1 Esercizi in pseudocodice

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

1.1 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*.

Si dice *esecutore* quella macchina astratta capace eseguire le istruzioni specificate dall'algoritmo.

Affinché la risoluzione di un problema possa essere realizzata attraverso l'uso di un calcolatore come esecutore, il metodo risolutivo deve poter essere definito come una sequenza di azioni-istruzioni elementari: occorre codificare la soluzione del problema in un algoritmo.

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

Eseguibilità Ogni azione dev'essere eseguibile dall'esecutore in tempo finito.

Non-ambiguità Ogni azione dev'essere univocamente interpretabile dall'esecutore.

Finitezza Il numero totale di azioni da eseguire, per ogni insieme di dati in ingresso, dev'essere finito.

1.2 Il linguaggio "SIMITA"

Introduciamo uno pseudolinguaggio di programmazione, cioè l'italiano strutturato per ripetizioni e scelte tra alternative, che chiameremo SIMITA.

Immaginiamo di poter scrivere i valori dei dati (sia quelli in ingresso, sia quelli ottenuti durante l'esecuzione del programma) in dei "foglietti".

Sui foglietti 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:

$f \leftarrow g$

Altre espressioni per codificare delle funzionalità di cui dispone 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".

SIMITA possiede il costrutto condizionale:

```
Istruzione 0

Se condizione allora
    Istruzione 1
    Istruzione 2
    ...
Altrimenti
    Istruzione 3
    Istruzione 4
    ...
chiudi Se
```

```
Istruzione N
```

Dopo aver eseguito "Istruzione 0", viene valutata "condizione". Se "condizione" risulta vera, allora vengono eseguite "Istruzione 1", "Istruzione 2", etc. In caso contrario vengono eseguite solamente "Istruzione 3", "Istruzione 4", etc. Al termine viene eseguita "Istruzione N". Il blocco "**Altrimenti**" è opzionale: se assente, l'esecuzione passa direttamente ad "Istruzione N".

SIMITA possiede anche i costrutti ciclici:

```
Istruzione 0        //questo è un commento

Finché condizione esegui
    Istruzione 1
    Istruzione 2
    ...
chiudi Finché

Istruzione N
```

La sequenza di istruzioni 1 e 2 viene eseguita un numero *finito* di volte. Solo alla prima passata viene eseguita "Istruzione 0". Poi viene valutata "condizione": se risulta vera, allora vengono eseguite "Istruzione 1", "Istruzione 2", etc. In ogni caso viene rivalutata "condizione": se vera, si ripete l'esecuzione di "Istruzione 1" e "Istruzione 2", etc. L'iterazione *termina*, e quindi non vengono più eseguite "Istruzione 1" ed "Istruzione 2", solo quando "condizione" diviene falsa. Quando questo avviene, si esce dal ciclo e si esegue "Istruzione N".

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

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

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

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

...

$B[n]$ indica l'n-esimo foglietto del blocco

Si noti che i singoli foglietti sono, a tutti gli effetti, dei normali foglietti. Quindi, ad

esempio, le istruzioni seguenti sono valide:

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

B[i] ← 3        //scrive 3 nel primo foglietto

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

leggi(B[i])       //legge un dato e lo scrive nel foglietto 1
```

1.3 Esercizi

Esercizio 1.1

1. Calcolare il prodotto di due numeri interi e stamparlo a video. Il vostro calcolatore è in grado di eseguire solamente somme.

2. Produrre una variante ottimizzata dell'algoritmo proposto nella parte precedente.

Esercizio 1.2

Calcolare il fattoriale di un numero e stamparlo a video.

Esercizio 1.3

Calcolare la radice quadrata intera di un numero intero (assunto maggiore di 10) e stamparla a video.

Esercizio 1.4

Determinare se una funzione continua ha uno zero nell'intervallo $[a, b]$ conoscendo il seguente teorema (teorema di Bolzano):

Ipotesi: $f : [a, b] \mapsto \mathbb{R}$, continua in $[a, b] \subseteq \mathcal{R}$, 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'istruzione $calcola(f, a, g)$, che scrive il risultato sul foglietto g.

Esercizio 1.5

Determinare almeno uno zero di una funzione continua nell'intervallo $[a, b]$.

Esercizio 1.6

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.

Esercizio 1.7

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.

1. Si estenda l'algoritmo per calcolare la media m dei valori inseriti.
2. 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.
3. È possibile implementare questi algoritmi senza l'uso di blocchi di foglietti?
4. Si estenda l'algoritmo per calcolare anche la deviazione standard della sequenza, supponendo di avere a disposizione una funzione che calcola la radice quadrata di un numero. È possibile risolvere questo problema senza ricorrere ai blocchi di fogli?

1.3.1 Soluzioni

Soluzione dell'esercizio 1.1

1. Dopo aver letto i due numeri ed averli memorizzati in due "foglietti" distinti, a e b, calcoliamo il prodotto in modo cumulativo utilizzando un ciclo.

```
leggi(f)        //leggi il primo numero
leggi(g)        //leggi il secondo numero

risultato ← 0

Finché g > 0 esegui
    risultato ← risultato + f        //aggiungi f per g volte
    g ← g − 1
chiudi Finché

stampa(risultato)
```

2. Una caratteristica fondamentale degli algoritmi è l'efficienza, ci aspettiamo che le soluzioni devono essere fornite nel modo più veloce possibile.

```
leggi(f)        //leggi il primo numero
leggi(g)        //leggi il secondo numero
risultato ← 0

Se g < f allora
    Finché g > 0 esegui
        risultato ← risultato + f        //aggiungi f per g volte
        g ← g − 1
    chiudi Finché
Altrimenti
    Finché f > 0 esegui
        risultato ← risultato + g        //aggiungi g per f volte
        f ← f − 1
    chiudi Finché
chiudi Se

stampa(risultato)
```

Si noti che è possibile produrre un algoritmo analogo, comunque ottimizzato, ma con meno istruzioni.

```
leggi(f)        //leggi il primo numero
leggi(g)        //leggi il secondo numero

risultato ← 0

Se g < f allora
    min ← g
Altrimenti
    min ← f
chiudi Se

Finché min > 0 esegui
    risultato ← risultato + f        //aggiungi f per min volte
    min ← min − 1
chiudi Finché

stampa(risultato)
```

L'efficienza di un algoritmo si misura anche sulla base della memoria consumata; si preferiscono algoritmi che impiegno meno memoria possibile. Abbiamo introdotto 4 variabili, non sono troppe?

```
leggi(f)        //leggi il primo numero
leggi(g)        //leggi il secondo numero

tmp ← 0

Se g < f allora        //assicuriamoci che f < g
    tmp ← f        //metto f su un foglietto temporaneo
    f ← g        //g contiene il numero minore
    g ← tmp
Altrimenti
    tmp ← g        //l'altro lo scrivo in g
chiudi Se

Finché f > 1 esegui        //ripeti la somma f − 1 volte
    tmp ← tmp + g        //aggiungi f per min volte
    f ← f − 1
chiudi Finché

stampa(risultato)
```

Soluzione dell'esercizio 1.2

Si legge il numero e lo si scrive in f, si esegue un ciclo moltiplicando il fattoriale (inizialmente pari ad f) ad f per un numero di volte pari ad $f - 1$.

```
leggi(f)
fattoriale ← f

Finché f > 1 esegui
    f ← f − 1
    fattoriale ← fattoriale ∗ f
chiudi Finché

stampa(fattoriale)
```

Soluzione dell'esercizio 1.3

Si legge il radicando e lo si scrive in r. Partendo da $r/2$ si cerca quel numero x tale per cui $x^2 = r$ (condizione di ciclo).

```
leggi(r)
x = r/2

Finché x * x > r esegui
    x ← x − 1
chiudi Finché
```

Soluzione dell'esercizio 1.4

```
leggi(a)        //lettura estremo inferiore
leggi(b)        //lettura estremo superiore

calcola(f, a, A)
calcola(f, b, B)

Se A * B < 0 allora
    stampa("esiste almeno uno zero")
Altrimenti
    stampa("potrebbe non esistere alcuno zero")
chiudi Se
```

Soluzione dell'esercizio 1.5

```
leggi(a)        //lettura estremo inferiore
leggi(b)        //lettura estremo superiore

calcola(f, a, A)
calcola(f, b, B)

Se A * B > 0 allora
    stampa("potrebbe non esistere alcuno zero")
Altrimenti
    K ← 1

    Finché K ≠ 0 esegui
        c ← (b − a)/2
        calcola(f, c, K)

        Se A * K < 0 allora
            b ← c
        chiudi Se

        Se B * K < 0 allora
            a ← c
        chiudi Se
    chiudi Finché
chiudi Se

stampa(c)
```

Attenzione: Il programma potrebbe non terminare nel caso in cui $c \leftarrow (b-a)/2$ non sia esattamente il valore dello zero. Questo perché, essendo in $\mathbb{R}$, ci si può avvicinare sempre di più allo zero senza mai arrivarci esattamente.

Inoltre, si osservi che il programma non gestisce il caso in cui lo zero c sia $c = a$ o $c = b$.

Soluzione dell'esercizio 1.6

```
max ← 0
i ← 0
p ← i

Finché i ≤ 10 esegui
    leggi(N[i])
    i ← i + 1
chiudi Finché

i ← 0

Finché i < 10 esegui
    Se N[i] > max allora
        max ← N[i]
        p ← i
    chiudi Se
    i ← i + 1
chiudi Finché

stampa("il valore massimo è:")
stampa(max)

stampa("ed è in posizione:")
stampa(p)
```

Soluzione dell'esercizio 1.7

Prima di tutto notiamo che il primo ciclo, quello di acquisizione, memorizza i numeri nel blocco di foglietti. Tuttavia, il secondo ciclo, non fa altro che esaminarli (nello stesso ordine). Quindi, è possibile eliminare del tutto la necessità di memorizzare i numeri, e svolgere le istruzioni per il controllo "maggiore di" subito dopo l'acquisizione.

Inoltre, eliminando l'assunzione di numeri positivi, non abbiamo un estremo inferiore (zero). Perciò è necessario inizializzare il massimo al primo valore incontrato (che potrebbe essere inferiore a zero).

```
i ← 0
p ← i
n ← 0      //foglietto per il numero letto

Finché i < 10 esegui
    leggi(n)

    Se i == 0 ∧ n > max allora      //primo numero o maggiore
        max ← n      //salvo il valore massimo
        p ← i      //salvo la posizione
    chiudi Se

    i ← i + 1      //incremento il contatore
chiudi Finché

stampa("il valore massimo è:")
stampa(max)

stampa("ed è in posizione:")
stampa(p)
```

1. Il calcolo della media richiede che, durante l'acquisizione, si calcoli anche la somma incrementale di tutti i numeri letti. Al termine del ciclo, tale somma verrà divisa per il contatore.

```
i ← 0
p ← i
media ← 0
n ← 0      //foglietto per il numero letto

Finché i < 10 esegui
    leggi(n)

    Se i == 0 ∧ n > max allora      //primo numero o maggiore
        max ← n      //salvo il valore massimo
        p ← i      //salvo la posizione
    chiudi Se

    media ← media + n      //somma incrementale
    i ← i + 1      //incremento il contatore
chiudi Finché

media ← media/(i + 1)

stampa("il valore massimo è:")
stampa(max)

stampa("ed è in posizione:")
stampa(p)

stampa("il valore medio è:")
stampa(media)
```

2. Per calcolare lo scarto dalla media di ogni elemento è necessario memorizare tutti gli elementi, perché la media sarà calcolabile solo dopo aver letto tutti i numeri. Quindi:

```
i ← 0
p ← i
media ← 0
n ← 0      //foglietto per il numero letto

Finché i < 10 esegui
    leggi(n)

    Se i == 0 ∧ n > max allora      //primo numero o maggiore
        max ← n      //salvo il valore massimo
        p ← i      //salvo la posizione
    chiudi Se

    N[i] ← n      //memorizzazione elemento i-esimo
    media ← media + n      //somma incrementale
    i ← i + 1      //incremento il contatore
chiudi Finché

media ← media/(i + 1)
i ← 0

Finché i < 10 esegui      //ciclo su tutti i foglietti
    stampa("lo scarto dalla media di ")
    stampa(N[i])
    stampa("è: ")
    stampa(N[i] − media)      //scarto dalla media
    i ← i + 1
chiudi Finché

stampa("il valore massimo è:")
stampa(max)

stampa("ed è in posizione:")
stampa(p)

stampa("il valore medio è:")
stampa(media)
```

3. No, per i motivi spiegati nella precedente risposta.

4. La deviazione standard è definita come la media degli scarti quadratici dalla media, ovvero:

$$\mathrm{devstd}(\{x_1, \ldots, x_M\}) = \sqrt{\frac{\sum_{i=0}^{M}(x_i - m)^2}{M}}$$

si deve quindi calcolare la somma degli scarti quadratici $N[i] - m$, dividere tale numero per il contatore ed estrarne la radice quadrata.

```
i ← 0
p ← i
media ← 0
devstd ← 0
n ← 0      //foglietto per il numero letto

Finché i < 10 esegui
    leggi(n)

    Se i == 0 ∧ n > max allora      //primo numero o maggiore
        max ← n      //salvo il valore massimo
        p ← i      //salvo la posizione
    chiudi Se

    N[i] ← n      //memorizzazione elemento i-esimo
    media ← media + n      //somma incrementale
    i ← i + 1      //incremento il contatore
chiudi Finché

media ← media/(i + 1)
i ← 0

Finché i < 10 esegui      //ciclo su tutti i foglietti
    scarto ← (N[i] − media)
    scarto ← scarto ∗ scarto      //scarto quadratico
    devstd ← devstd + scarto      //somma scarti
    i ← i + 1
chiudi Finché

devstd ← devstd/(i + 1)      //media somma scarti
devstd ← sqrt(devstd)      //deviazione standard

stampa("il valore massimo è:")
stampa(max)

stampa("ed è in posizione:")
stampa(p)

stampa("il valore medio è:")
stampa(media)

stampa("la deviazione standard è:")
stampa(devstd)
```

Come nel caso degli scarti dalla media, anche in questo caso non è possibile risolvere questo problema senza poter memorizzare tutta la sequenza di numeri.