014