0 Esercizi in pseudo-codice

Questa dispensa propone esercizi sulla scrittura di algoritmi in un linguaggio semi-formale, utile all'acquisizione delle abilità essenziali per implementare algoritmi in qualsiasi linguaggio di programmazione.

0.1 Algoritmi ed esecutori

Dato un problema ed un opportuno metodo risolutivo, la *risoluzione* di tale problema è quel processo che trasforma i *dati in ingresso* nei corrispondenti *dati finali*. Affinché la risoluzione di un problema possa essere realizzata attraverso l'uso di un calcolatore, il metodo risolutivo deve poter essere definito come una sequenza di azioni o istruzioni elementari, ovvero occorre codificare la risoluzione del problema in un algoritmo. Si dice *esecutore* quella macchina astratta capace eseguire le istruzioni specificate dall'algoritmo.

Un *algoritmo* è una sequenza finita di istruzioni, definita con precisione, che portano alla risoluzione di un compito. Un algoritmo è tale se possiede le seguenti proprietà:

- **Eseguibilità**: ogni istruzione deve essere eseguibile dal calcolatore in tempo finito;
- **Non-ambiguità**: ogni istruzione deve essere univocamente interpretabile dal calcolatore;
- **Finitezza**: il numero totale di azioni da eseguire, per ogni insieme di dati in ingresso, deve essere finito.

0.2 Il linguaggio "SIMITA"

Introduciamo uno pseudo–linguaggio di programmazione, che chiameremo SIMITA.[1] Vogliamo che questo linguaggio sia in grado di gestire:

[1] Se invece vi volete divertire con un linguaggio differente `https://github.com/esseks/monicelli`

- Dati singoli (sia quelli in ingresso, sia quelli ottenuti durante l'esecuzione del programma, sia quelli finali);
- Scelte tra alternative;
- Ripetizioni;
- Gruppi di dati.

Immaginiamo di poter gestire i valori dei dati su dei "foglietti". Sui di essi possiamo *scrivere*, *leggere*, *cancellare* e *riscrivere* (come nella memoria dei calcolatori). Per indicare scrittura di un numero N su foglietto f:

$f \leftarrow N$

Allo stesso modo indichiamo la scrittura del contenuto del foglietto g nel foglietto f come:

$f \leftarrow g$

In entrambi i casi l'operazione di scrittura cancellerà tutto ciò che era scritto sul foglietto f. Inoltre, disponiamo di altre espressioni per codificare delle funzionalità di cui dispone l'esecutore (nel nostro caso l'elaboratore):

- $leggi(f)$: indica l'operazione di lettura di un valore introdotto dall'utente (ad esempio, da tastiera) e la scrittura di tale valore nel foglietto f;
- $stampa(f)$: indica l'operazione di lettura del valore contenuto nel foglietto f e la stampa (ad esempio, a video) di tale valore. L'istruzione $stampa$ permette anche di stampare dei caratteri, ad esempio $stampa$("ciao come stai") stampa a video i caratteri "ciao come stai".

Gestiamo nel linguaggio SIMITA il costrutto condizionale, capace di esprimere il concetto di scelta tra alternative, nel seguente modo:

```
Istruzione 0

Se condizione Allora
    Istruzione 1
    Istruzione 2
Altrimenti
    Istruzione 3
    Istruzione 4
Chiudi Se
```

```
Istruzione 5
```

Nell'esempio precedente l'algoritmo prescrive al calcolatore di eseguire innanzitutto "Istruzione 0". Successivamente, esso valuta "condizione": se "condizione" risulta vera, allora vengono eseguite "Istruzione 1" ed "Istruzione 2", in caso contrario vengono eseguite "Istruzione 3" ed "Istruzione 4". Al termine viene eseguita "Istruzione 5". Il blocco "**Altrimenti**" è opzionale: se assente, nel caso in cui "condizione" risulti falsa l'esecuzione passa direttamente ad "Istruzione 5". Non vi è limite al numero di istruzioni specificate in ogni blocco del costrutto condizionale.

Per poter eseguire più volte una stessa operazione, in SIMITA, abbiamo un costrutto ciclico (o iterativo), capace di esprimere il concetto di ripetitività:

```
Istruzione 0        //questo e' un commento

Finché condizione Esegui
    Istruzione 1
    Istruzione 2
Chiudi Finché

Istruzione 3
```

Nell'esempio precedente la sequenza di istruzioni 1 e 2 viene eseguita un numero *finito* di volte. L' "Istruzione 0" viene eseguita una sola volta. Dopodiché viene valutata "condizione": se risulta vera, allora vengono eseguite "Istruzione 1" ed "Istruzione 2". Viene poi valutata nuovamente "condizione": se vera, si ripete l'esecuzione di "Istruzione 1" ed "Istruzione 2". L'iterazione *termina*, e quindi non vengono più eseguite "Istruzione 1" ed "Istruzione 2", solo quando "condizione" diventa falsa. Quando questo avviene, si esce dal ciclo e si esegue "Istruzione 3". Solitamente, al fine di evitare che il costrutto ciclico non abbia fine, è buona norma che le istruzioni all'interno del ciclo modifichino i foglietti che vengono valutati in "condizione".

Sempre nell'esempi precedente è stata inserita una porzione di istruzione, detta *commento*, che non verrà considerata dall'esecutore, ossia "questo e' un commento". Solitamente un commento viene inserito per descrivere in linguaggio naturale ciò che le istruzioni specificano e quindi per chiarificare a chi legge le istruzioni il compito svolto dall'algoritmo.

SIMITA ha anche la nozione di *blocchi di foglietti*:

$B[0]$	$B[1]$	$B[2]$	$B[3]$	foglietti
0	1	2	3	indice

$B[0]$ indica il primo foglietto del blocco;

$B[1]$ indica il secondo foglietto del blocco;

...

$B[n-1]$ indica l'n-simo foglietto del blocco.

Si noti che ogni foglietto $B[n]$ è, a tutti gli effetti, un normale foglietto. Quindi, ad esempio, le seguenti istruzioni sono valide:

```
i ← 1
f ← B[i]        //scrive il contenuto del secondo foglietto
                //B[1] in f

B[i] ← 3        //scrive 3 nel secondo foglietto B[1]

stampa(B[4])      //stampa il contenuto del quinto foglietto

leggi(B[i])      //legge un dato
                  //e lo scrive nel secondo foglietto
```

0.3 Esercizi

Esercizio 0.1

Scrivere in linguaggio SIMITA l'algoritmo per descrivere il problema di svegliarsi ed uscire di casa, sapendo che:

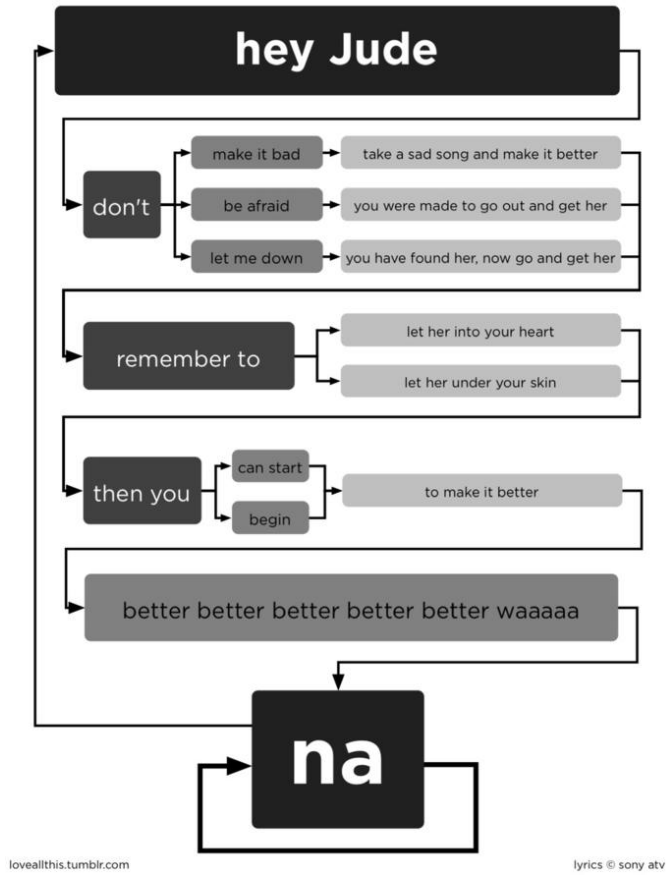
- Una volta sveglio si deve spegnere la sveglia (`spegniSveglia()`);
- Viviamo in una casa di 3 piani e prima di uscire dobbiamo chiudere una finestra (se è aperta) ad ogni piano;
- Dormiamo al secondo piano, dove c'è il bagno, in cui ci dobbiamo fare la doccia (`doccia()`);
- Al primo piano c'è la cucina dove dobbiamo fare colazione (`colazione()`), se è domenica facciamo una colazione sostanziosa (`colazioneSostanziosa()`);
- Al piano terreno c'è la porta di uscita che dobbiamo aprire e chiudere.

L'utente ci comunica prima dell'inizio delle operazioni della sveglia quali sono le finestre aperte e se è domenica. Ogni volta che eseguiamo un'azione dobbiamo comunicarla all'utente stampandola a video.

Esercizio 0.2

Dire se il flowchart nella seguente figura è da considerarsi un algoritmo e, nel caso non lo sia, spiegare come poterlo modificare perchè lo diventi.

Scrivere lo pseudo-codice in SIMITA dell'algoritmo così ottenuto.



Esercizio 0.3

Calcolare il prodotto di due numeri interi positivi e stamparlo a video. L'esecutore è in grado di eseguire solamente somme e sottrazioni.

Esercizio 0.4

Calcolare il fattoriale di un numero intero non negativo e stamparlo a video. L'esecutore è in grado di eseguire moltiplicazioni e sottrazioni.

Esercizio 0.5

1. Scrivere un algoritmo per valutare e stampare a schermo se un numero intero positivo è pari o dispari utilizzando solo le 4 operazioni fondamentali.

2. Scrivere il medesimo algoritmo supponendo di avere un operatore div che restituisce il risultato intero della divisione (ad esempio, $7\ div\ 3$ ha come risultato 2).

Esercizio 0.6

1. Scrivere un programma che data una sequenza di numeri (positivi) in ingresso restituisce il maggiore e la sua posizione nella sequenza. Supponiamo che la sequenza abbia 10 numeri.

2. Si provi ad implementare un algoritmo analogo a quello dell'esercizio precedente, utilizzando un solo ciclo ed eliminando l'assunzione che i numeri inseriti siano positivi.

3. Si estenda l'algoritmo per calcolare la media m dei valori inseriti.

4. Si estenda l'algoritmo per calcolare e stampare a video la differenza $N[i] - m$ tra ogni elemento della sequenza e la media della sequenza.

5. È possibile implementare questi algoritmi senza l'uso di blocchi di foglietti?

6. Si estenda l'algoritmo per calcolare anche la stima della deviazione standard della sequenza, ovvero:

$$\sigma(x_1, \ldots, x_N) = \frac{\sum_{i=1}^{N}(x_i - \bar{x})^2}{N-1}$$

   dove $\bar{x}$ è la stima della media, supponendo di avere a disposizione una funzione che calcola la radice quadrata di un numero (`sqrt()`). È possibile risolvere questo problema senza ricorrere ai blocchi di fogli?

Esercizio 0.7

Determinare se una funzione continua ha uno zero nell'intervallo $[a, b]$, dove gli estremi sono forniti dall'utente. Si faccia ricorso al seguente teorema (teorema di Bolzano):

- Ipotesi: $f : [a, b] \mapsto \mathbb{R}$, continua in $[a, b] \subseteq \mathbb{R}$, tale che $f(a) \cdot f(b) < 0$.

- Tesi: $\exists c \in (a, b)$ tale che $f(c) = 0$ (zero di f).

Per valutare la funzione f in un punto generico a potete utilizzare l'operatore

$$g \leftarrow calcola(f, a),$$

che scrive il risultato di $f(a)$ sul foglietto g.

Esercizio 0.8

Determinare almeno uno zero di una funzione continua f nell'intervallo $[a, b]$. Il calcolatore dispone dell'istruzione $calcola(f, a)$ che valuta la funzione f nel punto a.

Esercizio 0.9

Calcolare la radice quadrata intera di un numero intero positivo e stamparla a video. Per radice quadrata intera di un numero n si intende un numero m tale che $m^2 \leq n$ e $(m + 1)^2 > n$.